МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

ПРОГРАММНЫЙ СЕРВИС ВЕДЕНИЯ УЧЕБНОГО ПРОЦЕССА

Студент _____ Кочетов Денис Витальевич
(подпись, дата) (ФИО полностью)

Руководитель _____ А.В. Дубравин
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ А.С. Бычков
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 20.06.23 г. № 14)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой Отлично (протокол заседания ГЭК от 22.06.2023 № 13)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«24» 06 2023 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Кочетов Денис Витальевич гр. 19ВВВ1 факультета вычислительной техники.

2.Тема работы: «Программный сервис ведения учебного процесса»

Тема утверждена приказом ПГУ № 458/о от "26" апреля 2023 г.

3. Руководитель бакалаврской работы к.т.н., доцент каф. ВТ ПГУ Дубравин А.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение «Программный сервис ведения учебного процесса», позволяющее участникам образовательного процесса получать информацию о ходе и результатах учебного процесса. Приложение должно обладать графическим интуитивно понятным интерфейсом для взаимодействия с пользователем. Система должна предоставлять возможность для взаимодействия с информацией об учителях, обучающихся, учебных классах, домашнем задании, оценках.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить существующие программные сервисы, выделить их достоинства и недостатки. Выбрать язык программирования, который подойдет для создания графического интерфейса. Определить архитектуру приложения и вспомогательные средства разработки. Изучить функционал библиотек для взаимодействия с СУБД, а также изучить методы для организации сетевого взаимодействия программ.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	26.04.2023 – 29.04.2023	
2	Анализ требований	32	30.04.2023 – 03.05.2023	
3	Разработка моделей базы данных	30	04.05.2023 – 07.05.2023	
4	Выбор технологий, применяемых при проектировании и разработке	40	07.05.2023 – 13.05.2023	
5	Реализация графического интерфейса	66	13.05.2023 – 21.05.2023	
6	Разработка основной части программы	70	22.05.2023 – 26.05.2023	
7	Оформление пояснительной записки	54	26.05.2023 – 31.05.2023	

Дата выдачи задания "26" апреля 2023 г.

Руководитель бакалаврской работы _____ Дубравин А.В.
(подпись, дата)

Задание к исполнению принял студент _____ Кочетов Д.В.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)

27.06.2023

Реферат

Бакалаврская работа содержит 201 лист, 28 рисунков 1 таблицу, 19 источников, 43 приложения

ПРОГРАММНЫЙ СЕРВИС, ЭЛЕКТРОННЫЙ ДНЕВНИК, ЭЛЕКТРОННЫЙ ЖУРНАЛ, JAVA, КЛИЕНТ-СЕРВЕР, SWING

Цель работы – разработать программный комплекс ведения учебного процесса школы, позволяющий вести учёт хода и результатов учебного процесса.

Объектом разработки является комплекс программ, передающий информацию между клиентом и сервером, а также сохраняющий её. Клиентское приложение должно обладать графическим пользовательским интерфейсом.

Разработка проводилась в среде программирования *NetBeans IDE* 8.2. В качестве языка программирования был выбран *Java*.

В результате проведенной работы разработано клиент-серверное приложение, удовлетворяющее поставленной задаче, а именно: удобный пользовательский интерфейс, возможность учёта хода и результатов учебного процесса, сохранение данных.

Содержание

Реферат 2
Содержание 3
Перечень сокращений и обозначений 5
Введение 6
1 Программный сервис ведения учебного процесса 7
1.1 Анализ предметной области 7
1.2 Анализ существующих аналогов 8
1.2.1 Дневник.ру 8
1.2.2 Баллов.Нет 13
1.2.3 СтадиАппс 14
1.3 Выбор языка программирования 16
1.3.1 Язык программирования *C++* 16
1.3.2 Язык программирования *C#* 16
1.3.3 Язык программирования *Python* 17
1.3.4 Язык программирования *Java* 18
1.4 Хранение данных 19
1.5 Выбор СУБД 20
1.5.1 СУБД *PostgreSQL* 20
1.5.2 СУБД *SQLite* 20
1.5.3 СУБД *MySQL* 20
1.6 Выводы 21
1.7 Выбор инструментов разработки 22
1.7.1 Выбор среды разработки 22
1.7.2 Выбор платформы для реализации графического интерфейса 22
1.7.3 Выбор инструмента проектирования БД 23
1.7.3 Выбор инструмента для взаимодействия с СУБД *MySQL* 24
1.8 Архитектура программного сервиса 25
1.8.1 Архитектура 25
1.8.2 Организация сервера 28

1.9 Разработка программного сервиса 29
1.9.1 Разработка сетевого протокола 29
1.9.2 Проектирование базы данных 30
1.9.3 Разработка графического интерфейса 32
1.10 Описание программного сервиса 36
1.10.1 Руководство пользователя 36
1.10.1.1 Использование вспомогательной программы «*DbInit*» 36
1.10.1.2 Запуск сервера 37
1.10.1.3 Работа с клиентской программой 38
1.10.1.4 Руководство пользователя 39
2 Охрана труда 42
2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения 42
2.1.1 Оборудование рабочего места 42
2.1.2 Освещенность рабочего места 44
2.1.3 Электробезопасность 45
2.1.4 Пожарная безопасность 45
2.1.5 Режим труда и отдыха 46
Заключение 48
Список использованных источников 49
Приложение А – Диаграммы 51
Приложение Б – Листинг программы 57
Приложение В – Презентация 184

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ПК – персональный компьютер

ПО – программное обеспечение

ПЭВМ – персональная электронно-вычислительная машина

Сокет – программная сущность (объект), представляющая собой конечную точку сетевого соединения

СУБД – система управления базами данных

Транзакция – группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными

ЭД – электронный дневник

ЭЖ – электронный журнал

ACID (Atomicity, Consistency, Isolation, Durability) – набор требований к транзакциям в СУБД, обеспечивающий наиболее надёжную и предсказуемую её работу

Connector/J – реализация *JDBC* для *MySQL*

GNU General Public License – лицензия на свободное ПО

Java – язык программирования

JDBC (*Java Database Connectivity*) – платформенно независимый промышленный стандарт взаимодействия *Java*-приложений с различными СУБД

JDK (*Java Development Kit*) – комплект разработчика приложений на языке *Java*

JVM (*Java Virtual Machine*) – виртуальная машина *Java*, платформа, на которой исполняются программы, написанные на *Java*

MVC (*Model-View-Controller*) – шаблон проектирования приложений с графическим интерфейсом

SQL (*Structured Query Language*) – язык структурированных запросов, применяемый для создания и модификации данных, а также управления ими в БД

Swing – библиотека для создания графического интерфейса для программ на языке *Java*

Введение

Начало XXI века ознаменовано развитием информационных технологий, самыми востребованными из которых оказались сетевые, что доказывается их бурным непрерывным развитием. Возможность передавать информацию быстро и беспрепятственно сегодня используется людьми во всех сферах жизни.

Информатизация позволила начать переход от ведения учёта на бумаге к ведению его в электронном виде на компьютерах и иных цифровых устройствах. Не осталась в стороне и сфера образования. Информатизация образовательного процесса упрощает коммуникацию между его участниками, делает образовательную среду более комфортной. И хотя полый отказ от бумажных носителей пока не произошёл, его наступление сильно упростит работу учителям и учёбу обучающимся, ведь всю необходимую информацию можно будет найти в одном месте с применением компьютерных технологий.

Целью выпускной работы является разработка системы программ клиент-серверной архитектуры для ведения учебного процесса.

В ходе данной работы необходимо разработать приложение, позволяющее участникам образовательного процесса получать информацию о ходе и результатах учебного процесса. Приложение должно обладать графическим интуитивно понятным интерфейсом для взаимодействия с пользователем. Система должна предоставлять возможность для взаимодействия с информацией об учителях, обучающихся, учебных классах, домашнем задании, оценках. Помимо этого, система должна хранить информацию о ходе и участниках образовательного процесса.

1 Программный сервис ведения учебного процесса

1.1 Анализ предметной области

Для ведения учебного процесса и учёта его результатов в общеобразовательных учреждениях используются классные журналы – книги внушительных размеров, каждая из которых отвечает за один класс в течение одного года, а обучающимся приходится иметь дневники: покупать новые каждый год, тратить время на их заполнение и носить с собой каждый учебный день. Замена бумажных средств ведения учебного процесса электронными позволит упростить и ускорить работу как учителям, так и обучающимся.

В России уже активно ведётся внедрение электронных систем ведения журналов успеваемости в образовательный процесс. Минобрнауки дало такое определение понятиям «электронный журнал» и «электронный дневник»:

- электронный журнал – ПО или электронные сервисы, обеспечивающие учет выполнения учебной программы, в том числе успеваемости и посещаемости обучающихся [1];
- электронный дневник – ПО или электронные сервисы, обеспечивающие в электронном виде информирование обучающихся и их родителей (законных представителей) о ходе и результатах учебного процесса [1].

Исходя из этих определений, можно составить минимальный список функций, которые должен реализовывать программный сервис:

- возможность ведения успеваемости обучающихся;
- возможность учёта посещения занятий;
- возможность информирования самих обучающихся и их родителей (законных представителей) о ходе и результатах учебного процесса.

1.2 Анализ существующих аналогов

Среди сервисов, позволяющих вести учебный процесс, можно выделить следующие:

- Дневник.ру;
- Баллов.Нет;
- СтадиАппс.

1.2.1 Дневник.ру

Согласно описанию сервиса [2], из основных особенностей данного аналога можно выделить систему доступов, которая определяет доступный пользователю функционал. Всего имеется четыре уровня доступов.

Преподаватель имеет возможность вносить изменения в расписание занятий и формировать его, а также вести электронный журнал и управлять домашним заданием. Помимо этого, есть формировать отчёты, включающая в себя ведомости и отчёты образовательной организации [2]. Графический интерфейс сервиса меню преподавателя представлен на рисунках 1.1 и 1.2.

Профиль родителей даёт пользователю просматривать дневник обучающегося, а также получать ежедневные уведомления о ходе учебного процесса на мобильный телефон или электронную почту [3]. Графический интерфейс сервиса меню родителей представлен на рисунках 1.3 и 1.4.

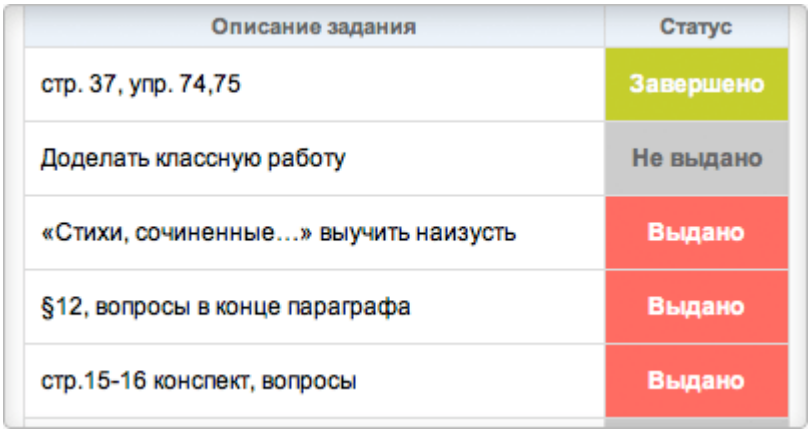


Рисунок 1.1 – Меню домашнего задания

		14	17		
		ОТВ	ОТВ	С/Р	
1	Сов Игорь	4	3	4	
2	Абрамов Даниил	5		5	Н
3	Абрамова Александра	Н	5	3	Б
4	Абрамова Елизавета	Б	4	2	
5	Амосова Вера	Н			Н

Рисунок 1.2 – Меню журнала

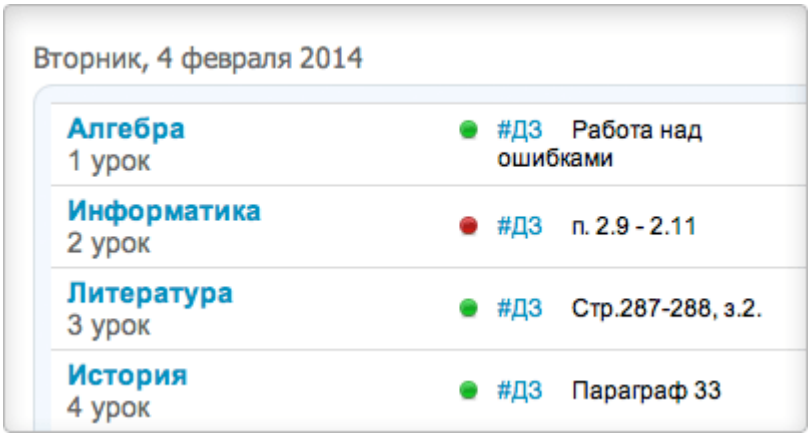
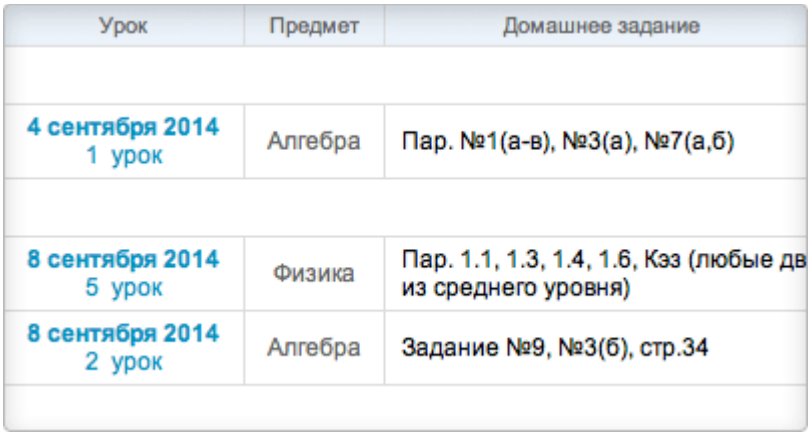


Рисунок 1.3 – Меню дневника профиля родителей



Урок	Предмет	Домашнее задание
4 сентября 2014 1 урок	Алгебра	Пар. №1(а-в), №3(а), №7(а,б)
8 сентября 2014 5 урок	Физика	Пар. 1.1, 1.3, 1.4, 1.6, Кэз (любые дв из среднего уровня)
8 сентября 2014 2 урок	Алгебра	Задание №9, №3(б), стр.34

Рисунок 1.4 – Меню просмотра домашнего задания профиля родителей

Обучающийся может просматривать расписание занятий, домашнее задание и оценки в меню дневника, а также имеет доступ к различным образовательным материалам. Помимо этого, у этого типа профиля есть возможность формировать

своё портфолио [4]. Графический интерфейс сервиса меню обучающегося представлен на рисунке 1.5.

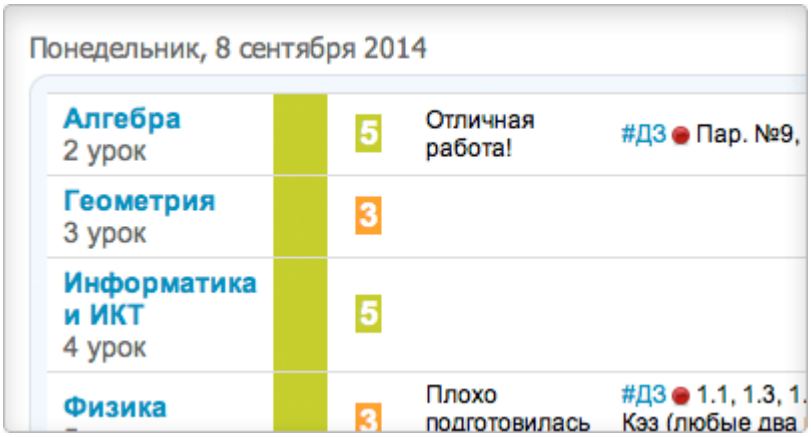


Рисунок 1.5 – Меню просмотра дневника профиля обучающегося

Помимо вышеназванных, сервис предоставляет возможности для государственных учреждений, обеспечивающих образовательную деятельность. Среди них:

- реализация госуслуг в сфере образования;
- сбор статистики;
- мониторинг подведомственных организаций;
- анализ данных, полученных в результате мониторинга;
- массовое информирование родителей, обучающихся или руководителей подведомственных организаций [5].

Графический интерфейс сервиса меню госучреждения представлен на рисунках 1.6 – 1.10.

Из преимуществ данного сервиса можно выделить следующие:

- большой функционал;
- наличие профилей доступа, которые позволяют разграничить взаимодействие с системой;
- удобный и опрятный интерфейс;
- интеграция со сторонними сервисами, в частности, электронной почтой и сетью мобильной связи.

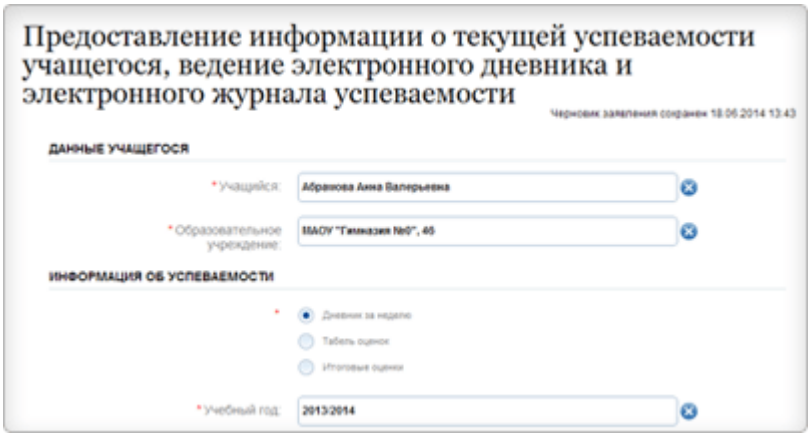


Рисунок 1.6 – Меню для интеграции с сервисом госуслуг

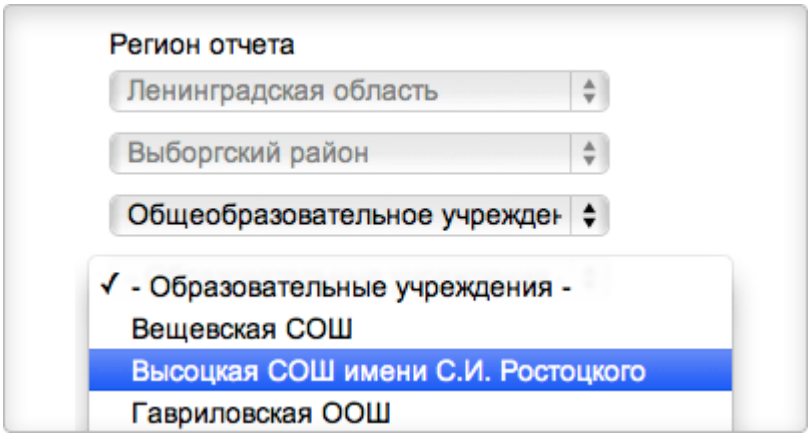


Рисунок 1.7 – Меню задания параметров для сбора статистики

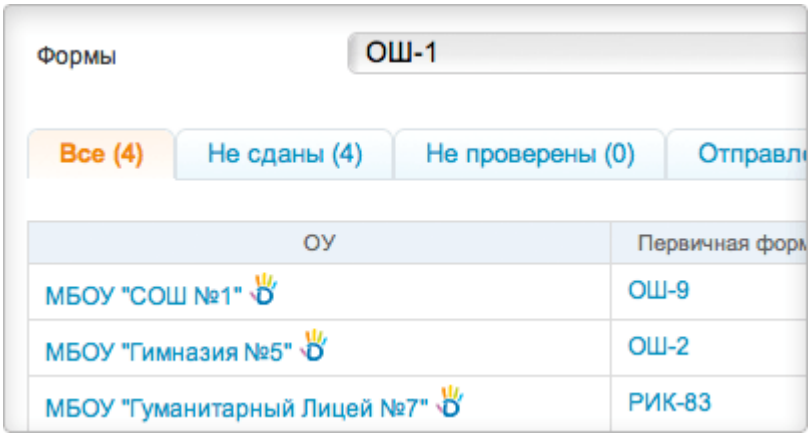


Рисунок 1.8 – Меню модуля мониторинга

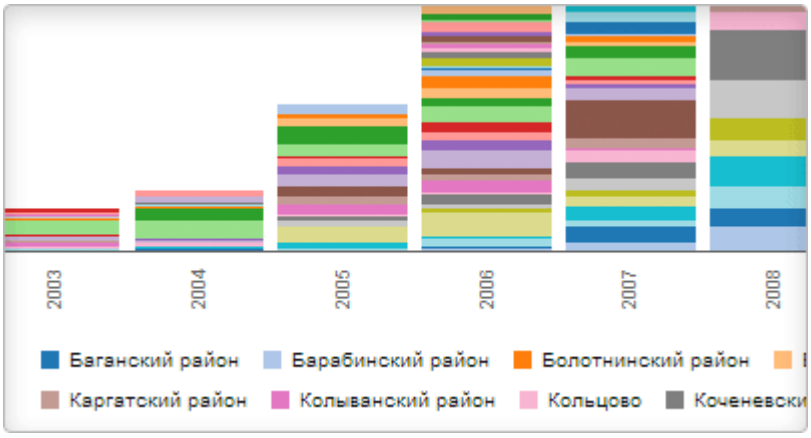


Рисунок 1.9 – Меню просмотра результатов анализа данных

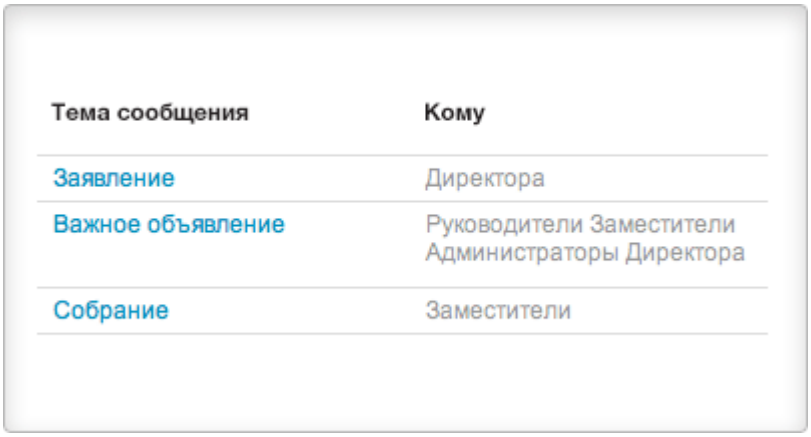


Рисунок 1.10 – Меню для просмотра оповещения массового информирования

Из недостатков можно выделить следующие:

- отсутствие профиля администратора образовательного учреждения; фактически, эта роль перекладывается на преподавателя;
- закрытая система: для пользования организации нужно пройти весь процесс регистрации образовательного учреждения;
- сторонний сервис: пользование платформой полностью передаёт ведение данных образовательного учреждения на сторону, что создаёт потенциальную угрозу их доступности, целостности и конфиденциальности;
- множество излишних графических функций, усложняющих навигацию по графическому интерфейсу сервиса.

1.2.2 Баллов.Нет

Графический интерфейс платформы «Баллов.Нет» представлен на рисунках 1.11 и 1.12.

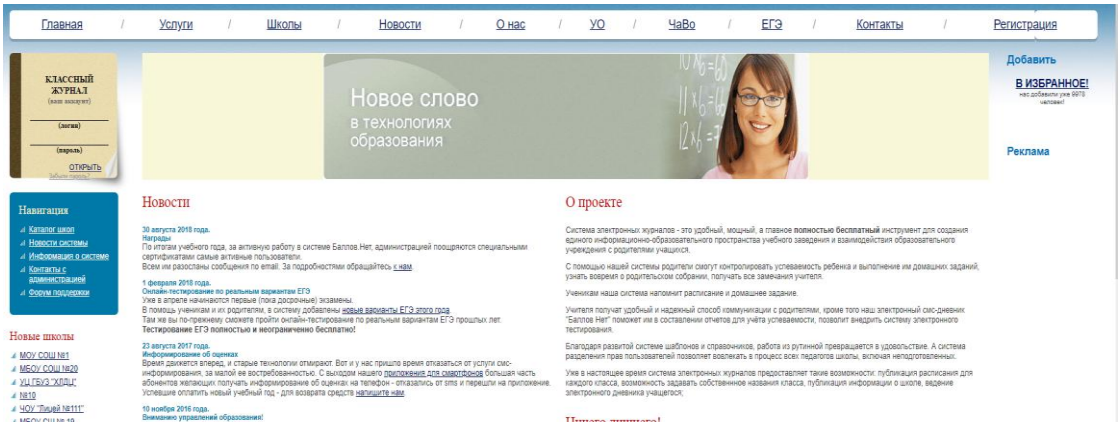


Рисунок 1.11 – Главная страница

Рисунок 1.12 – Форма регистрации

Платформа Баллов.Нет предоставляет функционал ЭЖ и ЭД в виде сетевого сервиса. Согласно описанию, уже в настоящее время система предоставляет такие возможности: публикация расписания для каждого класса, возможность задавать собственное название класса, публикация информации о школе, ведение электронного дневника учащегося [6].

Из преимуществ данного сервиса можно выделить следующие:

- наличие системы распределения прав пользователей;
- наличие роли администратора школы;
- минималистичный графический интерфейс;
- интеграция с сервисами пробного тестирования ЕГЭ, а также наличие информации, посвящённой экзамену;

- распространение на бесплатной основе.

Из недостатков данного сервиса можно выделить следующие:

- устаревший и неудобный графический интерфейс;
- невозможность исследовать функционирование сервиса без регистрации;
- наличие рекламы на сайте.

1.2.3 СтадиАппс

Согласно описанию, платформа «СтадиАппс» рассчитана для частных школ, так как она поддерживает далеко не только функционал "обычных" школ, но и то многое, о чём обычно не задумываются обычные общеобразовательные школы. «СтадиАппс» хорошо адаптирован для нестандартных ситуаций, которые часто встречаются как раз в частных школах. На платформе поддерживается большое количество систем оценивания [7].

Графический интерфейс платформы «СтадиАппс» представлен на рисунках 1.13 – 1.15.

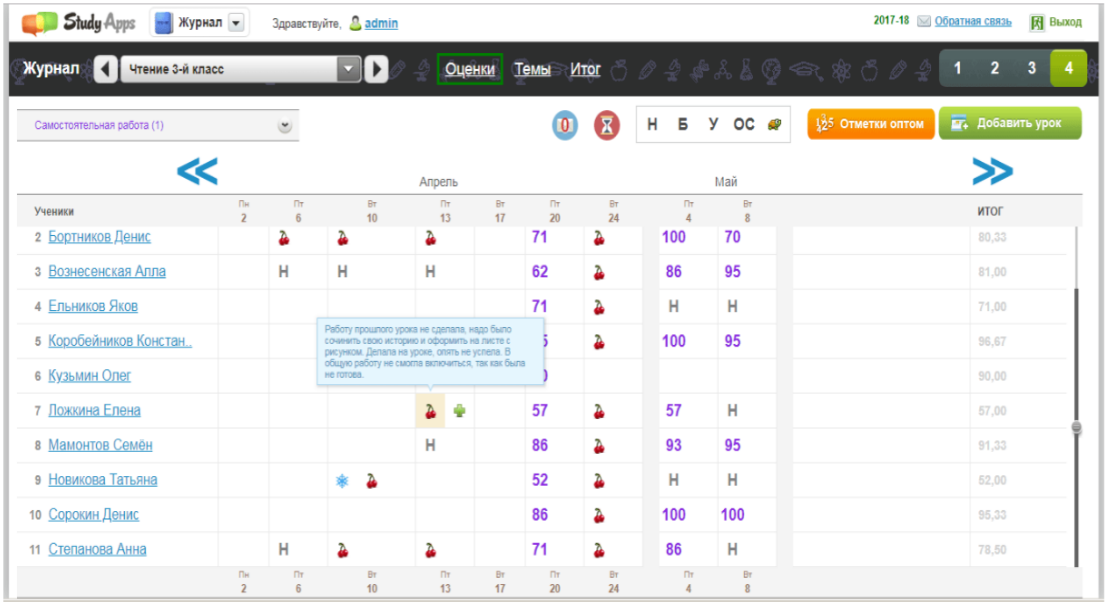


Рисунок 1.13 – Электронный журнал. Интерфейс учителя

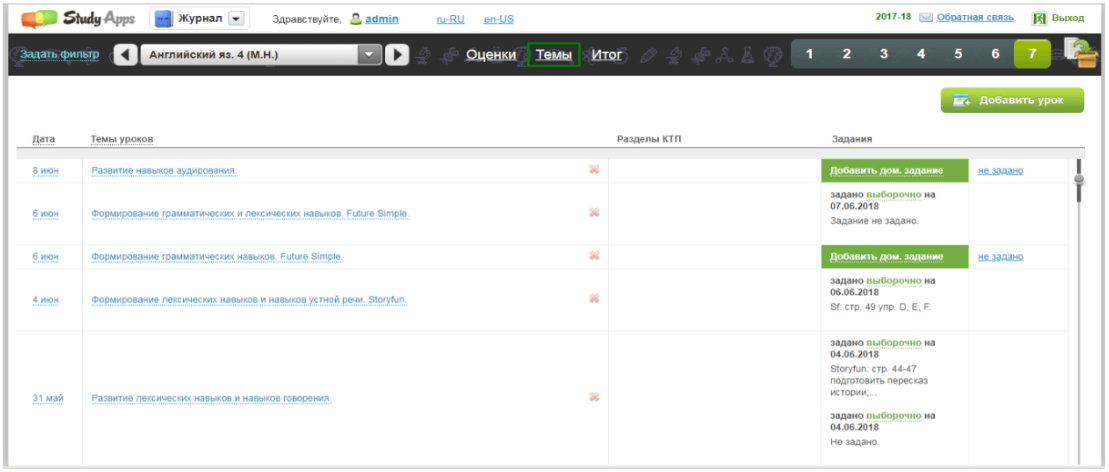


Рисунок 1.14 – Электронный журнал. Домашнее задание

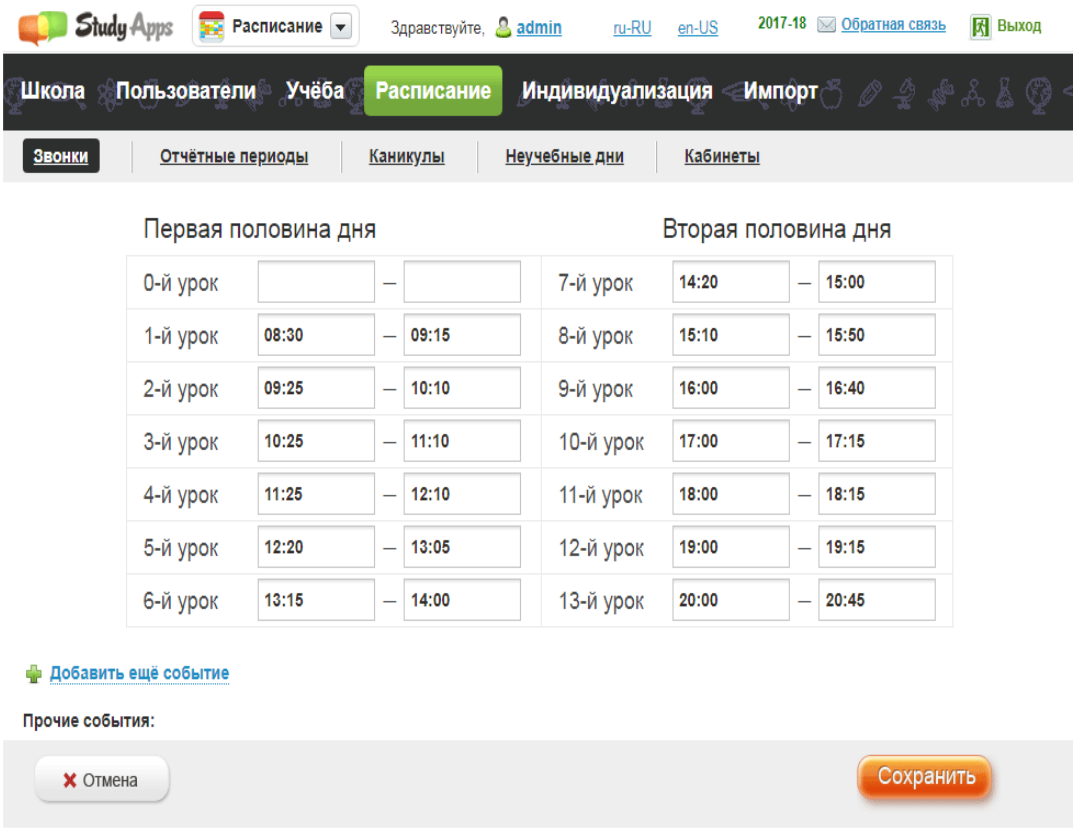


Рисунок 1.15 – Расписание звонков

Из преимуществ данного сервиса можно выделить следующие:

- большой функционал;
- распределение ролей;
- наличие возможности заказа создания пользовательского функционала;
- возможность создания отчётов, аналитики данных и ведение статистики.

Из недостатков данного сервиса можно выделить следующие:

- неподходящий для ведения учебного процесса интерфейс;
- большой объём выводимой информации;
- распространение на платной основе.

1.3 Выбор языка программирования

1.3.1 Язык программирования *C++*

C++ – компилируемый, статически типизированный язык программирования общего назначения. Он поддерживает различные парадигмы программирования, в числе которых процедурное, объектно-ориентированное, обобщённое и другие. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. *C++* сочетает свойства как высокоуровневых, так и низкоуровневых языков. Однако, к сожалению, создание графического интерфейса в языке сильно затруднено ввиду отсутствия такого функционала на уровне языка или стандартной библиотеки. Помимо этого, в языке также отсутствуют необходимые для обработки наборов данных средства.

1.3.2 Язык программирования *C#*

C# – объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании *Microsoft* под руководством Андерса Хейлсберга и Скотта Вильтаумота.

C# относится к семье языков с *C*-подобным синтаксисом, из них его синтаксис наиболее близок к *C++* и *Java*. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, *LINQ*, исключения [8].

Язык *C#* предоставляет удобный и эффективный инструментарий для создания приложений с графическим интерфейсом пользователя

в частности благодаря реализации шаблона «издатель — подписчик» на уровне языка.

Отдельно стоит упомянуть *LINQ – Language Integrated Query*. *LINQ* – это набор технологий, созданный для интеграции возможностей запросов напрямую в язык *C#*. На практике он представляет собой *SQL*-подобный синтаксис, который можно применять к двум типам источников данных: к объектам и непосредственно к *SQL*. В первом случае это означает, что запросы применяются к объектам тех классов, которые реализуют интерфейс *IEnumerable* или *IEnumerable<T>* [9]. Во втором же случае данный набор технологий применяется к внешнему источнику данных в виде БД, избавляя программиста от необходимости писать код, связанный с взаимодействием программы с внешней СУБД, позволяя интерпретировать таблицы в БД как коллекции из первого случая [10].

При всех своих преимуществах, язык *C#* имеет один существенный недостаток – он создан для разработки приложений для операционной системы *Microsoft Windows*, из-за чего его использование для разработки кроссплатформенных приложений сильно ограничено. Для запуска приложения на языке *C#* на компьютере должна быть установлена платформа *.NET*, которая существует в двух видах: *.NET Framework* – полноценная развитая платформа, работающая только под операционной системой *Microsoft Windows*, и *.NET* (ранее известная как *.NET Core*), которая может также работать под операционными системами семейства *Linux*, но всё ещё не обладающая всем функционалом *.NET Framework*. Помимо этого, использование *LINQ* ограничено использованием *Microsoft SQL Server* – проприетарной СУБД, распространяющейся на платной основе.

1.3.3 Язык программирования *Python*

Python – высокоуровневый язык программирования общего назначения с динамической типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём

программ. Благодаря обильному количеству библиотек, в том числе и для статистического анализа данных, простоте синтаксиса и кроссплатформенности, этот язык прекрасно подходит, но не ограничивается им, для написания программ по обработке т.н. «больших данных» (*big data*). Однако, как и в случае с *C++*, в *Python* требуется установить дополнительные библиотеки для проектирования графического интерфейса. Более того, этот язык известен своей медленной скоростью работы, что также ограничивает возможность применения этого языка для создания графического интерфейса. В дополнение, первые два фактора – обильное количество библиотек и простота синтаксиса – парадоксальным образом в совокупности лишь ухудшают читаемость кода: чтение кода программы и понимание его логики заменяются чтением документации библиотек.

1.3.4 Язык программирования *Java*

Java – статически типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией *Sun Microsystems*. Этот язык разрабатывался с расчётом на то, что программы будут исполняться на виртуальной машине, а не на реальной (как, например, программы, написанные на *C*/*C++*). Эта виртуальная машина получила название *JVM*. Подобный «отрыв» программы от конкретной аппаратной платформы позволил, во-первых, повысить безопасность исполняемой программы, а во-вторых, создать независимость написанной программы от конкретной программно-аппаратной платформы. Единственное требование – под эту платформу должна существовать *JVM* [11].

Начиная с версии *Java SE 8*, в языке присутствует инструмент *Java Stream API*. Это инструмент, позволяющий упростить и сократить написание кода для обработки наборов данных (коллекций). Фактически, он стал аналогом *LINQ* в языке *Java*, но ограниченным работой с коллекциями и файлами [12].

Swing – библиотека языка *Java* для разработки графического интерфейса пользователя. В этой библиотеке применяется видоизмененный вариант архитектуры *MVC*, где представление и контроллер объединены в один логический объект, называемый представитель пользовательского

интерфейса [13]. Это позволяет легко и быстро проектировать нужный по функционалу графический интерфейс в приложении. В данной работе это особенно важно для табличных элементов интерфейса, т.к. специфика прикладной области предполагает основную работу именно с данными в табличном виде.

Помимо всего этого, язык *Java* предоставляет *JDBC* – стандарт взаимодействия приложений с различными СУБД. Бремя реализации этого стандарта в виде библиотеки, которую могут использовать программисты, ложится на плечи разработчиков СУБД.

Учитывая все преимущества и недостатки описанных языков программирования, был выбран язык *Java* в версии *Java SE 8*.

1.4 Хранение данных

Для структурированного хранения данных существуют БД.

На текущий момент предоставлено множество видов БД и систем управления ими. Существует два основных класса БД: реляционные и нереляционные. Основное отличие заключается в представлении данных.

Реляционные БД называют также табличными, потому что данные в них хранятся в виде таблиц.

Данные в нереляционных БД могут хранится в разных видах, но в основном, это словари, где ключом является первичный ключ сущности, а значением – остальные поля таблицы. Подобный подход позволяет абстрагироваться от строгой типизации и строгого формата данных. Также нереляционные БД позволяют хранить иерархические структуры данных, в то время как для реализации подобной задачи с помощью реляционных БД потребовалось бы создавать дополнительные таблицы.

Ввиду специфики прикладной области, выбор будет производиться из числа реляционных СУБД, т.к. представление данных в виде таблиц является естественным для данной прикладной области, что позволит писать более простые запросы к БД, что увеличит скорость работы с ней.

1.5 Выбор СУБД

Исходя из принятых в подразделах 1.3 и 1.4 решений, выбор СУБД будет осуществляться из тех реляционных СУБД, для которых существует реализация *JDBC*.

1.5.1 СУБД *PostgreSQL*

PostgreSQL – свободная объектно-реляционная СУБД.

Сильными сторонами *PostgreSQL* считаются:

- высокопроизводительные и надёжные механизмы транзакций и репликации;
- наследование;
- расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных) [14].

Помимо этого, данная СУБД полностью удовлетворяет требованиям *ACID* и имеет полноценную реализацию *JDBC*.

1.5.2 СУБД *SQLite*

SQLite – компактная встраиваемая СУБД. Её основным преимуществом является компактность, что позволяет её использовать на ограниченных по аппаратным ресурсам машинах, в том числе встраиваемых системах. Эта компактность достигается за счёт некоторых упрощений (например, в этой СУБД нельзя создавать учётные записи пользователей) и того, что БД хранится на накопителе в виде одного файла, а сама подсистема хранения является частью самого приложения, а не отдельно работающей программой. Несмотря на это, *SQLite* удовлетворяет требованиям *ACID*, а скорость её работы высока. При этом под неё также существует реализация *JDBC*.

1.5.3 СУБД *MySQL*

MySQL – свободная реляционная СУБД. Она может распространяться как под *GNU General Public License*, так и под собственной коммерческой лицензией. *MySQL* является одной из немногих СУБД, в которой можно изменить

подсистему хранения, что повышает гибкость её использования. Одна из этих систем – *InnoDB* – удовлетворяет требованиям *ACID*. Помимо этого, преимуществами *MySQL* являются:

- быстродействие: *MySQL* является одной из самых быстродействующих СУБД;
- простота использования;
- наличие программных интерфейсов ко многим языкам программирования, в числе которых *Java*, *C++*, *PHP* и *Python*, что в совокупности позволяет создавать как обычные клиентские приложения с графическим интерфейсом, так и веб-приложения, работающие с одним и тем же экземпляром СУБД;
- безопасность: *MySQL* снабжена развитой системой защиты от несанкционированного доступа;
- малый размер;
- переносимость;
- двойное лицензирование [15].

1.6 Выводы

По результатам анализа существующих аналогов было принято решение разработать программный сервис ведения учебного процесса школы со следующими требованиями:

- иметь клиент-серверную архитектуру;
- реализовывать графический интерфейс для клиентской части;
- иметь разделение ролей на администратора, преподавателя (учителя) и обучающегося;
- обладать возможностью записи домашнего задания, учёта оценок и посещаемости преподавателями (учителями);
- возможность просмотра успеваемости, посещаемости и домашнего задания обучающимися или их родителями (законными представителями).

Разработка будет производиться с использованием языка *Java* в версии *Java SE* 8, так как *Java* позволяет создавать удобный и понятный для пользователя графический интерфейс, и СУБД *MySQL*, сконфигурированной на работу с подсистемой хранения *InnoDB*.

1.7 Выбор инструментов разработки

1.7.1 Выбор среды разработки

В качестве среды разработки было решено выбрать *NetBeans IDE* версии 8.2. Немаловажным преимуществом данной среды является наличие встроенной системы сборки и компиляции программ. Программисту не нужно компилировать файлы исходного кода по отдельности, а потом собирать их в один программный модуль, или же искать сторонние средства автоматизации сборки. Также, данная среда поддерживает интеграцию с системами управления версиями, что позволяет разработчику отслеживать изменения и фиксировать их, не переключая рабочее окно. В дополнение, в среде *NetBeans IDE* предусмотрен графический редактор форм, позволяющий существенно упростить разработку и проектирование графических интерфейсов. Всё вышеперечисленное делает *NetBeans IDE* крайне удобной средой для разработки приложений на *Java*.

1.7.2 Выбор платформы для реализации графического интерфейса

В качестве графической платформы приложения была выбрана библиотека *Swing*.

Swing – это библиотека для создания приложений на *Java*, которая позволяет разрабатывать многофункциональные пользовательские интерфейсы.

Помимо упомянутого в пункте 1.3.4 подраздела 1.3, библиотека обладает и иными преимуществами. Как указано в работе [13], почти все компоненты библиотеки *Swing* являются легковесными. Это означает, что они написаны исключительно на *Java* и не преобразуются в равноправные компоненты для конкретной платформы. Следовательно, легковесные компоненты являются более эффективными и гибкими. Более того, легковесные компоненты не преобразуются в равноправные платформенно-ориентированные компоненты,

поэтому внешний вид каждого компонента определяется библиотекой *Swing*, а не базовой операционной системой. Это означает, что каждый компонент будет действовать одинаково на всех платформах.

Все графические элементы определяются с помощью кода на языке *Java*, однако, в среде *NetBeans IDE* код описания графического интерфейса не требуется писать вручную: он генерируется автоматически из файла разметки с расширением *.form*, которые генерируются самой средой разработки при изменении элементов интерфейса разрабатываемого приложения во встроенном графическом редакторе.

Исходя из вышеописанных преимуществ и особенностей, библиотека *Swing* удовлетворяет требованиями для создания графического интерфейса разрабатываемой автоматизированной системы.

1.7.3 Выбор инструмента проектирования БД

В качестве инструмента для проектирования БД была выбрана программа *MySQL Workbench*.

MySQL Workbench – инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для СУБД *MySQL*.

Возможности данной программы:

- позволяет наглядно представить модель базы данных в графическом виде;
- наглядный и функциональный механизм установки связей между таблицами, в том числе «многие ко многим» с созданием таблицы связей;
- *reverse engineering* – восстановление структуры таблиц из уже существующей на сервере БД;
- удобный редактор *SQL* запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы;
- возможность редактирования данных в таблице в визуальном режиме.

1.7.3 Выбор инструмента для взаимодействия с СУБД *MySQL*

Для работы с СУБД используется библиотека *Connector/J*. *Connector/J* – это реализация *JDBC* для *MySQL*, которую можно подключить к программе на языке *Java* как простую библиотеку.

В языке *Java*, все компоненты *JDBC* находятся в пакете *java.sql*. Основными компонентами по работе с СУБД являются:

- *DriverManager* – основной класс для управления драйверами *JDBC*;
- *Connection* – интерфейс, определяющий взаимодействие с объектом класса, реализующего соединение с СУБД;
- *Statement* – интерфейс, определяющий взаимодействие с объектом класса, представляющего *SQL*-запрос;
- *PreparedStatement* – интерфейс, определяющий взаимодействие с объектом класса, представляющего подготовленный *SQL*-запрос;
- *CallableStatement* – интерфейс, определяющий взаимодействие с объектом класса, представляющего подготовленный *SQL*-запрос для вызова хранимых процедур;
- *ResultSet* – интерфейс, определяющий взаимодействие с объектом класса, представляющего результат исполнения *SQL*-запроса;
- *ResultSetMetaData* – интерфейс, определяющий взаимодействие с объектом класса, представляющего метаданные полученного объекта *ResultSet*;
- *SQLException* – класс, являющийся базовым для всех исключений, возникающих при работе с СУБД.

В общем случае, алгоритм работы с библиотекой выглядит следующим образом:

а) установить соединение с СУБД, вызвав метод *DriverManager.getConnection*, передав все необходимые параметры;

б) создать представление *SQL*-запроса, вызвав на полученном объекте *Connection* метод *createStatement*, *prepareStatement* или *prepareCall*;

в) для полученного соответствующего объекта вызвать необходимую вариацию метода *execute*;

г) если запрос предполагал получение результата, то обработать возвращённый объект *ResultSet* и, по необходимости, *ResultSetMetaData*;

д) закрыть объект, полученный на шаге б), вызвав на нём метод *close*;

е) при необходимости обработать возникшие исключения.

Также следует учитывать, что при закрытии объекта, полученного на шаге б), будут закрыты все порождённые этим объектом объекты *ResultSet* и *ResultSetMetaData*.

Отдельно стоит рассмотреть такую возможность, как «*prepared statement*». *Prepared statement* (подготовленный запрос, параметризированный запрос, запрос со связываемыми переменными) – это часть функциональности *SQL*-баз данных, предназначенная для отделения данных запроса и собственно выполняемого *SQL*-запроса [16].

При использовании связываемых переменных есть несколько преимуществ:

- один и тот же подготовленный запрос можно использовать несколько раз для разных данных, тем самым сокращая код;
- запросы со связываемыми переменными лучше кэшируются сервером, сокращая время синтаксического разбора;
- запросы со связываемыми переменными обладают готовой встроенной защитой от *SQL*-инъекций; это достигается за счёт того, что данные и запрос передаются на сервер отдельно друг от друга [16].

И *JDBC*, и *MySQL* поддерживают использование параметризированных запросов, что позволит повысить скорость работы и защищённость данных в разрабатываемом программном сервисе.

В *JDBC* параметризированный запрос представлен интерфейсами *PreparedStatement* и *CallableStatement* (т.к. второй расширяет функционал первого).

1.8 Архитектура программного сервиса

1.8.1 Архитектура

При разработке программного сервиса основной архитектурой

разрабатываемого комплекса программ будет «клиент-сервер». Термин «клиент-сервер» означает такую архитектуру программного комплекса, в которой его функциональные части взаимодействуют по схеме «запрос-ответ». Если рассмотреть две взаимодействующие части этого комплекса, то одна из них (клиент) выполняет активную функцию, т. е. инициирует запросы, а другая (сервер) пассивно на них отвечает [17]. Классическая схема клиент-серверного приложения с БД представлена на рисунке 1.16.

Рисунок 1.16 – клиент-серверное приложение с БД

Любая информационная система должна иметь минимум три основные функциональные части: модули хранения данных, их обработки и интерфейса с пользователем. Каждая из этих частей может быть реализована независимо от двух других. Например, не изменяя программ, используемых для хранения и обработки данных, можно изменить интерфейс с пользователем таким образом, что одни и те же данные будут отображаться в виде таблиц, графиков или гистограмм. Не меняя программ представления данных и их хранения, можно изменить программы обработки, например, изменив алгоритм полнотекстового поиска. И наконец, не меняя программ представления и обработки данных, можно изменить программное обеспечение для хранения данных, перейдя, например, на другую файловую систему [17].

Классическая архитектура клиент-сервер распределяет три основные части приложения на два физических модуля. Обычно, прикладной компонент располагается на сервере (например, сервере БД), компонент представления данных – на стороне клиента, а компонент управления ресурсом распределяется

между клиентской и серверной частями. В этом заключается основной недостаток классической двухуровневой архитектуры.

Используя двухзвенную архитектуру во время распределения алгоритма обработки данных, разработчик должен обладать полной информацией о том, какие последние изменения были внесены в систему, и какова значимость этих изменений, из-за чего возникает большая сложность во время разработки клиент-серверной системы, её установки и обслуживании. Такие мероприятия стоят значительных усилий и постоянной координации действий, совершаемых разными группами специалистов.

Чтобы предотвратить сопутствующую несогласованность работы различных элементов такой архитектуры специалисты создали две модификации для двухзвенной архитектуры «клиент-сервер»: одна модификация стала называться «толстый клиент», а другая «тонкий клиент».

В данной двухзвенной архитектуре разработчики пытались перенести процесс выполнения обработки данных на одну из двух физических частей – на сторону клиента («толстого клиента»), или на сторону сервера («тонкого клиента»).

Каждый из описанных подходов обладал своими недостатками. В первом подходе неоправданно перегружалась сеть, так как по ней передавались необработанные а, следовательно, избыточного размера данные. Также был усложнён процесс поддержки системы и процесс её изменения, так как, чтобы заменить алгоритм для вычисления или исправления ошибок нужно одновременно полностью заменить весь интерфейс в программах, чтобы не возникли ошибки или несогласованности в обработке данных. Если же весь процесс обработки информации выполнялся бы на сервере, то возникла бы проблема в процессе описания встроенных процедур и процесса их отладки. Систему, обрабатывающую информацию на сервере невозможно было ни как перенести на другую платформу или операционную систему, без серьезных сбоев в работе [17].

С учётом всего вышеописанного, для разработки сервиса была выбрана архитектура «толстого клиента», т.к. само клиентское приложение должно не только получать данные и выводить их пользователю, но и производить их обработку, что в случае «тонкого клиента» требовало бы отправки этих данных на сервер и, как следствие, приводило бы к усложнению сетевого взаимодействия клиентской и серверной программ.

1.8.2 Организация сервера

На сегодняшний день существует два основных подхода к созданию серверных приложений: многопоточный сервер и мультиплексирующий сервер.

Концепция многопоточного сервера является самой простой как для понимания, так и для реализации: сервер после принятия нового соединения создаёт отдельный поток, в который передаёт полученный сокет, и этот поток работает с единственным подключением, которое ему было передано на создании. Соответственно, с закрытием сокета уничтожается и этот поток. Однако с распространением и ростом сетевых технологий и компьютерных сетей такой подход стал неэффективным.

На смену многопоточному серверу пришла концепция мультиплексирующего сервера. Сервер называется мультиплексирующим, если он работает с принятыми клиентскими соединениями как с одним набором данных. Это позволяет программе не создавать новый поток для работы с каждым новым подключённым клиентом, а ожидать, пока один из всего множества сокетов вызовет прерывание, что будет означать, что данный сокет требует внимания серверной программы. Подобный подход усложнил разработку сетевой части ПО, но при этом повысил эффективность работы операций сетевого ввода-вывода.

Язык *Java* поддерживает оба подхода. Многопоточный подход поддерживается с самого начала существования языка – *JDK* версии 1.0 –за счёт присутствия в нём пакета *java.net* [18]. Начиная с версии 1.4, в языке появился *NIO* – набор прикладных интерфейсов для реализации высокопроизводительных

операций ввода-вывода. Создание *NIO* позволило также реализовывать в языке *Java* мультиплексирующие серверы.

Учитывая всё вышеописанное, было принято решение использовать мультиплексирующий подход при создании серверной программы.

1.9 Разработка программного сервиса

1.9.1 Разработка сетевого протокола

Сетевой протокол должен предоставлять клиентской и серверной частям сервиса весь необходимый для реализации поставленных требований функционал. В ходе работы был разработан протокол, работа которого отражена на диаграмме последовательностей, приведённой в приложении А.1.

Основным классом, представляющим сетевой протокол, стал класс *NetPacket*. Он содержит в себе два поля: *code* и *data*. Поле *code* является кодом операции, на основании которого клиент и сервер принимают решение о том, как они должны интерпретировать поле *data*. Поле *data* же представляет собой непосредственно передаваемые данные.

Все возможные коды операций:

- *ERROR* – полученный пакет представляет собой сообщение об ошибке;
- *FETCH_PASSWORD* – полученный пакет представляет собой запрос на выборку пароля и соли по переданному логину;
- *FETCH_PASSWORD_RESPONSE* – полученный пакет представляет собой успешный ответ на запрос *FETCH_PASSWORD*;
- *LOGIN_SUCCESS* – полученный пакет представляет собой ответ о том, что проверка учётных данных пройдена успешно и пользователя можно авторизовать;
- *LOGIN_RESPONSE* – полученный пакет представляет собой ответ о том, что сервер успешно авторизовал пользователя;
- *DISCONNECT_REQUEST* – полученный пакет представляет собой запрос на закрытие текущего соединения с СУБД (при этом соединение с серверной программой не закрывается);

- *SQL_QUERY* – полученный пакет представляет собой *SQL*-запрос для получения результата (*ResultSet*);

- *SQL_RESPONSE* – полученный пакет представляет собой успешный ответ на *SQL_QUERY*;

- *SQL_UPDATE_TABLE* – полученный пакет представляет собой *SQL*-запрос на обновление таблицы;

- *SQL_UPDATE_BATCH* – полученный пакет представляет собой *SQL*-запрос на формирование и исполнение указанного *SQL*-запроса в виде пакета запросов;

- *SQL_UPDATE_RESPONSE* – полученный пакет представляет собой успешный ответ на *SQL_UPDATE_TABLE* или *SQL_UPDATE_BATCH*.

1.9.2 Проектирование базы данных

Для разработки структуры БД использовалась *ER*-модель, соответствующая методологии *IDEF*1*X*. *IDEF1X* является методом для разработки реляционных БД и использует условный синтаксис, специально разработанный для построения концептуальной схемы структуры данных, независимой от конечной реализации базы данных и аппаратной платформы. Основными элементами модели *IDEF1X* являются сущности, атрибуты и отношения.

ER-модель (модель «сущность-связь») –модель данных, позволяющая описать данные предметной области с помощью сущностей и связей между ними. В это модели данных атрибуты являются частью сущностей, а связи представляют собой отношение нескольких сущностей.

Для определения экземпляра сущности в модели используются первичные ключи. Первичный ключ – это атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности. Также, в одной сущности могут оказаться несколько атрибутов или наборов атрибутов, претендующих на роль первичного ключа. Такие претенденты называются потенциальными ключами. Альтернативный ключ – это потенциальный ключ, не ставший первичным [19]. Одна сущность может иметь несколько альтернативных ключей, но первичный ключ может быть всегда только один.

Существует три типа связей: идентифицирующая связь "один ко многим", связь "многие ко многим" и неидентифицирующая связь "один ко многим". С типами связей связывают и различные типы сущностей.

Различают два типа сущностей: зависимые и независимые. Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой и зависимой сущностями. Экземпляр зависимой сущности определяется только через отношение к родительской сущности [19].

Неидентифицирующая связь служит для связывания независимых сущностей.

Связь "многие ко многим" может быть создана только на уровне логической модели.

Диаграмма «сущность-связь» – графическое представление (диаграмма) *ER*-модели.

Логическая и физическая модели, созданные в ходе данной работы, приведены в приложениях А.2 и А.3 соответственно.

Users – таблица, хранящая информацию о пользователях. Её столбцы:

- *user_ID* – идентификатор пользователя в БД, является главным ключом;
- *login* – логин пользователя, является альтернативным ключом;
- *pwd* – хэш пароля пользователя;
- *salt* – соль для хэширования пароля;
- *name* – имя пользователя;
- *sname* – фамилия пользователя;
- *pname* – отчество пользователя;
- *role* – роль пользователя (администратор, учитель, обучающийся).

Примечание – в разрабатываемом сервисе не предусмотрена роль для родителей обучающихся, т.к. функционально эта роль совпадает с ролью обучающегося.

Classes – таблица, хранящая информацию об учебных занятиях. Её столбцы:

- *class* – название класса;
- *subject* – название предмета;

- *teacher_ID* – идентификатор учителя; является внешним ключом на столбец *user_ID* таблицы *users*;

- *entry_ID* – идентификатор записи, является главным ключом.

Также в таблице имеется составной ключ, состоящий из столбцов *class*, *subject* и *teacher_ID*.

Student_info – таблица, хранящая информацию об обучающихся. Её столбцы:

- *student_ID* – идентификатор обучающегося, является внешним ключом на столбец *user_ID* таблицы *users* и главным ключом таблицы;
- *class* – название класса.

Homework – таблица, хранящая информацию о домашнем задании. Её столбцы:

- *class* – название класса;
- *subject* – название предмета;
- *hw_date* – срок (дата), к которому нужно выполнить домашнее задание;
- *homework* – собственно домашнее задание;
- *entry_ID* – идентификатор записи, является главным ключом.

Также в таблице имеется составной ключ, состоящий из столбцов *class*, *subject* и *hw_date*.

Marks – таблица, хранящая информацию об оценках. Её столбцы:

- *student_ID* – идентификатор обучающегося, является внешним ключом на столбец *user_ID* таблицы *users*;
- *subject* – название предмета;
- *m_date* – дата выставления оценки;
- *mark* – собственно оценка;
- *entry_ID* – идентификатор записи, является главным ключом.

1.9.3 Разработка графического интерфейса

Графический интерфейс должен предоставлять возможности для

взаимодействия с клиентской программой в соответствии с диаграммой прецедентов, приведённой в приложении А.4.

Графический интерфейс состоит из разных окон, называемых формами, и элементов управления. Каждый элемент управления или каждая форма исполняют конкретную задачу, позволяя пользователю взаимодействовать с клиентской программой.

LoginForm – форма, позволяющая пользователю авторизоваться в системе.

AdminForm – форма, представляющая главное окно пользователя-администратора.

TeacherForm – форма, представляющая главное окно пользователя-учителя.

StudentForm – форма, представляющая главное окно пользователя-обучающегося.

Все последующие окна являются диалоговыми, то есть, принадлежат одной из вышеперечисленных форм.

usersFrame – окно, представляющее форму работы с таблицей пользователей.

studentsFrame – окно, представляющее форму работы с таблицей обучающихся.

subjectsFrame – окно, представляющее форму работы с таблицей учебных занятий.

usersFrame – окно, представляющее форму работы с таблицей пользователей.

addUserDialog – диалоговое окно для добавления пользователя.

addSubjectDialog – диалоговое окно для добавления нового учебного занятия.

changePwdDialog формы *AdminForm* – диалоговое окно для изменения пароля существующего пользователя. Также, в форме *WorkForm* есть одноимённое диалоговое окно для изменения пароля текущего авторизованного пользователя.

hwFrame формы *TeacherForm* – окно, представляющее форму работы с таблицей домашнего задания для пользователя-учителя.

mFrame формы *TeacherForm* – окно, представляющее форму работы с таблицей оценок и учёта посещаемости для пользователя-учителя.

chooseClassSubjectDialog – диалоговое окно для выбора класса и предмета, для которых требуется вывести таблицу оценок и учёта посещаемости.

hwFrame формы *StudentForm* – окно, представляющее форму работы с таблицей домашнего задания для пользователя-обучающегося.

mFrame формы *StudentForm* – окно, представляющее форму работы с таблицей оценок и учёта посещаемости для пользователя-обучающегося.

Графический интерфейс разработанных окон и форм представлен на рисунках 1.17-1.22.

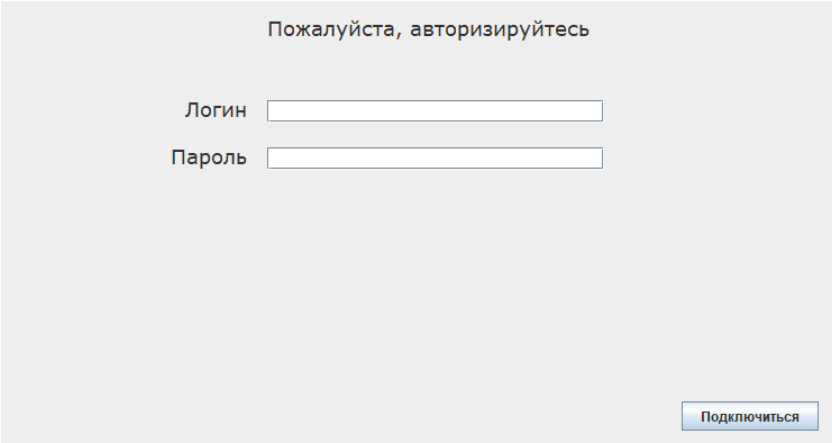


Рисунок 1.17 – форма *LoginForm*

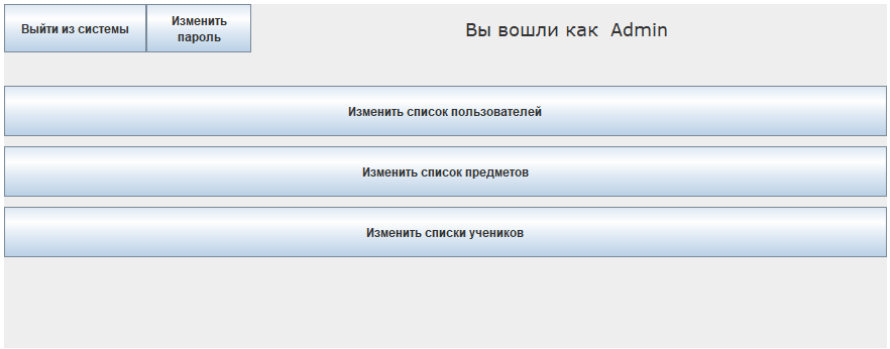


Рисунок 1.18 – форма *AdminForm*

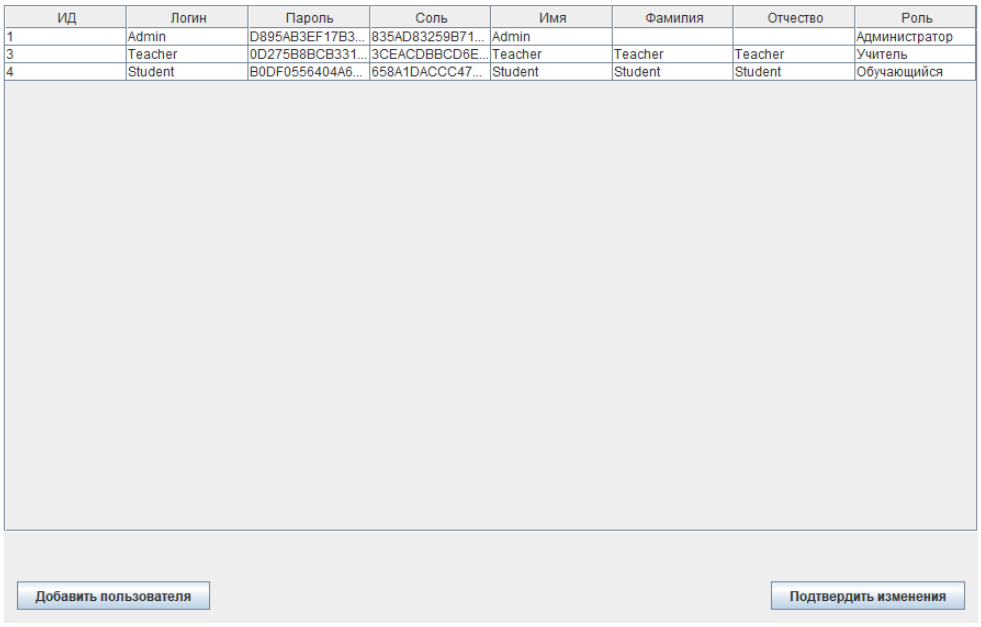


Рисунок 1.19 – окно *usersFrame*

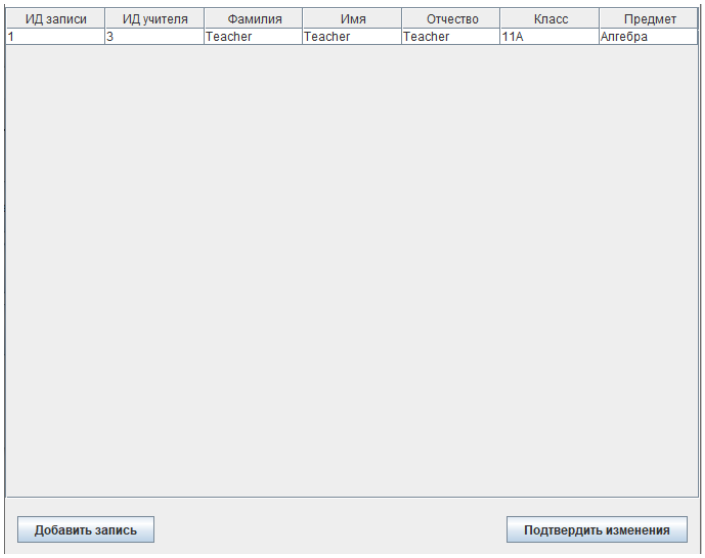


Рисунок 1.20 – окно *subjectsFrame*

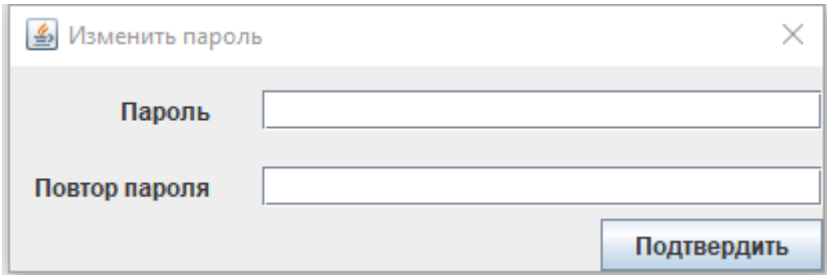


Рисунок 1.21 – диалоговое окно *changePwdDialog*

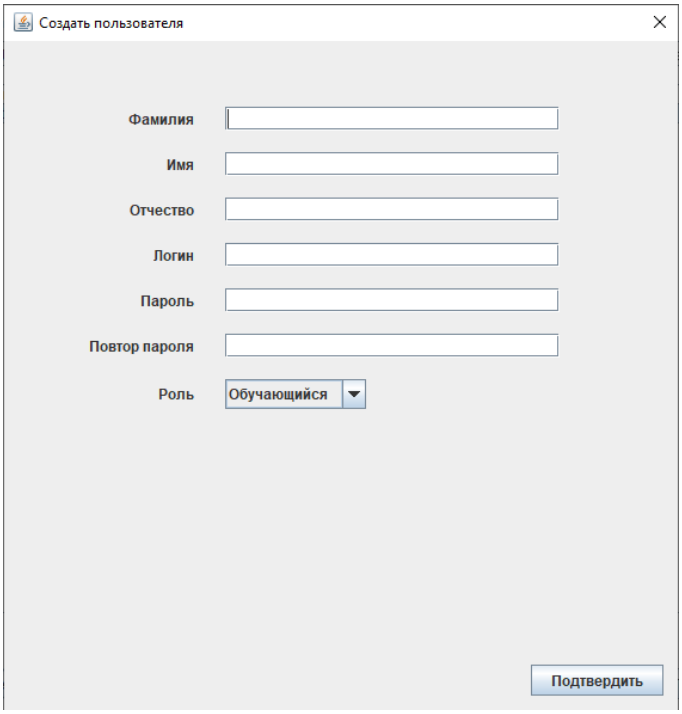


Рисунок 1.22 – диалоговое окно *addUserDialog*

Вид и дизайн остальных окон аналогичен приведённым на рисунках 1.17-1.22.

1.10 Описание программного сервиса

1.10.1 Руководство пользователя

1.10.1.1 Использование вспомогательной программы «*DbInit*»

Перед началом работы программного сервиса необходимо инициализировать БД. Для этого была создана вспомогательная программа «*DbInit*». Предполагается, что до её запуска уже предварительно установлен и настроен сервер СУБД *MySQL*. Программа запускается из консоли, на запуске ей должны быть переданы параметры командной строки. Первый параметр – название создаваемой базы данных, второй – порт сервера СУБД. После запуска программа выведет на экран приглашение на ввод пароля пользователя «*root*». Если все необходимые данные введены правильно, программа выведет на экран сообщение об успешном завершении. В случае возникновения ошибки на экран будет выведена трассировка стека. Пример успешного завершения данной программы приведён на рисунке 1.23.

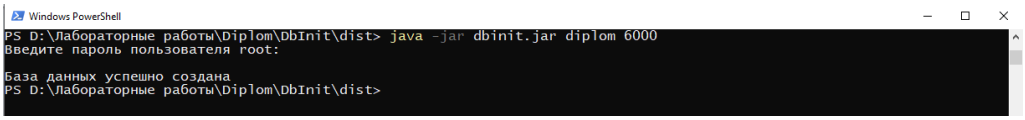


Рисунок 1.23 – успешная инициализация БД

Результатом работы программы является созданная БД, в которой зарегистрирована учётная запись администратора с логином и паролем «Admin». При желании, администратор затем может удалить эту учётную запись.

1.10.1.2 Запуск сервера

После инициализации БД, требуется запустить серверную программу. Для корректного запуска серверной программы необходимо создать файл конфигурации и хранилище сертификатов безопасности. Файл конфигурации должен называться «*sconfig.cfg*» и задавать следующие значения:

- *port* – порт, на котором будет работать сервер, если не задан, то по умолчанию программа запустится на порте 3333;
- *dbport* – порт сервера СУБД;
- *dbname* – название БД, с которой будет работать программа;
- *keystorepwd* – пароль хранилища сертификатов безопасности;
- *keypwd* – пароль ключа в хранилище.

Для создания хранилища сертификатов безопасности (*keystore*) требуется утилита «*keytool*», которая входит в состав *JDK*. С этой утилитой необходимо выполнить следующие действия в указанном порядке:

а) открыть от имени администратора командный интерпретатор в папке с установленной утилитой «*keytool*»;

б) выполнить команду «*keytool -genkey -keyalg RSA -alias* <название> *-keystore KeyStore.jks*», где <название> – название создаваемой записи;

в) следовать инструкциям на экране при создании ключей, при этом пароли хранилища и записи должны быть теми, что указаны в файле конфигурации;

г) выполнить команду «*keytool -export -alias* <название> *-keystore KeyStore.jks -file* <выходной_файл>», где <название> – название из шага б), а <выходной_файл> – имя выходного файла;

г) выполнить команду «*keytool -export -alias* <название> *-keystore KeyStore.jks -file* <выходной_файл>», где <название> – название из шага б), а <выходной_файл> – имя выходного файла;

д) следовать инструкциям на экране;

е) выполнить команду «*keytool -import -file* <файл> *-alias* <название> *-keystore truststore.jts*», где <название> – название из шага б), а <файл> – имя выходного файла из шага г);

ж) следовать инструкциям на экране.

Полученные файлы «*sconfig.cfg*» и «*KeyStore.jks*» необходимо сохранить в папке «*resources*», находящейся в той же папке, в которой хранится папка с исполняемым файлом серверной программы, т.е. если серверная программа расположена, например, по пути «*D:\Server\bin*», то указанные файлы должны быть в папке «*D:\Server\resources*».

Если все вышеперечисленные действия выполнены правильно, то можно запускать сервер командой «*java -jar Server.jar*» из командного интерпретатора. При этом необходимо, чтобы серверная программа работала на том же компьютере, что и сервер СУБД. Если ошибок не возникло, программа выведет на экран сообщение о начале работы, как показано на рисунке 1.24.

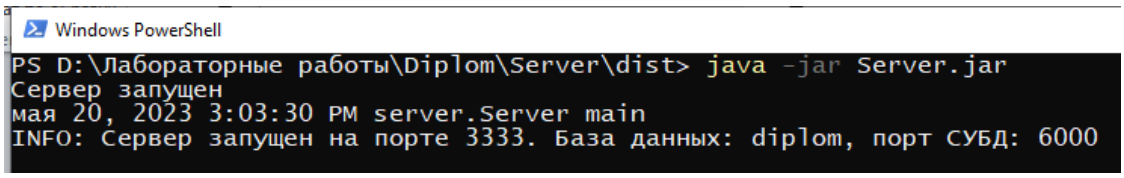


Рисунок 1.24 – успешный запуск серверной программы

1.10.1.3 Работа с клиентской программой

Для работы с клиентской программой необходимо иметь корректно настроенный файл конфигурации и файл хранилища доверенных сертификатов. Файл хранилища доверенных сертификатов – это файл «*truststore.jts*»,

полученный в подпункте 1.10.1.2. Файл конфигурации должен называться «*config.cfg*» и задавать следующие значения:

- *address* – адрес компьютера, на котором запущена серверная программа, может быть как IP-адресом, так и символьным именем удалённого компьютера;
- *serverport* – порт, на котором слушает соединения серверная программа, если не задан, то по умолчанию программа принимает это значение равным 3333;

- *truststorepwd* – пароль от хранилища доверенных сертификатов;
- *keypwd* – пароль ключа в хранилище.

Предполагается, что эти файлы будут доставлены пользователям вместе с самой программой. Требования к расположению этих файлов те же, что и для серверной программы.

Появление окна авторизации, представленного на рисунке 1.17, означает успешное подключение к серверу. В случае ошибки подключения, программа выведет диалоговое окно с предложением переподключиться к серверу, как показано на рисунке 1.25.

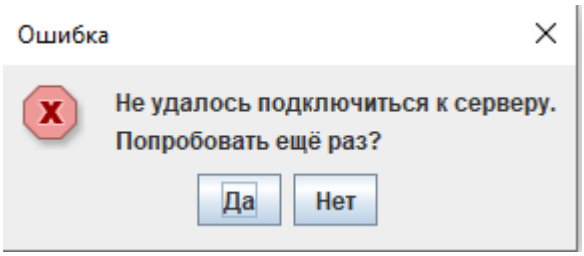


Рисунок 1.25 – диалоговое окно при ошибке подключения к серверу

В случае иной ошибки, программа завершит работу, а трассировка стека будет записана в файл журнала.

После появления формы авторизации, пользователю необходимо авторизоваться с помощью известных ему логина и пароля, а затем следовать инструкции для соответствующей роли.

1.10.1.4 Руководство пользователя

Главная форма администратора представлена на рисунке 1.18. На ней представлены командные кнопки роли, а также общие кнопки «Выйти из системы» и «Изменить пароль».

Нажатие кнопки «Выйти из системы» отправит на сервер команду, по которой тот закроет соединение с СУБД, связанное с текущим пользователем, т.е. фактически произойдёт отмена авторизации. Однако, нажатие данной кнопки не отключит пользователя от сервера окончательно, т.к. его вернёт на этап авторизации.

Кнопка «Изменить пароль» выведет пользователю диалоговое окно, показанное на рисунке 1.26. В этом диалоговом окне пользователю потребуется ввести его текущий пароль, который далее будет проверен, как при процедуре авторизации, и новый пароль два раза. Нажатие на кнопку «Подтвердить» запустит процедуру смены пароля. Если проверка текущего пароля пройдена успешно, то программа изменяет пароль для текущего авторизованного пользователя непосредственно в БД. В случае какой-либо ошибки пользователю будет выведено сообщение о соответствующей ошибке.

Общие кнопки функционируют одинаково для всех ролей.

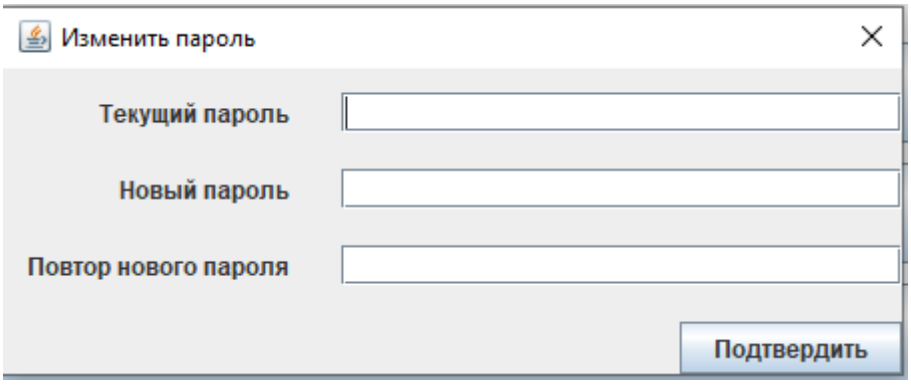


Рисунок 1.26 – Диалоговое окно *changePwdDialog*

Кнопка «Изменить список пользователей» позволяет администратору сначала просмотреть текущее содержание таблицы пользователей (*users*), а затем изменить его, при необходимости. Это происходит в окне, приведённом на рисунке 1.19. Добавление пользователя происходит в диалоговом окне,

представленном на рисунке 1.22. Это окно появляется по нажатию кнопки «Добавить пользователя» в окне *usersFrame* или контекстного меню, вызываемого щелчком правой кнопкой мыши по таблице. Также, в этом контекстном меню предусмотрены кнопки «Удалить пользователя» и «Изменить пароль». Первая кнопка удаляет запись о выбранном пользователе, вторая открывает диалоговое окно для смены пароля, как показано на рисунке 1.21. Это окно позволяет изменить пароль выбранному пользователю. Однако, все сделанные изменения локальны и не отражены в БД. Для фиксации изменений необходимо нажать кнопку «Подтвердить изменения», после чего программой будет сформирован запрос на обновление указанной таблицы и отправлен на сервер. В случае успешного применения изменений, пользователь увидит обновлённую таблицу с сервера БД, а в случае ошибки – получит соответствующее уведомление.

Для остальных ролей и окон последовательность действий аналогична описанной выше с учётом специфики области применения соответствующей таблицы.

Пользователи должны помнить, что не все поля таблиц они имеют возможность изменить. Обучающиеся же могут только просматривать данные, т.е. они вообще не могут вносить изменения в БД.

Поскольку роль родителя не предусмотрена архитектурой сервиса, эту роль может исполнять роль обучающегося, т.к. с точки зрения прикладной области, они функционально идентичны.

2 Охрана труда

2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения

Работа с вычислительной техникой, совершается в условиях ограниченной подвижности, с длительным статическим мышечным напряжением. Также происходит перенапряжение зрительных органов. Воздействие на человека выше указанных неблагоприятных факторов приводит к снижению работоспособности, переутомлению и психологическому раздражению, появлению физических недомогания, общей усталости.

Для снижения вредного воздействия вычислительной техники на человека, важно соблюдать ряд правил использования ПЭВМ. Основными, из которых являются:

- правильная организация рабочего места;
- освещенность рабочего места;
- соблюдение правил электробезопасности;
- соблюдение правил пожарной безопасности:
- режим труда и отдыха.

2.1.1 Оборудование рабочего места

Ниже представлены основные требования, которым должно удовлетворять рабочее место:

- расстояние от глаз до экрана должно быть в пределах 600 - 700 мм;
- монитор должен располагаться ниже уровня глаз. Угол между направлением при взгляде прямо и направлением на центр экрана должен составлять 10-15 градусов;
- рабочий стол должен регулироваться по высоте в пределах 680 - 800 мм, при отсутствии такой возможности его высота должна составлять 725мм;
- клавиатуру следует располагать на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной

регулируемой по высоте рабочей поверхности, отделенной от основной столешницы;

- рабочий стол должен иметь пространство для ног высотой не менее 600 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

На рисунке 2.1 изображена организация рабочего места, которая отвечает указанным требованиям.

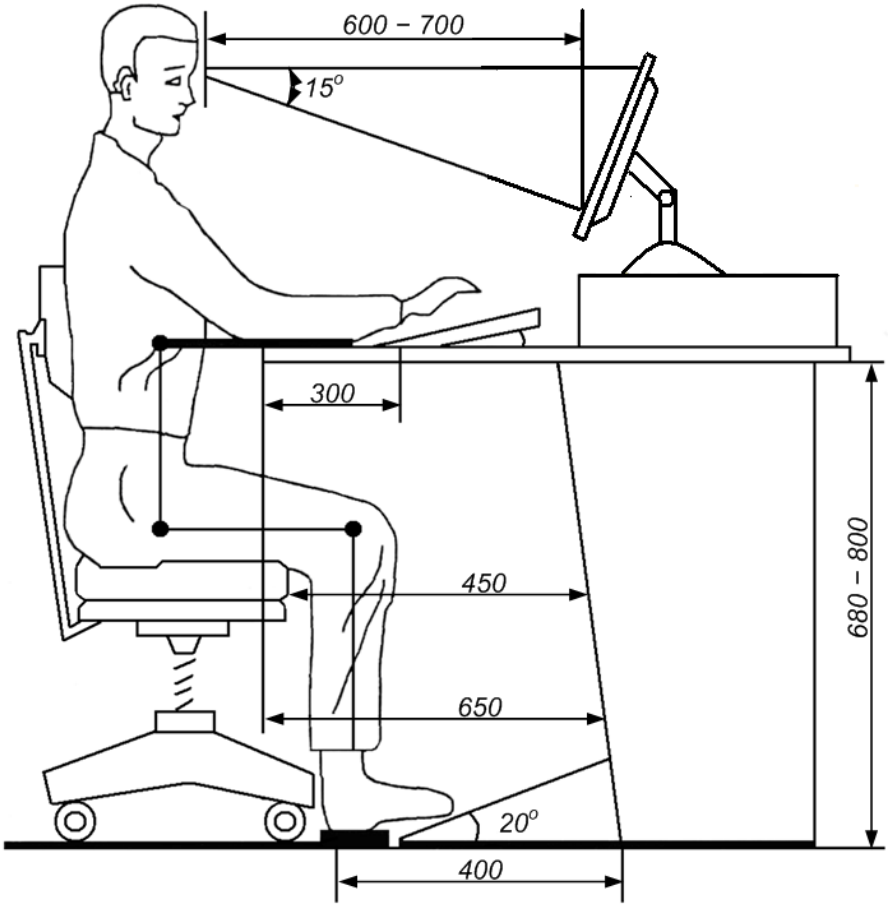


Рисунок 2.1 – Организация рабочего места

Рабочее место с дисплеем должно обеспечивать оператору возможность удобного выполнения работ в положении сидя и не создавать перегрузки костно-мышечной системы.

При правильно организованном рабочем месте должна обеспечиваться правильная посадка пользователя ПЭВМ, изображённая на рисунке 2.2

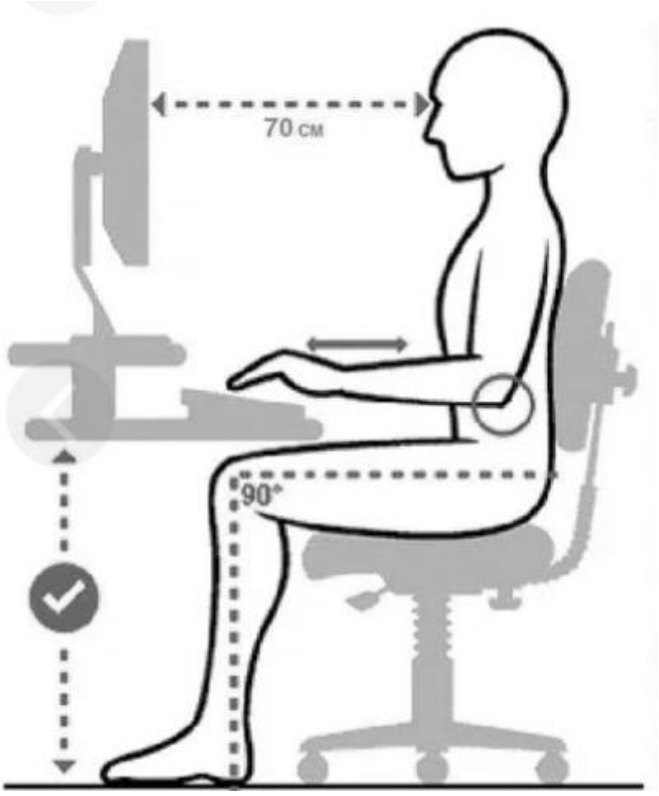


Рисунок 2.2 – Правильная посадка за ПК

Посадка должна обеспечивать выполнение следующих условий:

- запястья должны находиться в прямом положении;
- плечи находятся в расслабленном состоянии;
- локти должны полностью лежать на столе, не свисая и располагаться перпендикулярно полу (угол в локтевых суставах должен составлять 90-100 градусов).

2.1.2 Освещенность рабочего места

Нормы освещенности устанавливают с учетом характера выполняемой работы, места и поверхности. Для пультов ЭВМ и дисплеев – комбинированное освещение не менее 400 люкс, а общее освещение не менее 200 люкс.

На рабочем месте необходимо обеспечивать большую равномерность яркости, исключая наличие ярких и блестящих предметов, для снижения монотонности в поле зрения рекомендуется отдельные пестрые поверхности.

Для освещения используются в основном потолочные или встроенные светильники с люминесцентными лампами. Для исключения засвечивания

экранов прямыми световыми потоками светильники общего освещения располагают сбоку от рабочего места, параллельно линии зрения оператора.

2.1.3 Электробезопасность

К электробезопасности предъявляются требования по ГОСТ 12.1.019-2017 «ЭЛЕКТРОБЕЗОПАСНОСТЬ».

ПК – это комплекс устройств, работающий от сети переменного тока частотой 50 Гц с напряжением 230 В. Электрический ток при таком напряжении опасен для жизни. Поэтому при работе на ПК необходимо соблюдать следующие требования электробезопасности:

-- все устройства должны питаться от одной фазы электрической сети;

- сетевое электропитание устройств ПК должно производиться только от розеток типа «Европа» с заземляющими контактами;

- все электрические розетки, предназначенные для подключения к ним устройств ПК, должны иметь маркировку по напряжению;

- значение номинального напряжения сети необходимо наносить яркой краской, крупными символами (высотой не менее 50 мм) на стене или щите, возле розетки.

В процессе использования средств вычислительной техники и периферийного оборудования каждый работник должен внимательно относиться к электропроводке, приборам и аппаратам, всегда помнить, что несоблюдение правил безопасности угрожает здоровью и жизни человека.

Необходимо постоянно следить на своем рабочем месте за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть.

После обнаружения неисправности необходимо незамедлительно обесточить электрооборудование. Продолжение работы возможно только после устранения всех неисправностей.

2.1.4 Пожарная безопасность

При эксплуатации электроустановок и ПК наиболее частыми причинами пожаров, возникающих при использовании электроустановок, являются:

- короткие замыкания в проводках и электрических устройствах;
- воспламенение горючих материалов вблизи от электрических устройств;
- большие переходные сопротивления в местах контактных соединений.

Пожарные меры профилактики проводятся согласно федеральному закону от 21.12.1994 №69-ФЗ (ред. от 28.09.2010) "О пожарной безопасности". Для пожарной безопасности требуется выполнить ряд организационных мероприятий, направленных на обеспечение безопасности людей и предотвращение пожара с ограничением его распространения. А также создать условия для успешного тушения пожаров. Для повышения противопожарной безопасности помещения необходимы следующие мероприятия:

- проходы в помещения и запасные выходы не должны быть загромождены посторонними предметами, которые препятствуют свободному прохождению;
- к средствам пожаротушения всегда должен быть свободный доступ;
- система вентиляции должна быть оборудована устройствам, обеспечивающим автоматическое отключение ее при пожаре, а также огне задерживающим клапаном;
- установленная автоматическая пожарная сигнализация и средства пожаротушения должны содержаться в исправном состоянии и в соответствии с типовыми правилами технического содержания установок пожарной автоматики.

2.1.5 Режим труда и отдыха

Работник, работающий с ПК, обязан соблюдать режим труда и отдыха в зависимости от продолжительности, вида и категории трудовой деятельности:

Существуют следующие виды трудовой деятельности связанные с персональным компьютером:

- группа А – работа по считыванию информации с экрана видео дисплейного терминала с предварительным запросом;
- группа Б – работа по вводу информации;
- группа В – творческая работа в режиме диалога с ПЭВМ.

При выполнении работ, относящихся к различным видам трудовой деятельности, за основную работу с компьютером следует принимать тот вид деятельности, который занимает не менее 50% времени в течение всей рабочей смены или рабочего дня.

Для видов трудовой деятельности устанавливается 3 категории тяжести и напряженности работы с компьютером, которые определяются следующим образом:

- для группы А – по суммарному числу считываемых знаков за рабочую смену (не более 60000 знаков за смену);
- для группы Б – по суммарному числу считываемых или вводимых знаков за рабочую смену (не более 40000 знаков за смену);
- для группы В – по суммарному времени непосредственной работы с компьютером за рабочую смену (не более 6 часов за смену).

Суммарное время перерывов для каждой категории и видов работы указаны в таблице 2.1.

Таблица 2.1 – Время перерывов

Категория работы с ПЭВМ или ВДТ	Уровень нагрузки за рабочую смену при видах работы с ВДТ			Суммарное время регламентированных перерывов, мин	
	группа А, количество знаков	группа Б, количество знаков	группа В, час	при 8-часовой смене	при 12-часовой смене
1	До 20000	До 15000	До 2	50	80
2	До 40000	До 30000	До 4	70	110
3	До 60000	До 40000	До 6	90	140

Заключение

В ходе выполнения выпускной квалификационной работы был разработан программный сервис ведения учебного процесса школы на языке *Java*. Графический интерфейс был реализован с помощью библиотеки *Swing*.

Разработанная система позволяет вести учёт результатов учебного процесса, посещаемости обучающихся, хранить информацию о домашнем задании, а также имеет функционал разделения ролей и аутентификации пользователей. При этом вся информация прикладной области хранится на сервере БД.

Одним из достоинств разработанного приложения является наличие удобного и интуитивно понятного для пользователя графического интерфейса.

В дальнейшем можно усовершенствовать разработанный сервис и добавить возможность создавать статистические отчёты и проводить аналитику данных. Улучшить графический интерфейс, добавив возможность работать с несколькими таблицами сразу. Помимо этого, не лишней будет возможность получать доступ к сервису с помощью веб-интерфейса, то есть, интернет-браузера. Также можно добавить интеграцию с различными сторонними информационными сервисами, например, электронной почтой.

Список использованных источников

1. legalacts.ru: [Электронный ресурс]. URL: https://legalacts.ru/doc/pismo-minobrnauki-rossii-ot-15022012-n-ap-14707/ (дата обращения: 26.04.2023)
2. dnevnik.ru: [Электронный ресурс]. URL: https://dnevnik.ru/features#/teachers (дата обращения: 26.04.2023)
3. dnevnik.ru: [Электронный ресурс]. URL: https://dnevnik.ru/features#/parents (дата обращения: 26.04.2023)
4. dnevnik.ru: [Электронный ресурс]. URL: https://dnevnik.ru/features#/students (дата обращения: 26.04.2023)
5. dnevnik.ru: [Электронный ресурс]. URL: https://dnevnik.ru/features#/governments (дата обращения: 26.04.2023)
6. ballov.net: [Электронный ресурс]. URL: https://ballov.net (дата обращения: 26.04.2023)
7. studyapps.ru: [Электронный ресурс]. URL: https://studyapps.ru (дата обращения: 26.04.2023)
8. wikipedia.org: [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/C_Sharp (дата обращения: 27.04.2023)
9. *LINQ to Objects* (*C#*) | *Microsoft Learn*: [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/linq-to-objects (дата обращения: 27.04.2023)
10. *Getting Started* | *Microsoft Learn*: [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/sql/linq/getting-started (дата обращения: 27.04.2023)
11. metanit.com: [Электронный ресурс]. URL: https://metanit.com/java/tutorial/1.1.php (дата обращения: 27.04.2023)
12. metanit.com: [Электронный ресурс]. URL: https://metanit.com/java/tutorial/10.1.php (дата обращения: 27.04.2023)
13. Шилдт, Г. Java 8. Полное руководство; 9-е изд.: Пер. с англ. - М. : ООО "И.Д. Вильямс", 2015. — 1376 с.

14. wikipedia.org: [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/PostgreSQL (дата обращения: 27.04.2023)

15. Дюбуа, П. MySQL; 2-е изд.: Пер. с англ. - М. : "Издательский дом "Вильямс", 2004. — 1056 с.

16. habr.com: [Электронный ресурс]. URL: https://habr.com/ru/articles/148446/ (дата обращения: 28.04.2023)

17. osp.ru: [Электронный ресурс]. URL: https://www.osp.ru/nets/1997/06/142618 (дата обращения: 28.04.2023)

18. oracle.com: [Электронный ресурс]. URL: https://docs.oracle.com/javase/8/docs/api/java/net/package-summary.html (дата обращения: 28.04.2023)

19. intuit.ru: [Электронный ресурс]. URL: https://intuit.ru/studies/courses/599/455/lecture/10163?page=2 (дата обращения: 28.04.2023)

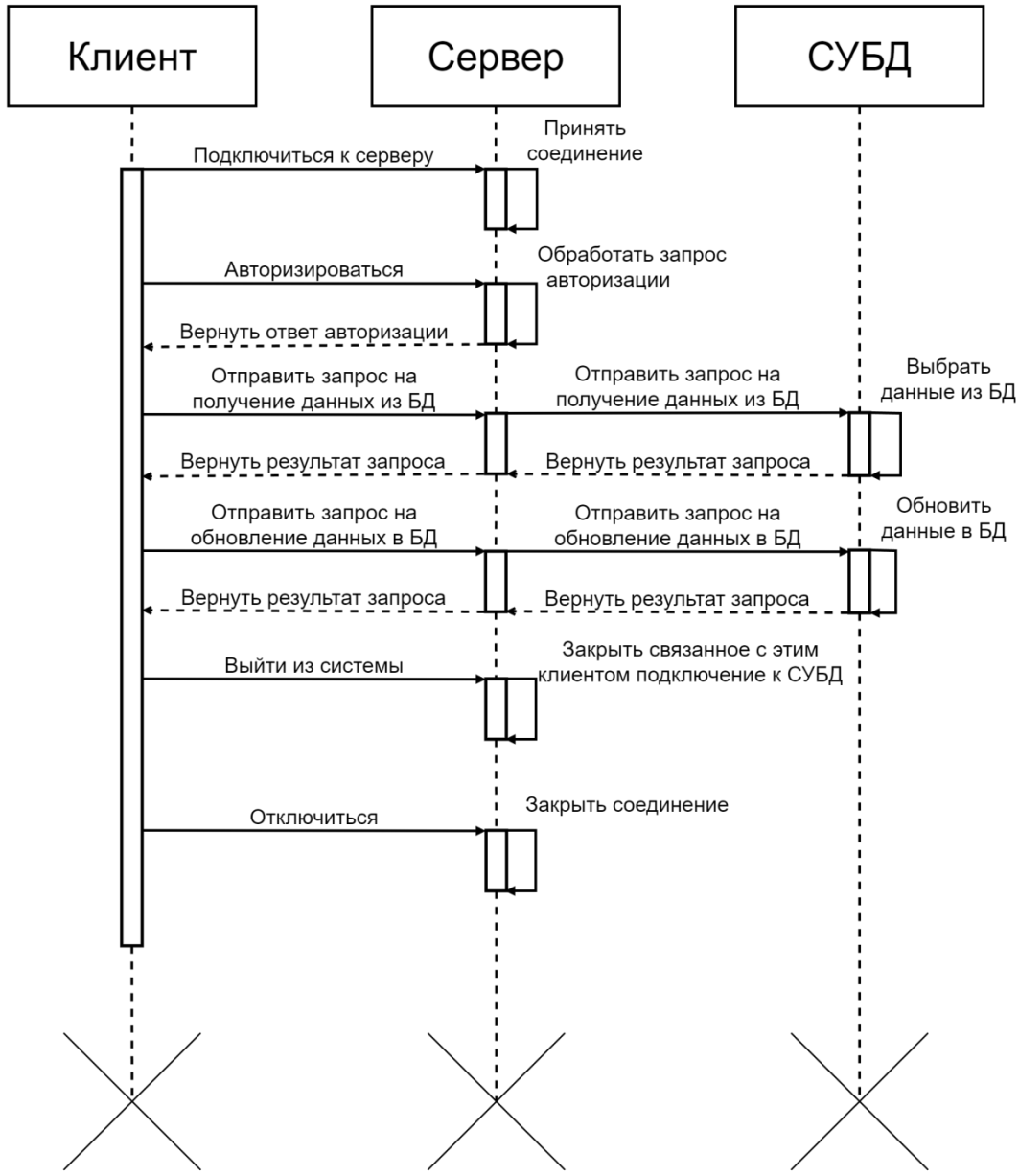
Приложение А

(обязательное)

Программный сервис ведения учебного процесса

Диаграммы

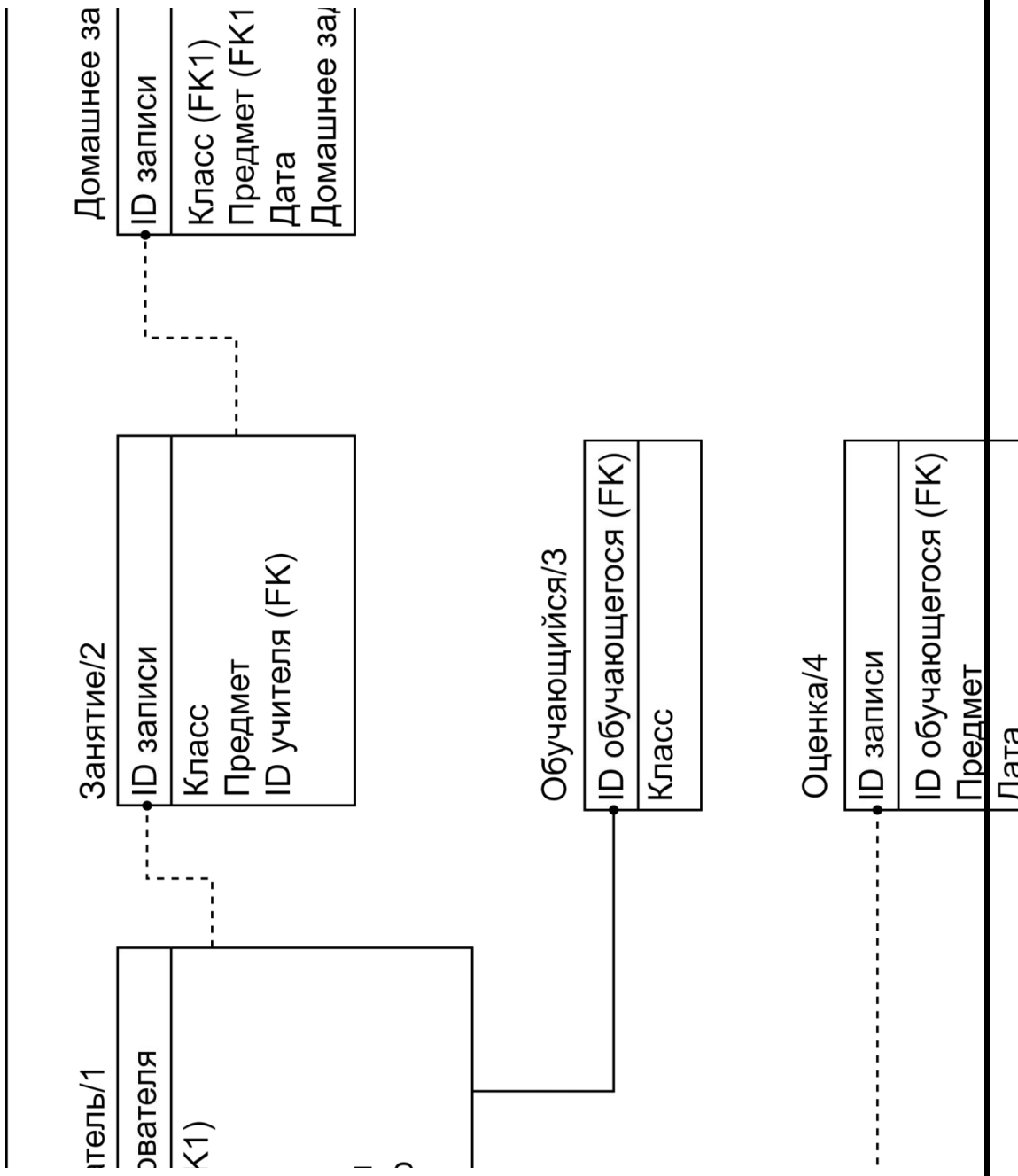
Приложение А.1 – Диаграмма последовательностей



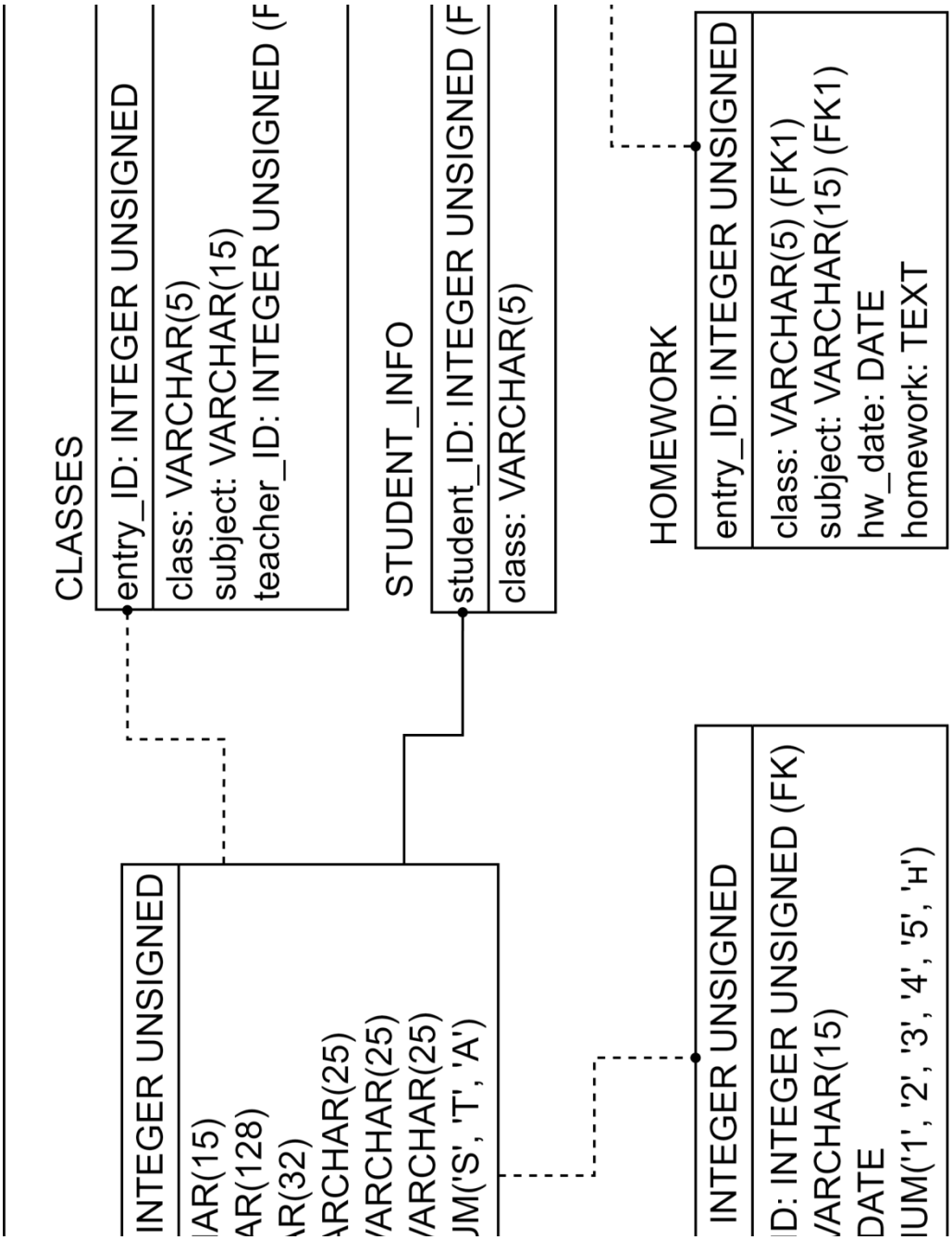
ПГУ1.090301.11.002 Д14

Программный сервис ведения учебного процесса
Диаграмма последовательностей

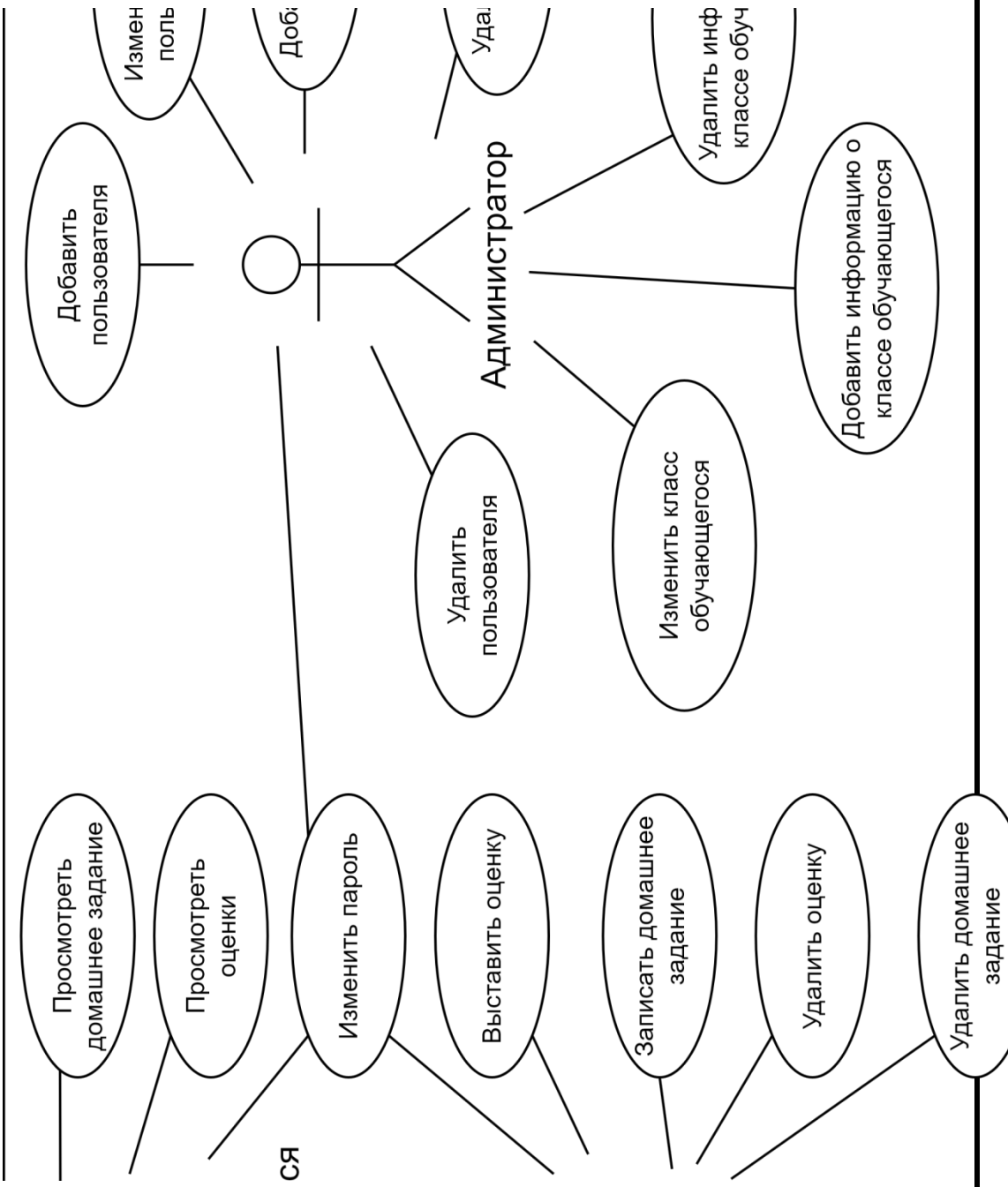
Приложение А.2 – Диаграмма «сущность–связь» логическая



Приложение А.3 – Диаграмма «сущность–связь» физическая



Приложение А.4 – Диаграмма прецедентов



Приложение А.5 – Диаграмма классов

Приложение Б

(обязательное)

Программный сервис ведения учебного процесса

Листинг программы

Файл «*Client.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;
import java.io.*;
import javax.swing.*;
import java.net.*;
import java.util.*;
import java.util.logging.*;
import my.*;
/**
 *
 * @author Admin
 */
public class Client {
    public static final Logger logger =
Logger.getLogger(Logger.GLOBAL_LOGGER_NAME);
    public static NioSslClient niocl;
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        AbstractForm wnd;
        String addr = "", truststorepwd = "", keypwd = "";
        int port = 3333;
        logger.setLevel(Level.ALL);
        try
        {
            Handler fileHandler = new FileHandler("clientlog", 200 *
1024 * 1024, 5);
            fileHandler.setFormatter(new MyFormatter());
            logger.addHandler(fileHandler);
        }
        catch(IOException ex)
        {
            ex.printStackTrace(System.out);
            return;
        }
        try(BufferedReader br = new BufferedReader(new
FileReader("../resources/config.cfg"))) {
            for(String line; (line = br.readLine()) != null; ) {
                String[] kv = line.split("\\s*=\\s*");
                switch(kv[0])
                {
                    case "address":
                        addr = kv[1];
                        break;
                    case "serverport":
```

```
                        port = Integer.parseInt(kv[1]);
                        break;
                    case "truststorepwd":
                        truststorepwd = kv[1];
                        break;
                    case "keypwd":
                        keypwd = kv[1];
                        break;
                }
            }
        }
        catch(IOException ex)
        {
            logger.log(Level.SEVERE, "Ошибка при чтении файла: ",
ex);
            return;
        }
        while(true)
        {
            try
            {
                niocl = new NioSslClient("TLSv1.2", addr, port,
truststorepwd, keypwd);
                niocl.connect();
                break;
            }
            catch(ConnectException ex)
            {
                logger.log(Level.WARNING, "Ошибка при подключении к
серверу: ", ex);
                if(niocl != null) niocl.shutdown();
                int ret = JOptionPane.showOptionDialog(null, "Не
удалось подключиться к серверу.\nПопробовать ещё раз?", "Ошибка",
                        JOptionPane.YES_NO_OPTION,
JOptionPane.ERROR_MESSAGE, null,
                        new String[] {"Да", "Нет"}, null);
                if(ret != 0) // 0 == Да
                    return;
            }
            catch(Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при создании
подключения: ", ex);
                return;
            }
        }
        wnd = new LoginForm();
        wnd.pack();
        wnd.setVisible(true);
    }

    public static void nullPwds(char[] pwd1, char[] pwd2)
    {
```

```
        if(pwd1 != null) Arrays.fill(pwd1, (char)0);
        if(pwd2 != null) Arrays.fill(pwd2, (char)0);
    }
}
```

Файл «*NioSslClient.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package client;

import static client.Client.logger;
import my.*;
import javax.net.ssl.*;
import java.nio.channels.*;
import java.nio.*;
import java.io.*;
import java.security.*;
import java.net.*;
import java.util.logging.Level;

/**
 *
 * @author Admin
 */
public class NioSslClient extends NioSslPeer {
    private final String remoteAddress;
    public final int port;
    private final SSLEngine engine;
    private SocketChannel socketChannel;

    public NioSslClient(String protocol, String remoteAddress, int
port, String truststorepwd, String keypwd) throws Exception  {
        this.remoteAddress = remoteAddress;
        this.port = port;
        SSLContext context = SSLContext.getInstance(protocol);

context.init(super.createKeyManagers("../resources/truststore.jts",
truststorepwd, keypwd),

super.createTrustManagers("../resources/truststore.jts",
truststorepwd), new SecureRandom());
        engine = context.createSSLEngine(remoteAddress, port);
        engine.setUseClientMode(true);

        SSLSession session = engine.getSession();
        myAppData = ByteBuffer.allocate(1024);
        myNetData =
ByteBuffer.allocate(session.getPacketBufferSize());
```

```
        peerAppData = ByteBuffer.allocate(1024);
        peerNetData =
ByteBuffer.allocate(session.getPacketBufferSize());
    }

    public boolean connect() throws Exception {
        socketChannel = SocketChannel.open();
        socketChannel.configureBlocking(true);
        socketChannel.connect(new InetSocketAddress(remoteAddress,
port));
        while (!socketChannel.finishConnect()) {
             // can do something here...
        }
        engine.beginHandshake();
        System.out.println("Connected to server");
        return doHandshake(socketChannel, engine);
    }

    public void write(byte[] message) throws IOException {
        write(socketChannel, engine, message);
    }

    @Override
    protected void write(SocketChannel socketChannel, SSLEngine
engine, byte[] message) throws IOException {
        myAppData.clear();
        myAppData.put(message);
        myAppData.flip();
        while (myAppData.hasRemaining()) {
            // The loop has a meaning for (outgoing) messages larger
than 16KB.
            // Every wrap call will remove 16KB from the original
message and send it to the remote peer.
            myNetData.clear();
            SSLEngineResult result = engine.wrap(myAppData,
myNetData);
            switch (result.getStatus()) {
            case OK:
                myNetData.flip();
                while (myNetData.hasRemaining()) {
                    socketChannel.write(myNetData);
                }
                break;
            case BUFFER_OVERFLOW:
                myNetData = enlargePacketBuffer(engine, myNetData);
                break;
            case BUFFER_UNDERFLOW:
                throw new SSLException("Buffer underflow occured
after a wrap. I don't think we should ever get here.");
            case CLOSED:
                closeConnection(socketChannel, engine);
                return;
            default:
```

```
                throw new IllegalStateException("Invalid SSL status:
" + result.getStatus());
            }
        }
    }

    public void read() throws Exception {
        read(socketChannel, engine);
    }

    protected void read(SocketChannel socketChannel, SSLEngine
engine) throws Exception  {
        peerNetData.clear();
        boolean exitReadLoop = false;
        while (!exitReadLoop) {
            int bytesRead = socketChannel.read(peerNetData);
            if (bytesRead > 0) {
                peerNetData.flip();
                while (peerNetData.hasRemaining()) {
                    peerAppData.clear();
                    SSLEngineResult result =
engine.unwrap(peerNetData, peerAppData);
                    switch (result.getStatus()) {
                    case OK:
                        peerAppData.flip();
                        exitReadLoop = true;
                        break;
                    case BUFFER_OVERFLOW:
                        peerAppData =
enlargeApplicationBuffer(engine, peerAppData);
                        break;
                    case BUFFER_UNDERFLOW:
                        peerNetData = handleBufferUnderflow(engine,
peerNetData);
                        break;
                    case CLOSED:
                        closeConnection(socketChannel, engine);
                        return;
                    default:
                        throw new IllegalStateException("Invalid SSL
status: " + result.getStatus());
                    }
                }
            } else if (bytesRead < 0) {
                handleEndOfStream(socketChannel, engine);
                return;
            }
        }
    }

    /**
     * Should be called when the client wants to explicitly close
the connection to the server.
```

```
     *
     */
    public void shutdown() {
        try
        {
            closeConnection(socketChannel, engine);
            executor.shutdown();
        }
        catch(IOException ex)
        {
            logger.log(Level.SEVERE, "Ошибка при закрытии
соединения: ", ex);
        }
    }

    public ByteBuffer peerAppData()
    {
        return peerAppData;
    }
}
```

Файл «*UserData.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;

/**
 *
 * @author Admin
 */
public class UserData {
    public final int id;
    public String pwd;
    public String salt;
    public UserData(int id, String pwd, String salt)
    {
        this.id = id;
        this.pwd = pwd;
        this.salt = salt;
    }
}
```

Файл «*AbstractForm.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package client;
import static client.Client.niocl;
import java.util.concurrent.*;
import my.*;

/**
 *
 * @author Admin
 */
public abstract class AbstractForm extends javax.swing.JFrame {
    protected NetPacket pkt;
    protected final ExecutorService executor;
    /**
     * Creates new form AbstractForm
     */
    protected AbstractForm() {
        initComponents();
        executor = Executors.newSingleThreadExecutor();
    }

    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Электронный дневник");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                formWindowClosing(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGap(0, 400, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
```

```
)
            .addGap(0, 300, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void formWindowClosing(java.awt.event.WindowEvent evt) {
        niocl.shutdown();
        executor.shutdown();
    }


    // Variables declaration - do not modify
    // End of variables declaration
}
```

Файл «*EditableForm.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;

import java.util.*;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import my.*;

/**
 *
 * @author Admin
 */
public abstract class EditableForm extends WorkForm {

    protected Map<Object, Map<String, Object>> changes;
    protected Set<Object> deletions;
    protected Set<TableRow> insertions;

    /**
     * Creates new form EditableForm
     */
    protected EditableForm() {
        initComponents();
        changes = new TreeMap<>();
        deletions = new HashSet<>();
        insertions = new HashSet<>();
    }

    @Override
    protected void reset()
```

```
    {
        super.reset();
        changes.clear();
        deletions.clear();
        insertions.clear();
    }
    @Override
    protected void WindowClosingAction(JTable t)
    {
        reset();
        t.setModel(new DefaultTableModel());
    }

    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGap(0, 400, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGap(0, 300, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    // Variables declaration - do not modify
    // End of variables declaration
}
```

Файл « *WorkForm.java*»

```
/*
 * To change this license header, choose License Headers in Project
```

```
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;

import static client.Client.logger;
import static client.Client.niocl;
import java.awt.HeadlessException;
import java.io.IOException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.sql.SQLException;
import java.util.*;
import my.*;
import java.util.logging.Level;
import javax.swing.BorderFactory;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author Admin
 */
public abstract class WorkForm extends AbstractForm {

    protected UserData udata;
    protected Map<String, String> colnames;
    /**
     * Creates new form WorkForm
     */
    protected WorkForm() {
        initComponents();
        colnames = new TreeMap<>();
    }

    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        changePwdDialog = new javax.swing.JDialog();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        changePwdConfirmButton = new javax.swing.JButton();
```

```
        newPwdField1 = new javax.swing.JPasswordField();
        newPwdField2 = new javax.swing.JPasswordField();
        curPwdField = new javax.swing.JPasswordField();
        jLabel14 = new javax.swing.JLabel();
        operationDialog = new javax.swing.JDialog();
        connectProgressBar = new javax.swing.JProgressBar();
        connectLabel = new javax.swing.JLabel();
        jPanel1 = new javax.swing.JPanel();
        changePwdButton = new javax.swing.JButton();
        disconButton = new javax.swing.JButton();
        nameLabel = new javax.swing.JLabel();

        changePwdDialog.setTitle("Изменить пароль");
        changePwdDialog.setModal(true);
        changePwdDialog.addWindowListener(new
java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                changePwdDialogWindowClosing(evt);
            }
        });

        jLabel12.setText("Новый пароль");

        jLabel13.setText("Повтор нового пароля");

        changePwdConfirmButton.setText("Подтвердить");
        changePwdConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                changePwdConfirmButtonMouseClicked(evt);
            }
        });

        jLabel14.setText("Текущий пароль");

        javax.swing.GroupLayout changePwdDialogLayout = new
javax.swing.GroupLayout(changePwdDialog.getContentPane());

changePwdDialog.getContentPane().setLayout(changePwdDialogLayout);
        changePwdDialogLayout.setHorizontalGroup(

changePwdDialogLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(changePwdDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
changePwdDialogLayout.createSequentialGroup()
```

```
.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.TRAILING)
                            .addComponent(jLabel12)
                            .addComponent(jLabel13))
                        .addGap(26, 26, 26)

.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.TRAILING)
                            .addComponent(newPwdField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(newPwdField2,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
changePwdDialogLayout.createSequentialGroup()
                        .addComponent(jLabel14)
                        .addGap(26, 26, 26)
                        .addComponent(curPwdField,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(changePwdConfirmButton,
javax.swing.GroupLayout.Alignment.TRAILING)))
        );
        changePwdDialogLayout.setVerticalGroup(

changePwdDialogLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(changePwdDialogLayout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jLabel14)
                    .addComponent(curPwdField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jLabel12)
                    .addComponent(newPwdField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
```

```
.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jLabel13)
                    .addComponent(newPwdField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addComponent(changePwdConfirmButton))
        );


operationDialog.setModalityType(java.awt.Dialog.ModalityType.APPLICA
TION_MODAL);
        operationDialog.setResizable(false);
        operationDialog.setUndecorated(true);

operationDialog.getRootPane().setBorder(BorderFactory.createRaisedBe
velBorder());

        connectProgressBar.setIndeterminate(true);

        connectLabel.setFont(new java.awt.Font("Verdana", 0, 18));
// NOI18N
        connectLabel.setText("Операция выполняется...");

        javax.swing.GroupLayout operationDialogLayout = new
javax.swing.GroupLayout(operationDialog.getContentPane());

operationDialog.getContentPane().setLayout(operationDialogLayout);
        operationDialogLayout.setHorizontalGroup(

operationDialogLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(operationDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(operationDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING, false)
                    .addComponent(connectLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(connectProgressBar,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        operationDialogLayout.setVerticalGroup(

operationDialogLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
```

```
            .addGroup(operationDialogLayout.createSequentialGroup()
                .addGap(39, 39, 39)
                .addComponent(connectLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATE
D)
                .addComponent(connectProgressBar,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(35, Short.MAX_VALUE))
        );


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setLayout(new java.awt.BorderLayout());


changePwdButton.setText("<html><center>Изменить<br>пароль</center></
html>");
        changePwdButton.setMaximumSize(new java.awt.Dimension(103,
23));
        changePwdButton.setMinimumSize(new java.awt.Dimension(103,
23));
        changePwdButton.setPreferredSize(new java.awt.Dimension(103,
23));
        changePwdButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                changePwdButtonMouseClicked(evt);
            }
        });
        jPanel1.add(changePwdButton, java.awt.BorderLayout.CENTER);

        disconButton.setText("Выйти из системы");
        disconButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                disconButtonMouseClicked(evt);
            }
        });
        jPanel1.add(disconButton, java.awt.BorderLayout.LINE_START);

        nameLabel.setFont(new java.awt.Font("Verdana", 0, 18)); //
NOI18N

nameLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        nameLabel.setMaximumSize(new java.awt.Dimension(999, 999));
        nameLabel.setPreferredSize(new java.awt.Dimension(630, 24));
        jPanel1.add(nameLabel, java.awt.BorderLayout.LINE_END);
        javax.swing.GroupLayout layout = new
```

```
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addComponent(jPanel1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 875, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 293, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void disconButtonMouseClicked(java.awt.event.MouseEvent
evt) {
        pkt = NetPacket.FormDisconnectRequest();
        try
        {
            niocl.write(pkt.Serialize());
        }
        catch (IOException ex)
        {
            logger.log(Level.SEVERE, "Ошибка в пакете запроса на
отключение: ", ex);
        }
        setVisible(false);
        dispose();
        udata = null;
        LoginForm.main();
    }

    private void
changePwdButtonMouseClicked(java.awt.event.MouseEvent evt) {
        SwingUtilities.invokeLater(() -> {
            changePwdDialog.pack();
            changePwdDialog.setLocationRelativeTo(this);
            changePwdDialog.setVisible(true);
        });
    }

    private void
changePwdConfirmButtonMouseClicked(java.awt.event.MouseEvent evt) {
```

```
        char[] curpwd = curPwdField.getPassword();
        char[] pwd1 = newPwdField1.getPassword();
        char[] pwd2 = newPwdField2.getPassword();
        if(pwd1.length == 0 || pwd2.length == 0)
        {
            JOptionPane.showMessageDialog(changePwdDialog, "Поле
пароля пустое", "Ошибка", JOptionPane.ERROR_MESSAGE);
            Client.nullPwds(pwd1, pwd2);
            newPwdField1.setText("");
            newPwdField2.setText("");
            return;
        }
        if(!PwdFunctions.equals(pwd1, pwd2))
        {
            JOptionPane.showMessageDialog(changePwdDialog, "Пароли
не совпадают", "Ошибка", JOptionPane.ERROR_MESSAGE);
            Client.nullPwds(pwd1, pwd2);
            newPwdField1.setText("");
            newPwdField2.setText("");
            return;
        }
        try
        {
            if(!PwdFunctions.validatePassword(curpwd, udata.pwd,
udata.salt))
            {
                JOptionPane.showMessageDialog(this, "Неверный
пароль", "Ошибка", JOptionPane.ERROR_MESSAGE);
                return;
            }
                    }
        catch (HeadlessException | NoSuchAlgorithmException |
InvalidKeySpecException ex)
        {
            logger.log(Level.SEVERE, "Ошибка при изменении пароля:
", ex);
            JOptionPane.showMessageDialog(this, "Ошибка при
изменении пароля: " + ex.getMessage(), "Ошибка",
JOptionPane.ERROR_MESSAGE);
            operationDialog.setVisible(false);
        }
        finally
        {
            newPwdField1.setText("");
            newPwdField2.setText("");
            changePwdDialog.setVisible(false);
        }
        Client.nullPwds(curpwd, null);
        showOpDialog();
        executor.execute(() -> {
            try
            {
                byte[] salt = PwdFunctions.getSalt();
```

```
                String saltstr = PwdFunctions.toHex(salt);
                String hash = PwdFunctions.hashPwd(pwd1, salt);
                Client.nullPwds(pwd1, pwd2);
                Map<Object, Map<String, Object>> changes = new
HashMap<>();
                changes.computeIfAbsent(udata.id, k -> new
HashMap<>()).put("pwd", hash);
                changes.computeIfAbsent(udata.id, k -> new
HashMap<>()).put("salt", saltstr);
                pkt = NetPacket.FormUpdateTable("users", "user_ID",
                        null, new TableRow(new Object[] {changes,
null, null}));
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                switch (pkt.getCode()) {
                    case SQL_UPDATE_RESPONSE:
                    {
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(this,
"Пароль успешно изменён!", "Сообщение",
JOptionPane.INFORMATION_MESSAGE);
                        });
                        break;
                    }
                    case ERROR:
                    {
                        int err = (int)pkt.getData()[0];
                        String msg = (String)pkt.getData()[1];
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(this,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                        });
                        break;
                    }
                    default:
                    {
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(this,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                        });
                        break;
                    }
                }
            }
            catch (SQLException ex)
            {
                logger.log(Level.SEVERE, "Ошибка при изменении
```

```
пароля: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                    JOptionPane.showMessageDialog(this, "MySQL
вернул ошибку: " +
                            ex.getMessage() + "\nКод: " +
ex.getErrorCode(), "Ошибка", JOptionPane.ERROR_MESSAGE);
                });
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при изменении
пароля: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                    JOptionPane.showMessageDialog(this, "Ошибка при
изменении пароля: " + ex.getMessage(), "Ошибка",
JOptionPane.ERROR_MESSAGE);
                });
            }
        });
    }

    private void
changePwdDialogWindowClosing(java.awt.event.WindowEvent evt) {
        newPwdField1.setText("");
        newPwdField2.setText("");
    }

    public static void main(NetPacket pkt, String pwd, String salt)
{
        char role = (char)pkt.getData()[0];
        String name = (String)pkt.getData()[1];
        String sname = (String)pkt.getData()[2];
        String pname = (String)pkt.getData()[3];
        /* Create and display the form */
        WorkForm form;
        switch(role) // select authority
        {
        case 'S': // Student
            form = new StudentForm();
            break;
        case 'T': // Teacher
            form = new TeacherForm();
            break;
        case 'A': // Admin
            form = new AdminForm();
            break;
        default: // Should be unreachable
            return;
        }
        form.udata = new UserData((int)pkt.getData()[4], pwd, salt);
        if(name == null) name = "";
```

```
        if(sname == null) sname = "";
        if(pname == null) pname = "";
        form.nameLabel.setText(String.format("Вы вошли как %s %s
%s", sname, name, pname));
        form.pack();
        form.setVisible(true);
    }

    protected void formTable(DefaultTableModel tmodel)
    {
        tmodel.setRowCount(0);
        Object[] data = pkt.getData();
        ResMetaData rmd = (ResMetaData)data[0];
        Object[][] tbldata;
        tbldata = new Object[data.length - 1][];
        for(int i = 1; i < data.length; i++)
        {
            tbldata[i - 1] = ((TableRow)data[i]).elems;
        }
        tmodel.setDataVector(tbldata, rmd.colLabels);
    }

    protected void formColnames()
    {
        ResMetaData rmd = (ResMetaData)pkt.getData()[0];
        for(int i = 0; i < rmd.colLabels.length; i++)
        {
            colnames.put(rmd.colLabels[i], rmd.colNames[i]);
        }
    }

    protected void showOpDialog()
    {
        SwingUtilities.invokeLater(() -> {
            operationDialog.pack();
            operationDialog.setLocationRelativeTo(this);
            operationDialog.setVisible(true);
        });
    }

    protected void reset()
    {
        colnames.clear();
    }
    protected void WindowClosingAction(JTable t)
    {
        reset();
        t.setModel(new DefaultTableModel());
    }

    // Variables declaration - do not modify
    private javax.swing.JButton changePwdButton;
    private javax.swing.JButton changePwdConfirmButton;
```

```
    private javax.swing.JDialog changePwdDialog;
    private javax.swing.JLabel connectLabel;
    private javax.swing.JProgressBar connectProgressBar;
    private javax.swing.JPasswordField curPwdField;
    private javax.swing.JButton disconButton;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JPasswordField newPwdField1;
    private javax.swing.JPasswordField newPwdField2;
    protected javax.swing.JDialog operationDialog;
    // End of variables declaration
}
```

Файл «*LoginForm.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;
import static client.Client.logger;
import static client.Client.niocl;
import java.io.*;
import java.util.logging.Level;
import my.*;
import javax.swing.*;
/**
 *
 * @author Admin
 */
public class LoginForm extends AbstractForm {
    /**
     * Creates new form LoginForm
     */
    public LoginForm() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        connectDialog = new javax.swing.JDialog();
```

```
        connectProgressBar = new javax.swing.JProgressBar();
        connectLabel = new javax.swing.JLabel();
        promptLabel = new javax.swing.JLabel();
        lgnLabel = new javax.swing.JLabel();
        pwdLabel = new javax.swing.JLabel();
        loginField = new javax.swing.JTextField();
        pwdField = new javax.swing.JPasswordField();
        conButton = new javax.swing.JButton();


connectDialog.setModalityType(java.awt.Dialog.ModalityType.APPLICATI
ON_MODAL);
        connectDialog.setResizable(false);
        connectDialog.setUndecorated(true);

connectDialog.getRootPane().setBorder(BorderFactory.createRaisedBeve
lBorder());

        connectProgressBar.setIndeterminate(true);

        connectLabel.setFont(new java.awt.Font("Verdana", 0, 18));
// NOI18N
        connectLabel.setText("Подключение к серверу...");

        javax.swing.GroupLayout connectDialogLayout = new
javax.swing.GroupLayout(connectDialog.getContentPane());

connectDialog.getContentPane().setLayout(connectDialogLayout);
        connectDialogLayout.setHorizontalGroup(

connectDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(connectDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(connectDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING, false)
                    .addComponent(connectLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(connectProgressBar,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        connectDialogLayout.setVerticalGroup(

connectDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(connectDialogLayout.createSequentialGroup()
```

```
                .addGap(39, 39, 39)
                .addComponent(connectLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATE
D)
                .addComponent(connectProgressBar,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(35, Short.MAX_VALUE))
        );


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        promptLabel.setFont(new java.awt.Font("Verdana", 0, 18)); //
NOI18N
        promptLabel.setText("Пожалуйста, авторизируйтесь");

        lgnLabel.setFont(new java.awt.Font("Verdana", 0, 18)); //
NOI18N
        lgnLabel.setText("Логин");

        pwdLabel.setFont(new java.awt.Font("Verdana", 0, 18)); //
NOI18N
        pwdLabel.setText("Пароль");

        conButton.setText("Подключиться");
        conButton.addMouseListener(new java.awt.event.MouseAdapter()
{
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                conButtonMouseClicked(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(layout.createSequentialGroup()
                .addGap(151, 151, 151)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.TRAILING)
                    .addComponent(pwdLabel)
                    .addComponent(lgnLabel))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
```

```
nt.LEADING, false)
                    .addComponent(promptLabel)
                    .addComponent(loginField,
javax.swing.GroupLayout.DEFAULT_SIZE, 300, Short.MAX_VALUE)
                    .addComponent(pwdField))
                .addContainerGap(203, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(conButton)
                .addContainerGap())
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(promptLabel)
                .addGap(49, 49, 49)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                    .addComponent(lgnLabel)
                    .addComponent(loginField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                    .addComponent(pwdLabel)
                    .addComponent(pwdField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
206, Short.MAX_VALUE)
                .addComponent(conButton)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void conButtonMouseClicked(java.awt.event.MouseEvent
evt) {
        SwingUtilities.invokeLater(() -> {
            connectDialog.pack();
```

```
            connectDialog.setLocationRelativeTo(this);
            connectDialog.setVisible(true);
        });
        executor.execute(() -> {
            try
            {
                String login = loginField.getText();
                pkt = NetPacket.FormFetchPassword(login);
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                if(pkt.getCode() !=
NetPacket.Opcode.FETCH_PASSWORD_RESPONSE)
                    throw new LoginFailedException();
                String pwdhash = (String)pkt.getData()[0];
                String salt = (String)pkt.getData()[1];
                char[] pwd = pwdField.getPassword();
                if(!PwdFunctions.validatePassword(pwd, pwdhash,
salt))
                {
                    Client.nullPwds(pwd, null);
                    throw new LoginFailedException();
                }
                Client.nullPwds(pwd, null);
                pkt = NetPacket.FormLoginSuccess(login);
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                SwingUtilities.invokeLater(() -> {
                    connectDialog.setVisible(false);
                    connectDialog.dispose();
                    setVisible(false);
                    dispose();
                    WorkForm.main(pkt, pwdhash, salt);
                    pkt = null;
                });
            }
            catch(LoginFailedException | IOException ex)
            {
                SwingUtilities.invokeLater(() -> {
                    connectDialog.setVisible(false);
                    JOptionPane.showMessageDialog(this,
ex.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
                });
                logger.log(Level.SEVERE, "Ошибка авторизации: ",
ex);
            }
            catch(Exception ex)
            {
                SwingUtilities.invokeLater(() -> {
                    connectDialog.setVisible(false);
```

```
                });
                logger.log(Level.SEVERE, "Ошибка авторизации: ",
ex);
            }
        });
    }

     public static void main() {
        /* Create and display the form */
        SwingUtilities.invokeLater(() -> {
            LoginForm form = new LoginForm();
            form.pack();
            form.setVisible(true);
            });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton conButton;
    private javax.swing.JDialog connectDialog;
    private javax.swing.JLabel connectLabel;
    private javax.swing.JProgressBar connectProgressBar;
    private javax.swing.JLabel lgnLabel;
    private javax.swing.JTextField loginField;
    private javax.swing.JLabel promptLabel;
    private javax.swing.JPasswordField pwdField;
    private javax.swing.JLabel pwdLabel;
    // End of variables declaration
}
```

Файл «*LoginFailedException.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;

/**
 *
 * @author Admin
 */
public class LoginFailedException extends Exception {
    public LoginFailedException()
    {
        super("Ошибка входа в систему: неверные имя пользователя или
пароль.");
    }
}
```

Файл «*StudentForm.java*»

```
/*
 * To change this license header, choose License Headers in Project
```

```
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;

import static client.Client.logger;
import static client.Client.niocl;
import java.util.logging.Level;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import my.NetPacket;

/**
 *
 * @author Admin
 */
public class StudentForm extends WorkForm {

    /**
     * Creates new form StudentForm
     */
    protected StudentForm() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        hwFrame = new javax.swing.JDialog();
        jScrollPane1 = new javax.swing.JScrollPane();
        hwTable = new javax.swing.JTable();
        mFrame = new javax.swing.JDialog();
        jScrollPane2 = new javax.swing.JScrollPane();
        mTable = new javax.swing.JTable();
        jPanel1 = new javax.swing.JPanel();
        mButton = new javax.swing.JButton();
        hwButton = new javax.swing.JButton();

        hwFrame.setTitle("Домашнее задание");
        hwFrame.setModal(true);
        hwFrame.addWindowListener(new java.awt.event.WindowAdapter()
{
            public void windowClosing(java.awt.event.WindowEvent
evt) {
```

```
                hwFrameWindowClosing(evt);
            }
        });

        hwTable.setAutoCreateRowSorter(true);
        hwTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {

            }
        ));
        jScrollPane1.setViewportView(hwTable);

        javax.swing.GroupLayout hwFrameLayout = new
javax.swing.GroupLayout(hwFrame.getContentPane());
        hwFrame.getContentPane().setLayout(hwFrameLayout);
        hwFrameLayout.setHorizontalGroup(

hwFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addGap(0, 900, Short.MAX_VALUE)

.addGroup(hwFrameLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 900, Short.MAX_VALUE))
        );
        hwFrameLayout.setVerticalGroup(

hwFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addGap(0, 560, Short.MAX_VALUE)

.addGroup(hwFrameLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 560, Short.MAX_VALUE))
        );

        mFrame.setTitle("Оценки");
        mFrame.setModal(true);
        mFrame.addWindowListener(new java.awt.event.WindowAdapter()
{
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                mFrameWindowClosing(evt);
            }
        });

        mTable.setAutoCreateRowSorter(true);
        mTable.setModel(new javax.swing.table.DefaultTableModel(
```

```
            new Object [][] {
                {},
                {},
                {},
                {}
            },
            new String [] {

            }
        ));
        jScrollPane2.setViewportView(mTable);

        javax.swing.GroupLayout mFrameLayout = new
javax.swing.GroupLayout(mFrame.getContentPane());
        mFrame.getContentPane().setLayout(mFrameLayout);
        mFrameLayout.setHorizontalGroup(

mFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 900, Short.MAX_VALUE)
        );
        mFrameLayout.setVerticalGroup(

mFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 560, Short.MAX_VALUE)
        );


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setMaximumSize(new java.awt.Dimension(2147483647,
2147483647));
        jPanel1.setMinimumSize(new java.awt.Dimension(0, 0));
        jPanel1.setPreferredSize(new java.awt.Dimension(875, 341));
        jPanel1.setLayout(new java.awt.GridLayout(2, 1, 20, 10));

        mButton.setText("Просмотреть оценки");
        mButton.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                mButtonMouseClicked(evt);
            }
        });
        jPanel1.add(mButton);

        hwButton.setText("Просмотреть домашнее задание");
        hwButton.addMouseListener(new java.awt.event.MouseAdapter()
{
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
```

```
                hwButtonMouseClicked(evt);
            }
        });
        jPanel1.add(hwButton);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addComponent(jPanel1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(86, Short.MAX_VALUE)
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 120,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(135, 135, 135))
        );

        pack();
    }// </editor-fold>

    private void hwButtonMouseClicked(java.awt.event.MouseEvent evt)
{
        showOpDialog();
        executor.execute(() -> {
            try {
                pkt = NetPacket.FormSqlQuery("SELECT subject AS
'Предмет', DATE_FORMAT(hw_date, '%d.%m.%Y')"
                        + "AS 'Дата', homework AS 'Задание'"
                        + "FROM homework RIGHT OUTER JOIN (SELECT
student_info.class FROM student_info WHERE student_ID = ?)"
                        + "AS s ON s.class=homework.class ORDER BY
'Дата' ASC", udata.id);
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                SwingUtilities.invokeLater(() -> {
                    switch (pkt.getCode()) {
                        case SQL_RESPONSE:
                        {
```

```
                                DefaultTableModel t = new
DefaultTableModel()
                                {
                                    @Override
                                    public boolean isCellEditable(int
row, int column)
                                    {return false;}
                                };
                                formTable(t);
                                formColnames();
                                hwTable.setModel(t);
                                SwingUtilities.invokeLater(() -> {
                                    hwFrame.pack();
                                    hwFrame.setLocationRelativeTo(this);
                                    hwFrame.setVisible(true);
                                });
                                break;
                            }
                            case ERROR:
                            {
                                int err = (int)pkt.getData()[0];
                                String msg = (String)pkt.getData()[1];
                                JOptionPane.showMessageDialog(this,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                                break;
                            }
                            default:
                            {
                                JOptionPane.showMessageDialog(this,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                                break;
                            }
                        }
                        operationDialog.setVisible(false);
                    });
                }
                catch (Exception ex)
                {
                    logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                    SwingUtilities.invokeLater(() -> {
                        operationDialog.setVisible(false);
                    });
                }
            });
        }

        private void mButtonMouseClicked(java.awt.event.MouseEvent evt)
{
            showOpDialog();
            executor.execute(() -> {
                try {
```

```
                pkt = NetPacket.FormSqlQuery("SELECT subject AS
'Предмет', DATE_FORMAT(m_date, '%d.%m.%Y') AS 'Дата',"
                    + "mark AS 'Оценка' FROM marks WHERE student_ID
= ? ORDER BY 'Дата' ASC", udata.id);
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                SwingUtilities.invokeLater(() -> {
                    switch (pkt.getCode()) {
                        case SQL_RESPONSE:
                        {
                            DefaultTableModel t = new
DefaultTableModel()
                            {
                                @Override
                                public boolean isCellEditable(int
row, int column)
                                {return false;}
                            };
                            formTable(t);
                            formColnames();
                            mTable.setModel(t);
                            SwingUtilities.invokeLater(() -> {
                                mFrame.pack();
                                mFrame.setLocationRelativeTo(this);
                                mFrame.setVisible(true);
                            });
                            break;
                        }
                        case ERROR:
                        {
                            int err = (int)pkt.getData()[0];
                            String msg = (String)pkt.getData()[1];
                            JOptionPane.showMessageDialog(this,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                            break;
                        }
                        default:
                        {
                            JOptionPane.showMessageDialog(this,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                            break;
                        }
                    }
                    operationDialog.setVisible(false);
                });
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
```

```
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
        });

    }

    private void hwFrameWindowClosing(java.awt.event.WindowEvent
evt) {
        WindowClosingAction(hwTable);
    }

    private void mFrameWindowClosing(java.awt.event.WindowEvent evt)
{
        WindowClosingAction(mTable);
    }

    // Variables declaration - do not modify
    private javax.swing.JButton hwButton;
    private javax.swing.JDialog hwFrame;
    private javax.swing.JTable hwTable;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JButton mButton;
    private javax.swing.JDialog mFrame;
    private javax.swing.JTable mTable;
    // End of variables declaration
}
```

Файл «*TeacherForm.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;

import static client.Client.logger;
import static client.Client.niocl;
import java.io.IOException;
import java.text.*;
import java.util.*;
import java.util.logging.Level;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.table.*;
import static javax.swing.event.TableModelEvent.DELETE;
import static javax.swing.event.TableModelEvent.INSERT;
import static javax.swing.event.TableModelEvent.UPDATE;
import my.*;
```

```
/**
 *
 * @author Admin
 */
public class TeacherForm extends EditableForm {
    private TableRow currentrow;
    private boolean callFromInsert;
    private Map<String, JComboBox> classessubjects;
    public static final JComboBox marks;
    static
    {
        marks = new JComboBox();
        marks.addItem("1");
        marks.addItem("2");
        marks.addItem("3");
        marks.addItem("4");
        marks.addItem("5");
        marks.addItem("н");
        marks.addItem("-");
    }
    /**
     * Creates new form TeacherForm
     */
    protected TeacherForm() {
        initComponents();
        callFromInsert = false;
    }

    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        bindingGroup = new org.jdesktop.beansbinding.BindingGroup();

        hwFrame = new javax.swing.JDialog();
        jScrollPane1 = new javax.swing.JScrollPane();
        hwTable = new javax.swing.JTable();
        hwConfirmButton = new javax.swing.JButton();
        addButton = new javax.swing.JButton();
        hwPopupMenu = new javax.swing.JPopupMenu();
        addMenuItem = new javax.swing.JMenuItem();
        rmMenuItem = new javax.swing.JMenuItem();
        mFrame = new javax.swing.JDialog();
        jScrollPane2 = new javax.swing.JScrollPane();
        mTable = new javax.swing.JTable();
        mConfirmButton = new javax.swing.JButton();
        mAddButton = new javax.swing.JButton();
```

```
        jPanel2 = new javax.swing.JPanel();
        classLabel = new javax.swing.JLabel();
        subjectLabel = new javax.swing.JLabel();
        mPopupMenu = new javax.swing.JPopupMenu();
        mAddMenuItem = new javax.swing.JMenuItem();
        mRmMenuItem = new javax.swing.JMenuItem();
        chooseClassSubjectDialog = new javax.swing.JDialog();
        jLabel6 = new javax.swing.JLabel();
        chooseClassSubjectConfirmButton = new javax.swing.JButton();
        jLabel7 = new javax.swing.JLabel();
        subjectComboBox = new javax.swing.JComboBox<>();
        classComboBox = new javax.swing.JComboBox<>();
        jLabel1 = new javax.swing.JLabel();
        jPanel1 = new javax.swing.JPanel();
        hwButton = new javax.swing.JButton();
        mButton = new javax.swing.JButton();

        hwFrame.setTitle("Домашнее задание");
        hwFrame.setModal(true);
        hwFrame.addWindowListener(new java.awt.event.WindowAdapter()
{
            public void windowClosed(java.awt.event.WindowEvent evt)
{
                hwFrameWindowClosed(evt);
            }
        });

        hwTable.setAutoCreateRowSorter(true);

hwTable.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_NEXT_COLUMN
);

hwTable.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELEC
TION);
        hwTable.getTableHeader().setReorderingAllowed(false);

        org.jdesktop.beansbinding.Binding binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.be
ansbinding.AutoBinding.UpdateStrategy.READ_WRITE, hwPopupMenu,
org.jdesktop.beansbinding.ObjectProperty.create(), hwTable,
org.jdesktop.beansbinding.BeanProperty.create("componentPopupMenu"))
;
        bindingGroup.addBinding(binding);

        jScrollPane1.setViewportView(hwTable);

        hwConfirmButton.setText("Подтвердить изменения");
        hwConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                hwConfirmButtonMouseClicked(evt);
            }
```

```
        });
        addButton.setText("Добавить строку");
        addButton.addMouseListener(new java.awt.event.MouseAdapter()
{
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                addButtonMouseClicked(evt);
            }
        });

        javax.swing.GroupLayout hwFrameLayout = new
javax.swing.GroupLayout(hwFrame.getContentPane());
        hwFrame.getContentPane().setLayout(hwFrameLayout);
        hwFrameLayout.setHorizontalGroup(

hwFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 888, Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
hwFrameLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(addButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(hwConfirmButton)
                .addContainerGap())
        );
        hwFrameLayout.setVerticalGroup(

hwFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addGroup(hwFrameLayout.createSequentialGroup()
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 480,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
45, Short.MAX_VALUE)

.addGroup(hwFrameLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(hwConfirmButton)
                    .addComponent(addButton))
                .addContainerGap())
        );

        hwPopupMenu.setPreferredSize(new java.awt.Dimension(200,
100));

        addMenuItem.setText("Добавить строку");
```

```
        addMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                addMenuItemActionPerformed(evt);
            }
        });
        hwPopupMenu.add(addMenuItem);

        rmMenuItem.setText("Удалить строку");
        rmMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                rmMenuItemActionPerformed(evt);
            }
        });
        hwPopupMenu.add(rmMenuItem);

        mFrame.setTitle("Оценки");
        mFrame.setModal(true);
        mFrame.addWindowListener(new java.awt.event.WindowAdapter()
{
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                mFrameWindowClosing(evt);
            }
        });

        mTable.setAutoCreateRowSorter(true);

mTable.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_NEXT_COLUMN)
;

mTable.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECT
ION);
        mTable.getTableHeader().setReorderingAllowed(false);

        binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.be
ansbinding.AutoBinding.UpdateStrategy.READ_WRITE, mPopupMenu,
org.jdesktop.beansbinding.ObjectProperty.create(), mTable,
org.jdesktop.beansbinding.BeanProperty.create("componentPopupMenu"))
;
        bindingGroup.addBinding(binding);

        jScrollPane2.setViewportView(mTable);
        mTable.getSelectionModel().addListSelectionListener(new
ListSelectionListener() {

            public void valueChanged(ListSelectionEvent lse) {
                int idx =
((ListSelectionModel)lse.getSource()).getMinSelectionIndex();
```

```
                if(idx != -1)
                {
                    currentrow =
TableRow.FromTableModel(mTable.getModel(), idx, 0,
mTable.getModel().getColumnCount());
                }
            }
        });

        mConfirmButton.setText("Подтвердить изменения");
        mConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                mConfirmButtonMouseClicked(evt);
            }
        });

        mAddButton.setText("Добавить строку");
        mAddButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                mAddButtonMouseClicked(evt);
            }
        });

        jPanel2.setLayout(new java.awt.BorderLayout());

        classLabel.setFont(new java.awt.Font("Verdana", 0, 18)); //
NOI18N
        classLabel.setText("Класс: ");
        classLabel.setPreferredSize(new java.awt.Dimension(200,
23));
        jPanel2.add(classLabel, java.awt.BorderLayout.LINE_START);

        subjectLabel.setFont(new java.awt.Font("Verdana", 0, 18));
// NOI18N
        subjectLabel.setText("Предмет: ");
        subjectLabel.setPreferredSize(new java.awt.Dimension(200,
23));
        jPanel2.add(subjectLabel, java.awt.BorderLayout.CENTER);

        javax.swing.GroupLayout mFrameLayout = new
javax.swing.GroupLayout(mFrame.getContentPane());
        mFrame.getContentPane().setLayout(mFrameLayout);
        mFrameLayout.setHorizontalGroup(

mFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
mFrameLayout.createSequentialGroup()
```

```
                .addContainerGap()
                .addComponent(mAddButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
590, Short.MAX_VALUE)
                .addComponent(mConfirmButton)
                .addContainerGap())
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        mFrameLayout.setVerticalGroup(

mFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(mFrameLayout.createSequentialGroup()
                .addComponent(jPanel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, 0)
                .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 430,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
53, Short.MAX_VALUE)

.addGroup(mFrameLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.BASELINE)
                    .addComponent(mConfirmButton)
                    .addComponent(mAddButton))
                .addContainerGap())
        );

        mPopupMenu.setPreferredSize(new java.awt.Dimension(200,
100));

        mAddMenuItem.setText("Добавить строку");
        mAddMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                mAddMenuItemActionPerformed(evt);
            }
        });
        mPopupMenu.add(mAddMenuItem);

        mRmMenuItem.setText("Удалить строку");
        mRmMenuItem.setToolTipText("");
        mRmMenuItem.addActionListener(new
java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                mRmMenuItemActionPerformed(evt);
            }
        });
        mPopupMenu.add(mRmMenuItem);

        chooseClassSubjectDialog.setTitle("Выберите класс и
предмет");
        chooseClassSubjectDialog.setModal(true);

        jLabel6.setText("Класс");

        chooseClassSubjectConfirmButton.setText("Подтвердить");
        chooseClassSubjectConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                chooseClassSubjectConfirmButtonMouseClicked(evt);
            }
        });

        jLabel7.setText("Предмет");

        subjectComboBox.setEditable(true);

        classComboBox.setEditable(true);

        jLabel1.setFont(new java.awt.Font("Verdana", 0, 18)); //
NOI18N

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Выберите класс и предмет");

jLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER)
;

        javax.swing.GroupLayout chooseClassSubjectDialogLayout = new
javax.swing.GroupLayout(chooseClassSubjectDialog.getContentPane());

chooseClassSubjectDialog.getContentPane().setLayout(chooseClassSubje
ctDialogLayout);
        chooseClassSubjectDialogLayout.setHorizontalGroup(

chooseClassSubjectDialogLayout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)

.addGroup(chooseClassSubjectDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(chooseClassSubjectDialogLayout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.LEADING)
```

```
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
chooseClassSubjectDialogLayout.createSequentialGroup()
                        .addGap(0, 0, Short.MAX_VALUE)

.addGroup(chooseClassSubjectDialogLayout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.TRAILING)

.addComponent(chooseClassSubjectConfirmButton)

.addGroup(chooseClassSubjectDialogLayout.createSequentialGroup()

.addGroup(chooseClassSubjectDialogLayout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.TRAILING)
                                    .addComponent(jLabel6)
                                    .addComponent(jLabel7))
                                .addGap(18, 18, 18)

.addGroup(chooseClassSubjectDialogLayout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.LEADING, false)
                                    .addComponent(classComboBox, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                    .addComponent(subjectComboBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 200,
javax.swing.GroupLayout.PREFERRED_SIZE)))))
                    .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap())
        );
        chooseClassSubjectDialogLayout.setVerticalGroup(

chooseClassSubjectDialogLayout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
chooseClassSubjectDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(chooseClassSubjectDialogLayout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel6)
                    .addComponent(classComboBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(chooseClassSubjectDialogLayout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel7)
```

```
                    .addComponent(subjectComboBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addComponent(chooseClassSubjectConfirmButton)
                .addContainerGap())
        );


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setMaximumSize(new java.awt.Dimension(2147483647,
2147483647));
        jPanel1.setMinimumSize(new java.awt.Dimension(0, 0));
        jPanel1.setPreferredSize(new java.awt.Dimension(875, 341));
        jPanel1.setLayout(new java.awt.GridLayout(2, 1, 20, 10));

        hwButton.setText("Редактировать домашнее задание");
        hwButton.addMouseListener(new java.awt.event.MouseAdapter()
{
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                hwButtonMouseClicked(evt);
            }
        });
        jPanel1.add(hwButton);

        mButton.setText("Редактировать оценки");
        mButton.setToolTipText("");
        mButton.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                mButtonMouseClicked(evt);
            }
        });
        jPanel1.add(mButton);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
```

```
            .addGroup(layout.createSequentialGroup()
                .addGap(110, 110, 110)
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 120,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(111, Short.MAX_VALUE))
        );

        bindingGroup.bind();

        pack();
    }// </editor-fold>

    private void hwButtonMouseClicked(java.awt.event.MouseEvent evt)
{
        showOpDialog();
        executor.execute(() -> {
            try
            {
                fetchHwTable();
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
        });
    }

    private void mButtonMouseClicked(java.awt.event.MouseEvent evt)
{
        showOpDialog();
        executor.execute(() -> {
            try
            {
                pkt = NetPacket.FormSqlQuery("SELECT class, subject
FROM classes WHERE teacher_ID = ?", udata.id);
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                switch (pkt.getCode()) {
                case SQL_RESPONSE:
                {
                    classComboBox.setModel(formClassesSubjects(0,
1));

subjectComboBox.setModel(classessubjects.getOrDefault(classComboBox.
getSelectedItem(), new JComboBox()).getModel());
```

```
                    SwingUtilities.invokeLater(() -> {
                        chooseClassSubjectDialog.pack();

chooseClassSubjectDialog.setLocationRelativeTo(this);
                        chooseClassSubjectDialog.setVisible(true);
                    });
                    break;
                }
                case ERROR:
                {
                    int err = (int)pkt.getData()[0];
                    String msg = (String)pkt.getData()[1];
                    JOptionPane.showMessageDialog(this, "MySQL
вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                    break;
                }
                default:
                {
                    JOptionPane.showMessageDialog(this, "Неизвестная
ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                    break;
                }
                }
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
            }
            finally
            {
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
        });
    }

    private void
addMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        onAddAction();
    }

    private void
rmMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedRow = hwTable.getSelectedRow();
        DefaultTableModel mdl =
(DefaultTableModel)hwTable.getModel();
        if(selectedRow != -1)
        {
            Object primary = mdl.getValueAt(selectedRow, 0);
            if(primary == null)
```

```
            {
                TableRow secondary = new TableRow(new Object[]{
                    mdl.getValueAt(selectedRow, 1),
                    mdl.getValueAt(selectedRow, 2),
                    mdl.getValueAt(selectedRow, 3),
                    mdl.getValueAt(selectedRow, 4),
                    null
                });
                insertions.removeIf(e -> e.equals(secondary));
            }
            else
            {
                changes.remove(primary);
                deletions.add(primary);
            }
            mdl.removeRow(selectedRow);
        }
    }

    private void addButtonMouseClicked(java.awt.event.MouseEvent
evt) {
        onAddAction();
    }

    private void
hwConfirmButtonMouseClicked(java.awt.event.MouseEvent evt) {
        if(changes.isEmpty() && deletions.isEmpty() &&
insertions.isEmpty())
            return;
        showOpDialog();
        executor.execute(() -> {
            try
            {
                pkt = NetPacket.FormUpdateTable("homework",
"entry_ID",
                        "entry_ID", new TableRow(new Object[]
{changes, deletions, null}));
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                switch (pkt.getCode()) {
                    case SQL_UPDATE_RESPONSE:
                        break;
                    case ERROR:
                    {
                        int err = (int)pkt.getData()[0];
                        String msg = (String)pkt.getData()[1];
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(hwFrame,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
```

```
                        });
                        return;
                    }
                    default:
                    {
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(hwFrame,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                }
                //Инсёрты делаем отдельно, чтобы форматировать дату
                pkt = NetPacket.FormUpdateBatch("INSERT IGNORE INTO
homework VALUES (?,?,STR_TO_DATE(?, '%d.%m.%Y'),?,NULL)",
insertions.toArray(new TableRow[0]));
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                switch (pkt.getCode()) {
                    case SQL_UPDATE_RESPONSE:
                        break;
                    case ERROR:
                    {
                        int err = (int)pkt.getData()[0];
                        String msg = (String)pkt.getData()[1];
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(hwFrame,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                    default:
                    {
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(hwFrame,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                }
                fetchHwTable();
            }
            catch (IOException ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
```

```
                    operationDialog.setVisible(false);
                });
                JOptionPane.showMessageDialog(hwFrame, "Ошибка
соединения с сервером: " + ex.getMessage(), "Ошибка",
JOptionPane.ERROR_MESSAGE);
                System.exit(1);
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
            finally
            {
                reset();
            }
        });
    }

    private void mAddButtonMouseClicked(java.awt.event.MouseEvent
evt) {
        onMAddAction();
    }

    private void
mAddMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        onMAddAction();
    }

    private void
mRmMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedRow = mTable.getSelectedRow();
        DefaultTableModel mdl =
(DefaultTableModel)mTable.getModel();
        if(selectedRow != -1)
        {
            Object primary = mdl.getValueAt(selectedRow, 0);
            if(primary == null)
            {
                TableRow secondary = new TableRow(new Object[]{
                    mdl.getValueAt(selectedRow, 1),
                    mdl.getValueAt(selectedRow, 5),
                    mdl.getValueAt(selectedRow, 6),
                    mdl.getValueAt(selectedRow, 7),
                    null
                });
                insertions.removeIf(e -> e.equals(secondary));
            }
            else
            {
```

```
                changes.remove(primary);
                deletions.add(primary);
            }
            mdl.removeRow(selectedRow);
        }
    }

    private void hwFrameWindowClosed(java.awt.event.WindowEvent evt)
{
        WindowClosingAction(hwTable);
    }

    private void mFrameWindowClosing(java.awt.event.WindowEvent evt)
{
        WindowClosingAction(mTable);
    }

    private void
mConfirmButtonMouseClicked(java.awt.event.MouseEvent evt) {
        if(deletions.isEmpty() && insertions.isEmpty())
            return;
        showOpDialog();
        executor.execute(() -> {
            try
            {
                pkt = NetPacket.FormUpdateTable("marks", null,
                        "entry_ID", new TableRow(new Object[] {null,
deletions, null}));
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                switch (pkt.getCode()) {
                    case SQL_UPDATE_RESPONSE:
                        break;
                    case ERROR:
                    {
                        int err = (int)pkt.getData()[0];
                        String msg = (String)pkt.getData()[1];
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(mFrame,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                    default:
                    {
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(mFrame,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
```

```
                        });
                        return;
                    }
                }
                pkt = NetPacket.FormUpdateBatch("CALL
InsertMark(?,?,?,?,?,?,?)", insertions.toArray(new TableRow[0]));
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                switch (pkt.getCode()) {
                    case SQL_UPDATE_RESPONSE:
                        break;
                    case ERROR:
                    {
                        int err = (int)pkt.getData()[0];
                        String msg = (String)pkt.getData()[1];
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(mFrame,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                    default:
                    {
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(mFrame,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                }
                fetchMTable(classLabel.getText().replace("Класс: ",
""), subjectLabel.getText().replace("Предмет: ", ""));
            }
            catch (IOException ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
                JOptionPane.showMessageDialog(mFrame, "Ошибка
соединения с сервером: " + ex.getMessage(), "Ошибка",
JOptionPane.ERROR_MESSAGE);
                System.exit(1);
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
```

```
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
            finally
            {
                reset();
            }
        });
    }

    private void
chooseClassSubjectConfirmButtonMouseClicked(java.awt.event.MouseEven
t evt) {
        String clazz = (String)classComboBox.getSelectedItem();
        String subj = (String)subjectComboBox.getSelectedItem();
        if(clazz == null || subj == null ||clazz.isEmpty() ||
subj.isEmpty())
        {
            SwingUtilities.invokeLater(() -> {

JOptionPane.showMessageDialog(chooseClassSubjectDialog, "Не выбран
класс или предмет", "Ошибка", JOptionPane.ERROR_MESSAGE);
            });
            return;
        }
        showOpDialog();
        executor.execute(() -> {
            try
            {
                fetchMTable(clazz, subj);
                chooseClassSubjectDialog.setVisible(false);
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
        });
    }

    private void hwTableChanged(javax.swing.event.TableModelEvent
evt)
    {
        DefaultTableModel mdl = (DefaultTableModel)evt.getSource();
        if(callFromInsert)
        {
            callFromInsert = false;
            return;
```

```
        }
        switch(evt.getType())
        {
            case UPDATE:
            {
                int row = evt.getFirstRow();
                int col = evt.getColumn();
                String colname = mdl.getColumnName(col);
                Object key = mdl.getValueAt(row, 0);
                Object val = mdl.getValueAt(row, col);
                String clazz = (String)mdl.getValueAt(row, 1);
                JComboBox b1 =
(JComboBox)((DefaultCellEditor)hwTable.getColumnModel().getColumn(1)
.getCellEditor()).getComponent();
                JComboBox b2 = classessubjects.get(clazz);
                if(!Objects.equals(b1, b2))

hwTable.getColumnModel().getColumn(1).setCellEditor(new
DefaultCellEditor(
                                    classessubjects.getOrDefault(clazz,
new JComboBox())));
                if(key == null)
                {
                    TableRow trow = TableRow.FromTableModel(mdl,
row, 1, mdl.getColumnCount());
                    insertions.removeIf(e -> e.equals(trow));
                    if (!Arrays.asList(trow.elems).contains(null))
                        insertions.add(trow);
                }
                else
                {
                    try
                    {
                        changes.computeIfAbsent(key, k -> new
TreeMap<>()).put(colnames.get(colname), val);
                    }
                    catch(RuntimeException ex)
                    {
                        logger.log(Level.WARNING, "Ошибка при
формировании изменений: ", ex);
                    }
                }
                break;
            }
            case INSERT:
            {
                java.util.Date date = new java.util.Date();
                DateFormat df = new SimpleDateFormat("dd.MM.yyyy");
                callFromInsert = true;
                mdl.setValueAt(df.format(date), mdl.getRowCount() -
1, 3);
                break;
            }
```

```
            case DELETE:
                break;
        }
    }

    private void mTableChanged(TableModelEvent evt) {
        DefaultTableModel mdl = (DefaultTableModel)evt.getSource();
        if(callFromInsert)
        {
            callFromInsert = false;
            return;
        }
        switch(evt.getType())
        {
            case UPDATE:
            {
                int row = evt.getFirstRow();
                Object key = mdl.getValueAt(row, 0);
                if(key != null)
                {
                    deletions.add(key);
                }
                TableRow trow = new TableRow(new Object[] {
                    mdl.getValueAt(row, 1),
                    mdl.getValueAt(row, 2),
                    mdl.getValueAt(row, 3),
                    subjectLabel.getText().replace("Предмет: ", ""),
                    mdl.getValueAt(row, 4),
                    mdl.getValueAt(row, 5),
                    classLabel.getText().replace("Класс: ", "")
                });
                insertions.removeIf(e -> e.equals(currentrow));
                if (!Arrays.asList(trow.elems).contains(null))
                    insertions.add(trow);
                break;
            }
            case INSERT:
            {
                java.util.Date date = new java.util.Date();
                DateFormat df = new SimpleDateFormat("dd.MM.yyyy");
                callFromInsert = true;
                mdl.setValueAt(df.format(date), mdl.getRowCount() -
1, 4);
                break;
            }
            case DELETE:
                break;
        }
    }

    private void onAddAction()
    {
        DefaultTableModel model = (DefaultTableModel)
```

```
hwTable.getModel();
        Object[] row = new Object[model.getColumnCount()];
        model.addRow(row);
    }

    private void onMAddAction()
    {
        DefaultTableModel model = (DefaultTableModel)
mTable.getModel();
        Object[] row = new Object[model.getColumnCount()];
        model.addRow(row);
    }

    private void fetchHwTable() throws Exception
    {
        pkt = NetPacket.FormSqlQuery("SELECT entry_ID,
homework.class AS 'Класс', homework.subject AS 'Предмет'," +
"DATE_FORMAT(hw_date, '%d.%m.%Y') AS 'Дата', homework AS 'Задание'
FROM homework " +
"RIGHT OUTER JOIN (SELECT classes.subject, class FROM classes WHERE
teacher_ID = ?)" +
"AS s ON s.subject=homework.subject WHERE s.class = homework.class
ORDER BY 'Дата' ASC", udata.id);
        niocl.write(pkt.Serialize());
        niocl.read();
        pkt = NetPacket.Deserialize(niocl.peerAppData().array());
        SwingUtilities.invokeLater(() -> {
            switch (pkt.getCode()) {
                case SQL_RESPONSE:
                {
                    JComboBox classesComboBox = new
JComboBox(formClassesSubjects(1, 2));
                    DefaultTableModel t = new DefaultTableModel()
                    {
                        @Override
                        public boolean isCellEditable(int row, int
column)
                        {
                            return true;
                        }
                    };
                    formTable(t);
                    formColnames();
                    t.addTableModelListener(this::hwTableChanged);
                    hwTable.setModel(t);

hwTable.getColumnModel().getColumn(1).setCellEditor(new
DefaultCellEditor(classesComboBox));

hwTable.getColumnModel().getColumn(2).setCellEditor(new
DefaultCellEditor(
```

```
classessubjects.getOrDefault((String)classesComboBox.getSelectedItem
(), editableComboBox())));

hwTable.removeColumn(hwTable.getColumnModel().getColumn(0));
                    SwingUtilities.invokeLater(() -> {
                        hwFrame.pack();
                        hwFrame.setLocationRelativeTo(this);
                        hwFrame.setVisible(true);
                    });
                    break;
                }
                case ERROR:
                {
                    int err = (int)pkt.getData()[0];
                    String msg = (String)pkt.getData()[1];
                    JOptionPane.showMessageDialog(hwFrame, "MySQL
вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                    break;
                }
                default:
                {
                    JOptionPane.showMessageDialog(hwFrame,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                    break;
                }
            }
            operationDialog.setVisible(false);
        });
    }

    private void fetchMTable(String clazz, String subject) throws
Exception
    {
        pkt = NetPacket.FormSqlQuery("SELECT entry_ID, sname AS
'Фамилия', name AS 'Имя', pname AS 'Отчество'," +
"DATE_FORMAT(m_date, '%d.%m.%Y') AS 'Дата', mark AS 'Оценка' FROM
users " +
"RIGHT OUTER JOIN (SELECT marks.student_ID, m_date, mark, entry_ID
FROM marks " +
"RIGHT OUTER JOIN (SELECT student_ID FROM student_info WHERE class =
?)" +
"AS s ON s.student_ID = marks.student_ID WHERE subject = ?)" +
"AS s2 ON s2.student_ID = users.user_ID ORDER BY 'Дата' ASC", clazz,
subject);
        niocl.write(pkt.Serialize());
        niocl.read();
        pkt = NetPacket.Deserialize(niocl.peerAppData().array());
        SwingUtilities.invokeLater(() -> {
            switch (pkt.getCode()) {
                case SQL_RESPONSE:
                {
                    DefaultTableModel t = new DefaultTableModel()
```

```
                    {
                        @Override
                        public boolean isCellEditable(int row, int
column)
                        {return true;}
                    };
                    formTable(t);
                    for(int i = 0; i < t.getRowCount(); i++)
                    {
                        if(t.getValueAt(i, 5) == null)
                            t.setValueAt('-', i, 5);
                    }
                    formColnames();
                    t.addTableModelListener(this::mTableChanged);
                    mTable.setModel(t);

mTable.getColumnModel().getColumn(5).setCellEditor(new
DefaultCellEditor(marks));

mTable.removeColumn(mTable.getColumnModel().getColumn(0));
                    classLabel.setText("Класс: " + clazz);
                    subjectLabel.setText("Предмет: " + subject);
                    SwingUtilities.invokeLater(() -> {
                        mFrame.pack();
                        mFrame.setLocationRelativeTo(this);
                        mFrame.setVisible(true);
                    });
                    break;
                }
                case ERROR:
                {
                    int err = (int)pkt.getData()[0];
                    String msg = (String)pkt.getData()[1];
                    JOptionPane.showMessageDialog(this, "MySQL
вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                    break;
                }
                default:
                {
                    JOptionPane.showMessageDialog(this, "Неизвестная
ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                    break;
                }
            }
            operationDialog.setVisible(false);
        });
    }

    private DefaultComboBoxModel<String> formClassesSubjects(int
classidx, int subjidx)
    {
        Vector<String> items = new Vector();
```

```
        classessubjects = new HashMap<>();
        Arrays.stream(pkt.getData()).skip(1)
        .map(e -> ((TableRow)e).elems[classidx].toString())
        .distinct()
        .forEach(e -> {
             Arrays.stream(pkt.getData()).skip(1)
            .filter(e1 ->
Objects.equals(((TableRow)e1).elems[classidx].toString(), e))
            .map(e1 -> ((TableRow)e1).elems[subjidx].toString())
            .distinct()
            .forEach(e1 -> {
                classessubjects.computeIfAbsent(e, k -> new
JComboBox()).addItem(e1);
            });
            items.add(e);
        });
        return new DefaultComboBoxModel<>(items);
    }

     private JComboBox editableComboBox()
     {
         JComboBox b = new JComboBox();
         b.setEditable(true);
         return b;
     }

    // Variables declaration - do not modify
    private javax.swing.JButton addButton;
    private javax.swing.JMenuItem addMenuItem;
    private javax.swing.JButton chooseClassSubjectConfirmButton;
    private javax.swing.JDialog chooseClassSubjectDialog;
    private javax.swing.JComboBox<String> classComboBox;
    private javax.swing.JLabel classLabel;
    private javax.swing.JButton hwButton;
    private javax.swing.JButton hwConfirmButton;
    private javax.swing.JDialog hwFrame;
    private javax.swing.JPopupMenu hwPopupMenu;
    private javax.swing.JTable hwTable;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JButton mAddButton;
    private javax.swing.JMenuItem mAddMenuItem;
    private javax.swing.JButton mButton;
    private javax.swing.JButton mConfirmButton;
    private javax.swing.JDialog mFrame;
    private javax.swing.JPopupMenu mPopupMenu;
    private javax.swing.JMenuItem mRmMenuItem;
    private javax.swing.JTable mTable;
```

```
    private javax.swing.JMenuItem rmMenuItem;
    private javax.swing.JComboBox<String> subjectComboBox;
    private javax.swing.JLabel subjectLabel;
    private org.jdesktop.beansbinding.BindingGroup bindingGroup;
    // End of variables declaration
}
```

Файл «*AdminForm.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;

import static client.Client.logger;
import static client.Client.niocl;
import java.io.IOException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.util.*;
import java.util.function.Predicate;
import java.util.logging.Level;
import java.util.stream.*;
import javafx.util.Pair;
import javax.swing.*;
import javax.swing.event.*;
import static javax.swing.event.TableModelEvent.DELETE;
import static javax.swing.event.TableModelEvent.INSERT;
import static javax.swing.event.TableModelEvent.UPDATE;
import javax.swing.table.DefaultTableModel;
import my.*;
import my.NetPacket.Opcode;

/**
 *
 * @author Admin
 */
public class AdminForm extends EditableForm {
    private String oldlogin;
    private LinkedList<Pair<Integer, String>> teachers;
    /**
     * Creates new form AdminForm
     */
    protected AdminForm() {
        super();
        initComponents();
    }
    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
```

```
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        bindingGroup = new org.jdesktop.beansbinding.BindingGroup();

        usersFrame = new javax.swing.JDialog();
        jScrollPane1 = new javax.swing.JScrollPane();
        usersTable = new javax.swing.JTable();
        usersConfirmButton = new javax.swing.JButton();
        addButton = new javax.swing.JButton();
        usersPopupMenu = new javax.swing.JPopupMenu();
        addMenuItem = new javax.swing.JMenuItem();
        rmMenuItem = new javax.swing.JMenuItem();
        changePwdMenuItem = new javax.swing.JMenuItem();
        addUserDialog = new javax.swing.JDialog();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        sTextField = new javax.swing.JTextField();
        nTextField = new javax.swing.JTextField();
        pTextField = new javax.swing.JTextField();
        addUserConfirmButton = new javax.swing.JButton();
        lTextField = new javax.swing.JTextField();
        jLabel7 = new javax.swing.JLabel();
        pwdField1 = new javax.swing.JPasswordField();
        pwdField2 = new javax.swing.JPasswordField();
        roleComboBox = new javax.swing.JComboBox<>();
        addSubjectDialog = new javax.swing.JDialog();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        subjectTextField = new javax.swing.JTextField();
        classTextField = new javax.swing.JTextField();
        addSubjectConfirmButton = new javax.swing.JButton();
        teacherComboBox = new javax.swing.JComboBox<>();
        changePwdDialog = new javax.swing.JDialog();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        changePwdConfirmButton = new javax.swing.JButton();
        changePwdField1 = new javax.swing.JPasswordField();
        changePwdField2 = new javax.swing.JPasswordField();
        studentsFrame = new javax.swing.JDialog();
        jScrollPane2 = new javax.swing.JScrollPane();
        studentsTable = new javax.swing.JTable();
        studentsConfirmButton = new javax.swing.JButton();
        subjectsFrame = new javax.swing.JDialog();
        jScrollPane4 = new javax.swing.JScrollPane();
```

```
        subjectsTable = new javax.swing.JTable();
        subjectsConfirmButton = new javax.swing.JButton();
        addSubjectButton = new javax.swing.JButton();
        subjectsPopupMenu = new javax.swing.JPopupMenu();
        subjectsRmMenuItem = new javax.swing.JMenuItem();
        jPanel1 = new javax.swing.JPanel();
        changeUsersButton = new javax.swing.JButton();
        changeSubjectsButton = new javax.swing.JButton();
        changeStudentsButton = new javax.swing.JButton();

        usersFrame.setTitle("Список пользователей");

usersFrame.setModalityType(java.awt.Dialog.ModalityType.APPLICATION_
MODAL);
        usersFrame.addWindowListener(new
java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                usersFrameWindowClosing(evt);
            }
        });

        usersTable.setAutoCreateRowSorter(true);
        usersTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {},
                {},
                {},
                {}
            },
            new String [] {

            }
        ));
        usersTable.getSelectionModel().addListSelectionListener(new
ListSelectionListener() {

            public void valueChanged(ListSelectionEvent lse) {
                int idx =
((ListSelectionModel)lse.getSource()).getMinSelectionIndex();
                if(idx != -1)
                {
                    oldlogin = (String)usersTable.getValueAt(idx,
1);
                }
            }
        });

usersTable.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SE
LECTION);
        usersTable.getTableHeader().setReorderingAllowed(false);

        org.jdesktop.beansbinding.Binding binding =
```

```
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.be
ansbinding.AutoBinding.UpdateStrategy.READ_WRITE, usersPopupMenu,
org.jdesktop.beansbinding.ObjectProperty.create(), usersTable,
org.jdesktop.beansbinding.BeanProperty.create("componentPopupMenu"))
;
        bindingGroup.addBinding(binding);

        jScrollPane1.setViewportView(usersTable);

        usersConfirmButton.setText("Подтвердить изменения");
        usersConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                usersConfirmButtonMouseClicked(evt);
            }
        });

        addButton.setText("Добавить пользователя");
        addButton.addMouseListener(new java.awt.event.MouseAdapter()
{
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                addButtonMouseClicked(evt);
            }
        });

        javax.swing.GroupLayout usersFrameLayout = new
javax.swing.GroupLayout(usersFrame.getContentPane());
        usersFrame.getContentPane().setLayout(usersFrameLayout);
        usersFrameLayout.setHorizontalGroup(

usersFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 888, Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
usersFrameLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(addButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(usersConfirmButton)
                .addContainerGap())
        );
        usersFrameLayout.setVerticalGroup(

usersFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
            .addGroup(usersFrameLayout.createSequentialGroup()
                .addComponent(jScrollPane1,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE, 480,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
45, Short.MAX_VALUE)

.addGroup(usersFrameLayout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.BASELINE)
                    .addComponent(usersConfirmButton)
                    .addComponent(addButton))
                .addContainerGap())
        );

        usersPopupMenu.setPreferredSize(new java.awt.Dimension(200,
100));

        addMenuItem.setText("Добавить пользователя");
        addMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                addMenuItemActionPerformed(evt);
            }
        });
        usersPopupMenu.add(addMenuItem);

        rmMenuItem.setText("Удалить пользователя");
        rmMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                rmMenuItemActionPerformed(evt);
            }
        });
        usersPopupMenu.add(rmMenuItem);

        changePwdMenuItem.setText("Изменить пароль");
        changePwdMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                changePwdMenuItemActionPerformed(evt);
            }
        });
        usersPopupMenu.add(changePwdMenuItem);

        addUserDialog.setTitle("Создать пользователя");
        addUserDialog.setModal(true);
        addUserDialog.addWindowListener(new
java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                addUserDialogWindowClosing(evt);
```

```
            }
        });

        jLabel1.setText("Фамилия");

        jLabel2.setText("Имя");

        jLabel3.setText("Отчество");

        jLabel4.setText("Логин");

        jLabel5.setText("Пароль");

        jLabel6.setText("Повтор пароля");

        addUserConfirmButton.setText("Подтвердить");
        addUserConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                addUserConfirmButtonMouseClicked(evt);
            }
        });

        jLabel7.setText("Роль");

        roleComboBox.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "Обучающийся",
"Учитель", "Администратор" }));

        javax.swing.GroupLayout addUserDialogLayout = new
javax.swing.GroupLayout(addUserDialog.getContentPane());

addUserDialog.getContentPane().setLayout(addUserDialogLayout);
        addUserDialogLayout.setHorizontalGroup(

addUserDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(addUserDialogLayout.createSequentialGroup()

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
addUserDialogLayout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(addUserConfirmButton))

.addGroup(addUserDialogLayout.createSequentialGroup()
                        .addGap(71, 71, 71)
```

```
.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.TRAILING)
                                .addComponent(jLabel5)
                                .addComponent(jLabel4)
                                .addComponent(jLabel3)
                                .addComponent(jLabel2)
                                .addComponent(jLabel1)
                                .addComponent(jLabel6)
                                .addComponent(jLabel7))
                            .addGap(26, 26, 26)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.TRAILING)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.TRAILING)
                                            .addComponent(sTextField,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(nTextField,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(pTextField,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(lTextField,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addComponent(pwdField1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addComponent(pwdField2,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addComponent(roleComboBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(0, 87, Short.MAX_VALUE)))
                .addContainerGap())
        );
        addUserDialogLayout.setVerticalGroup(

addUserDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
```

```
            .addGroup(addUserDialogLayout.createSequentialGroup()
                .addGap(55, 55, 55)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(sTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(nTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(pTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(lTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(jLabel5)
                    .addComponent(pwdField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(jLabel6)
                    .addComponent(pwdField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(addUserDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(jLabel7)
                    .addComponent(roleComboBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
214, Short.MAX_VALUE)
                .addComponent(addUserConfirmButton)
                .addContainerGap())
        );

        addSubjectDialog.setTitle("Создать запись предмета");
        addSubjectDialog.setModal(true);
        addSubjectDialog.addWindowListener(new
java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                addSubjectDialogWindowClosing(evt);
            }
        });

        jLabel8.setText("Учитель");

        jLabel9.setText("Предмет");

        jLabel10.setText("Класс");

        addSubjectConfirmButton.setText("Подтвердить");
        addSubjectConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                addSubjectConfirmButtonMouseClicked(evt);
            }
        });

        teacherComboBox.setEditable(true);

        javax.swing.GroupLayout addSubjectDialogLayout = new
javax.swing.GroupLayout(addSubjectDialog.getContentPane());

addSubjectDialog.getContentPane().setLayout(addSubjectDialogLayout);
        addSubjectDialogLayout.setHorizontalGroup(

addSubjectDialogLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING)
```

```
            .addGroup(addSubjectDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(addSubjectDialogLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
addSubjectDialogLayout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
249, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(addSubjectConfirmButton))

.addGroup(addSubjectDialogLayout.createSequentialGroup()

.addGroup(addSubjectDialogLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.TRAILING)
                            .addComponent(jLabel10)
                            .addComponent(jLabel9)
                            .addComponent(jLabel8))
                        .addGap(26, 26, 26)

.addGroup(addSubjectDialogLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)

.addGroup(addSubjectDialogLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.TRAILING)
                                .addComponent(subjectTextField,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(classTextField,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addComponent(teacherComboBox,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        addSubjectDialogLayout.setVerticalGroup(

addSubjectDialogLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING)
            .addGroup(addSubjectDialogLayout.createSequentialGroup()
                .addGap(55, 55, 55)

.addGroup(addSubjectDialogLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.BASELINE)
                    .addComponent(jLabel8)
                    .addComponent(teacherComboBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(addSubjectDialogLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.BASELINE)
                    .addComponent(jLabel9)
                    .addComponent(subjectTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(addSubjectDialogLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.BASELINE)
                    .addComponent(jLabel10)
                    .addComponent(classTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(addSubjectConfirmButton)
                .addContainerGap())
        );

        changePwdDialog.setTitle("Изменить пароль");
        changePwdDialog.setModal(true);
        changePwdDialog.addWindowListener(new
java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                changePwdDialogWindowClosing(evt);
            }
        });

        jLabel12.setText("Пароль");

        jLabel13.setText("Повтор пароля");

        changePwdConfirmButton.setText("Подтвердить");
        changePwdConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                changePwdConfirmButtonMouseClicked(evt);
            }
        });

        javax.swing.GroupLayout changePwdDialogLayout = new
javax.swing.GroupLayout(changePwdDialog.getContentPane());
```

```
changePwdDialog.getContentPane().setLayout(changePwdDialogLayout);
        changePwdDialogLayout.setHorizontalGroup(

changePwdDialogLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.TRAILING)
            .addGroup(changePwdDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)

.addGroup(changePwdDialogLayout.createSequentialGroup()

.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.TRAILING)
                            .addComponent(jLabel12)
                            .addComponent(jLabel13))
                        .addGap(26, 26, 26)

.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.TRAILING)
                            .addComponent(changePwdField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(changePwdField2,
javax.swing.GroupLayout.PREFERRED_SIZE, 280,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
changePwdDialogLayout.createSequentialGroup()
                        .addComponent(changePwdConfirmButton)
                        .addGap(0, 0, 0))))
        );
        changePwdDialogLayout.setVerticalGroup(

changePwdDialogLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(changePwdDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jLabel12)
                    .addComponent(changePwdField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(changePwdDialogLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jLabel13)
```

```
                    .addComponent(changePwdField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(changePwdConfirmButton)
                .addGap(0, 0, 0))
        );

        studentsFrame.setTitle("Список учеников классов");

studentsFrame.setModalityType(java.awt.Dialog.ModalityType.APPLICATI
ON_MODAL);
        studentsFrame.addWindowListener(new
java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                studentsFrameWindowClosing(evt);
            }
        });

        studentsTable.setAutoCreateRowSorter(true);
        studentsTable.setModel(new
javax.swing.table.DefaultTableModel(
            new Object [][] {
                {},
                {},
                {},
                {}
            },
            new String [] {

            }
        ));

studentsTable.setSelectionMode(javax.swing.ListSelectionModel.SINGLE
_SELECTION);
        studentsTable.getTableHeader().setReorderingAllowed(false);
        jScrollPane2.setViewportView(studentsTable);

        studentsConfirmButton.setText("Подтвердить изменения");
        studentsConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                studentsConfirmButtonMouseClicked(evt);
            }
        });

        javax.swing.GroupLayout studentsFrameLayout = new
javax.swing.GroupLayout(studentsFrame.getContentPane());
```

```
studentsFrame.getContentPane().setLayout(studentsFrameLayout);
        studentsFrameLayout.setHorizontalGroup(

studentsFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 888, Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
studentsFrameLayout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(studentsConfirmButton)
                .addContainerGap())
        );
        studentsFrameLayout.setVerticalGroup(

studentsFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(studentsFrameLayout.createSequentialGroup()
                .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 480,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
45, Short.MAX_VALUE)
                .addComponent(studentsConfirmButton)
                .addContainerGap())
        );

        subjectsFrame.setTitle("Список предметов");

subjectsFrame.setModalityType(java.awt.Dialog.ModalityType.APPLICATI
ON_MODAL);
        subjectsFrame.addWindowListener(new
java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                subjectsFrameWindowClosing(evt);
            }
        });

        subjectsTable.setAutoCreateRowSorter(true);
        subjectsTable.setModel(new
javax.swing.table.DefaultTableModel(
            new Object [][] {
                {},
                {},
                {},
                {}
            },
            new String [] {
```

```
            }
        ));

subjectsTable.setSelectionMode(javax.swing.ListSelectionModel.SINGLE
_SELECTION);
        subjectsTable.getTableHeader().setReorderingAllowed(false);

        binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.be
ansbinding.AutoBinding.UpdateStrategy.READ_WRITE, subjectsPopupMenu,
org.jdesktop.beansbinding.ObjectProperty.create(), subjectsTable,
org.jdesktop.beansbinding.BeanProperty.create("componentPopupMenu"))
;
        bindingGroup.addBinding(binding);

        jScrollPane4.setViewportView(subjectsTable);

        subjectsConfirmButton.setText("Подтвердить изменения");
        subjectsConfirmButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                subjectsConfirmButtonMouseClicked(evt);
            }
        });

        addSubjectButton.setText("Добавить запись");
        addSubjectButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                addSubjectButtonMouseClicked(evt);
            }
        });

        javax.swing.GroupLayout subjectsFrameLayout = new
javax.swing.GroupLayout(subjectsFrame.getContentPane());

subjectsFrame.getContentPane().setLayout(subjectsFrameLayout);
        subjectsFrameLayout.setHorizontalGroup(

subjectsFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 888, Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
subjectsFrameLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(addSubjectButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(subjectsConfirmButton)
```

```
                .addContainerGap())
        );
        subjectsFrameLayout.setVerticalGroup(

subjectsFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(subjectsFrameLayout.createSequentialGroup()
                .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 480,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
45, Short.MAX_VALUE)

.addGroup(subjectsFrameLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(subjectsConfirmButton)
                    .addComponent(addSubjectButton))
                .addContainerGap())
        );

        subjectsPopupMenu.setPreferredSize(new
java.awt.Dimension(200, 33));

        subjectsRmMenuItem.setText("Удалить строку");
        subjectsRmMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                subjectsRmMenuItemActionPerformed(evt);
            }
        });
        subjectsPopupMenu.add(subjectsRmMenuItem);


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setMaximumSize(new java.awt.Dimension(2147483647,
2147483647));
        jPanel1.setMinimumSize(new java.awt.Dimension(0, 0));
        jPanel1.setPreferredSize(new java.awt.Dimension(875, 341));
        jPanel1.setLayout(new java.awt.GridLayout(3, 1, 20, 10));

        changeUsersButton.setText("Изменить список пользователей");
        changeUsersButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                changeUsersButtonMouseClicked(evt);
            }
        });
        jPanel1.add(changeUsersButton);
```

```
        changeSubjectsButton.setText("Изменить список предметов");
        changeSubjectsButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                changeSubjectsButtonMouseClicked(evt);
            }
        });
        jPanel1.add(changeSubjectsButton);

        changeStudentsButton.setText("Изменить списки учеников");
        changeStudentsButton.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt)
{
                changeStudentsButtonMouseClicked(evt);
            }
        });
        jPanel1.add(changeStudentsButton);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(81, Short.MAX_VALUE)
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 170,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(90, 90, 90))
        );

        bindingGroup.bind();

        pack();
    }// </editor-fold>

    private void
changeUsersButtonMouseClicked(java.awt.event.MouseEvent evt) {
        showOpDialog();
        executor.execute(() -> {
```

```
            pkt = NetPacket.FormSqlQuery("SELECT user_ID AS 'ИД',
login AS 'Логин', pwd AS 'Пароль', salt AS 'Соль', name AS 'Имя',
sname AS 'Фамилия', pname AS 'Отчество', role AS 'Роль' FROM
users");
            try
            {
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                switch (pkt.getCode()) {
                    case SQL_RESPONSE:
                    {
                        DefaultTableModel t = new
DefaultTableModel()
                        {
                            @Override
                            public boolean isCellEditable(int row,
int column)
                            {
                                //Можно редактировать только ФИО и
логин
                                return column == 1 || column == 4 ||
column == 5 || column == 6;
                            }
                        };
                        formTable(t);
                        printRoles(t);
                        formColnames();

t.addTableModelListener((javax.swing.event.TableModelEvent tevt) ->
{
                            tableChanged(tevt, e ->
Objects.equals(((TableRow)e).elems[1], oldlogin));
                        });
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            usersTable.setModel(t);
                            usersFrame.pack();
                            usersFrame.setLocationRelativeTo(this);
                            usersFrame.setVisible(true);
                        });
                        break;
                    }
                    case ERROR:
                    {
                        int err = (int)pkt.getData()[0];
                        String msg = (String)pkt.getData()[1];
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(this,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
```

```
JOptionPane.ERROR_MESSAGE);
                        });
                        break;
                    }
                    default:
                    {
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);
                            JOptionPane.showMessageDialog(this,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                        });
                        break;
                    }
                }
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
        });
    }

    private void
addMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        onAddAction();
    }

    private void
rmMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedRow = usersTable.getSelectedRow();
        DefaultTableModel mdl =
(DefaultTableModel)usersTable.getModel();
        if(selectedRow != -1)
        {
            Object primary = mdl.getValueAt(selectedRow, 0);
            if(primary == null)
            {
                final TableRow secondary =
TableRow.FromTableModel(mdl, selectedRow, 1, mdl.getColumnCount());
                insertions.removeIf(e -> e.equals(secondary));
            }
            else
            {
                changes.remove(primary);
                deletions.add(primary);
            }
            mdl.removeRow(selectedRow);
        }
    }
```

```
    private void
usersConfirmButtonMouseClicked(java.awt.event.MouseEvent evt) {
        if(changes.isEmpty() && deletions.isEmpty() &&
insertions.isEmpty())
            return;
        showOpDialog();
        executor.execute(() -> {
            try
            {
                pkt = NetPacket.FormUpdateTable("users", "user_ID",
                        "user_ID", new TableRow(new Object[]
{changes, deletions, insertions}));
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                switch (pkt.getCode()) {
                    case SQL_UPDATE_RESPONSE:
                        break;
                    case ERROR:
                    {
                        int err = (int)pkt.getData()[0];
                        String msg = (String)pkt.getData()[1];
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);

JOptionPane.showMessageDialog(usersFrame, "MySQL вернул ошибку: " +
msg + "\nКод: " + err, "Ошибка", JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                    default:
                    {
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);

JOptionPane.showMessageDialog(usersFrame, "Неизвестная ошибка",
"Ошибка", JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                }
                pkt = NetPacket.FormSqlQuery("SELECT user_ID AS
'ИД', login AS 'Логин', pwd AS 'Пароль', salt AS 'Соль', name AS
'Имя', sname AS 'Фамилия', pname AS 'Отчество', role AS 'Роль' FROM
users");
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                if(pkt.getCode() == Opcode.SQL_RESPONSE)
                {
```

```
                    TableRow[] rows =
Arrays.stream(pkt.getData()).skip(1)
                            .map(e -> new
TableRow(((TableRow)e).elems))
                            .filter(e ->
((String)e.elems[7]).charAt(0) == 'S')
                            .map(e -> new TableRow(new Object[]
{e.elems[0]}))
                            .collect(Collectors.toList())
                            .toArray(new TableRow[0]);
                    NetPacket p = NetPacket.FormUpdateBatch("INSERT
IGNORE INTO student_info VALUES (?,NULL)", rows);
                    niocl.write(p.Serialize());
                    niocl.read();
                }
                SwingUtilities.invokeLater(() -> {
                    switch (pkt.getCode()) {
                        case SQL_RESPONSE:
                        {
                           DefaultTableModel t = new
DefaultTableModel()
                            {
                                @Override
                                public boolean isCellEditable(int
row, int column)
                                {
                                    //Можно редактировать только ФИО
и логин
                                    return column == 1 || column ==
4 || column == 5 || column == 6;
                                }
                            };
                            formTable(t);
                            printRoles(t);
                            formColnames();

t.addTableModelListener((javax.swing.event.TableModelEvent tevt) ->
{
                                tableChanged(tevt, e ->
Objects.equals(((TableRow)e).elems[1], oldlogin));
                            });
                            usersTable.setModel(t);
                            break;
                        }
                        case ERROR:
                        {
                            int err = (int)pkt.getData()[0];
                            String msg = (String)pkt.getData()[1];

JOptionPane.showMessageDialog(usersFrame, "MySQL вернул ошибку: " +
msg + "\nКод: " + err, "Ошибка", JOptionPane.ERROR_MESSAGE);
                            break;
                        }
```

```
                        default:
                        {

JOptionPane.showMessageDialog(usersFrame, "Неизвестная ошибка",
"Ошибка", JOptionPane.ERROR_MESSAGE);
                            break;
                        }
                    }
                    operationDialog.setVisible(false);
                });
            }
            catch (IOException ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
                JOptionPane.showMessageDialog(usersFrame, "Ошибка
соединения с сервером: " + ex.getMessage(), "Ошибка",
JOptionPane.ERROR_MESSAGE);
                System.exit(1);
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
            finally
            {
                reset();
            }
        });
    }

    private void addButtonMouseClicked(java.awt.event.MouseEvent
evt) {
        onAddAction();
    }

    private void
addUserConfirmButtonMouseClicked(java.awt.event.MouseEvent evt) {
        char[] pwd1 = pwdField1.getPassword();
        char[] pwd2 = pwdField2.getPassword();
        String name = nTextField.getText();
        String sname = sTextField.getText();
        String pname = pTextField.getText();
        String login = lTextField.getText();
        String role = null;
        byte[] salt;
```

```
        String hash;
        try
        {
            if(login.isEmpty() ||
                pwd1.length == 0 ||
                pwd2.length == 0)
            {
                JOptionPane.showMessageDialog(addUserDialog, "Поля
логина или паролей пусты", "Ошибка", JOptionPane.ERROR_MESSAGE);
                Client.nullPwds(pwd1, pwd2);
                return;
            }
            if(!PwdFunctions.equals(pwd1, pwd2))
            {
                JOptionPane.showMessageDialog(addUserDialog, "Пароли
не совпадают", "Ошибка", JOptionPane.ERROR_MESSAGE);
                Client.nullPwds(pwd1, pwd2);
                return;
            }
            for(int i = 0; i < usersTable.getRowCount(); i++)
            {
                String lgn = (String)usersTable.getValueAt(i, 1);
                if(lgn.equals(login))
                {
                    JOptionPane.showMessageDialog(addUserDialog,
"Пользователь с таким логином уже существует", "Ошибка",
JOptionPane.ERROR_MESSAGE);
                    Client.nullPwds(pwd1, pwd2);
                    return;
                }
            }
            salt = PwdFunctions.getSalt();
            hash = PwdFunctions.hashPwd(pwd1, salt);
            Client.nullPwds(pwd1, pwd2);
            String stringrole =
roleComboBox.getSelectedItem().toString();
            switch(stringrole)
            {
                case "Обучающийся":
                    role = "S";
                    break;
                case "Учитель":
                    role = "T";
                    break;
                case "Администратор":
                    role = "A";
                    break;
            }
            Object[] row = new Object[]{null, login, hash,
PwdFunctions.toHex(salt), name, sname, pname, stringrole};
            ((DefaultTableModel)usersTable.getModel()).addRow(row);
            row[7] = role;
```

```
            insertions.add(new TableRow(row));
        }
        catch (NoSuchAlgorithmException | InvalidKeySpecException
ex)
        {
            logger.log(Level.SEVERE, "Ошибка при создании записи о
пользователе: ", ex);
        }
        finally
        {
            pwdField1.setText("");
            pwdField2.setText("");
            nTextField.setText("");
            sTextField.setText("");
            pTextField.setText("");
            lTextField.setText("");
            addUserDialog.setVisible(false);
        }
    }

    private void
changePwdMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        if(usersTable.getSelectedRow() == -1)
        {
            JOptionPane.showMessageDialog(usersFrame, "Не выбран
пользователь", "Ошибка", JOptionPane.WARNING_MESSAGE);
            return;
        }
        SwingUtilities.invokeLater(() -> {
            changePwdDialog.pack();
            changePwdDialog.setLocationRelativeTo(this);
            changePwdDialog.setVisible(true);
        });
    }

    private void
changePwdConfirmButtonMouseClicked(java.awt.event.MouseEvent evt) {
        char[] pwd1 = changePwdField1.getPassword();
        char[] pwd2 = changePwdField2.getPassword();
        int row = usersTable.getSelectedRow();
        Object key = usersTable.getValueAt(row, 0);
        if(pwd1.length == 0 || pwd2.length == 0)
        {
            JOptionPane.showMessageDialog(changePwdDialog, "Поле
пароля пустое", "Ошибка", JOptionPane.ERROR_MESSAGE);
            Client.nullPwds(pwd1, pwd2);
            return;
        }
        if(!PwdFunctions.equals(pwd1, pwd2))
        {
            JOptionPane.showMessageDialog(changePwdDialog, "Пароли
не совпадают", "Ошибка", JOptionPane.ERROR_MESSAGE);
            Client.nullPwds(pwd1, pwd2);
```

```
            return;
        }
        try
        {
            byte[] salt = PwdFunctions.getSalt();
            String saltstr = PwdFunctions.toHex(salt);
            String hash = PwdFunctions.hashPwd(pwd1, salt);
            Client.nullPwds(pwd1, pwd2);
            if(key == null)
            {
                //Достаём логин
                String login = (String)usersTable.getValueAt(row,
1);
                insertions.removeIf(e -> Objects.equals(e.elems[1],
login));
                Object[] rowarr = new Object[]{
                    (int)usersTable.getValueAt(row, 0), login,
                    hash,
                    saltstr,
                    (String)usersTable.getValueAt(row, 4),
                    (String)usersTable.getValueAt(row, 5),
                    (String)usersTable.getValueAt(row, 6),
                    (String)usersTable.getValueAt(row, 6)};
                insertions.add(new TableRow(rowarr));

((DefaultTableModel)usersTable.getModel()).addRow(rowarr);
            }
            else
            {
                try
                {
                    changes.computeIfAbsent(key, k -> new
TreeMap<>()).put("pwd", hash);
                    changes.computeIfAbsent(key, k -> new
TreeMap<>()).put("salt", saltstr);
                }
                catch(RuntimeException ex)
                {
                    logger.log(Level.WARNING, "Ошибка при
формировании изменений: ", ex);
                }
            }
        }
        catch (NoSuchAlgorithmException | InvalidKeySpecException
ex)
        {
            logger.log(Level.SEVERE, "Ошибка при создании записи о
пользователе: ", ex);
        }
        finally
        {
            changePwdDialog.setVisible(false);
        }
```

```
    }

    private void
addUserDialogWindowClosing(java.awt.event.WindowEvent evt) {
            pwdField1.setText("");
            pwdField2.setText("");
            nTextField.setText("");
            sTextField.setText("");
            pTextField.setText("");
            lTextField.setText("");
    }

    private void
changePwdDialogWindowClosing(java.awt.event.WindowEvent evt) {
        changePwdField1.setText("");
        changePwdField2.setText("");
    }

    private void
changeStudentsButtonMouseClicked(java.awt.event.MouseEvent evt) {
        showOpDialog();
        executor.execute(() -> {
            pkt = NetPacket.FormSqlQuery("SELECT user_ID AS 'ИД',
sname AS 'Фамилия', name AS 'Имя', pname AS 'Отчество' , class AS
'Класс' FROM users RIGHT OUTER JOIN (SELECT * FROM student_info) AS
s ON student_ID = user_ID");
            try
            {
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                SwingUtilities.invokeLater(() -> {
                    switch (pkt.getCode()) {
                        case SQL_RESPONSE:
                        {
                            DefaultTableModel t = new
DefaultTableModel()
                            {
                                @Override
                                public boolean isCellEditable(int
row, int column)
                                {
                                    //Можно редактировать только
класс;
                                    //информация о пользователях
изменяется
                                    //в другом меню
                                    return column == 4;
                                }
                            };
                            formTable(t);
                            formColnames();
```

```
t.addTableModelListener((javax.swing.event.TableModelEvent tevt) ->
{
                                    tableChanged(tevt, null);
                                });
                                studentsTable.setModel(t);
                                SwingUtilities.invokeLater(() -> {
                                    studentsFrame.pack();

studentsFrame.setLocationRelativeTo(this);
                                    studentsFrame.setVisible(true);
                                });
                                break;
                            }
                            case ERROR:
                            {
                                int err = (int)pkt.getData()[0];
                                String msg = (String)pkt.getData()[1];
                                JOptionPane.showMessageDialog(this,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                                break;
                            }
                            default:
                            {
                                JOptionPane.showMessageDialog(this,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                                break;
                            }
                        }
                        operationDialog.setVisible(false);
                    });
                }
                catch (Exception ex)
                {
                    logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                    SwingUtilities.invokeLater(() -> {
                        operationDialog.setVisible(false);
                    });
                }
            });
    }

    private void
studentsConfirmButtonMouseClicked(java.awt.event.MouseEvent evt) {
        if(changes.isEmpty() && deletions.isEmpty() &&
insertions.isEmpty())
            return;
        showOpDialog();
        executor.execute(() -> {
            try
            {
```

```
                pkt = NetPacket.FormUpdateTable("student_info",
"student_ID",
                        null, new TableRow(new Object[] {changes,
deletions, insertions}));
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                switch (pkt.getCode()) {
                    case SQL_UPDATE_RESPONSE:
                        break;
                    case ERROR:
                    {
                        int err = (int)pkt.getData()[0];
                        String msg = (String)pkt.getData()[1];
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);

JOptionPane.showMessageDialog(studentsFrame, "MySQL вернул ошибку: "
+ msg + "\nКод: " + err, "Ошибка", JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                    default:
                    {
                        SwingUtilities.invokeLater(() -> {
                            operationDialog.setVisible(false);

JOptionPane.showMessageDialog(studentsFrame, "Неизвестная ошибка",
"Ошибка", JOptionPane.ERROR_MESSAGE);
                        });
                        return;
                    }
                }
                pkt = NetPacket.FormSqlQuery("SELECT user_ID AS
'ИД', sname AS 'Фамилия', name AS 'Имя', pname AS 'Отчество' , class
AS 'Класс' FROM users LEFT OUTER JOIN (SELECT * FROM student_info)
AS s ON student_ID = user_ID WHERE role = 'S'");
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                SwingUtilities.invokeLater(() -> {
                    switch (pkt.getCode()) {
                        case SQL_RESPONSE:
                        {
                           DefaultTableModel t = new
DefaultTableModel()
                            {
                                @Override
                                public boolean isCellEditable(int
row, int column)
                                {
```

```
                                        //Можно редактировать только
класс;
                                        //информация о пользователях
изменяется
                                        //в другом меню
                                        return column == 4;
                                    }
                                };
                                formTable(t);
                                formColnames();

t.addTableModelListener((javax.swing.event.TableModelEvent tevt) ->
{
                                    tableChanged(tevt, null);
                                });
                                studentsTable.setModel(t);
                                break;
                            }
                            case ERROR:
                            {
                                int err = (int)pkt.getData()[0];
                                String msg = (String)pkt.getData()[1];

JOptionPane.showMessageDialog(studentsFrame, "MySQL вернул ошибку: "
+ msg + "\nКод: " + err, "Ошибка", JOptionPane.ERROR_MESSAGE);
                                break;
                            }
                            default:
                            {

JOptionPane.showMessageDialog(studentsFrame, "Неизвестная ошибка",
"Ошибка", JOptionPane.ERROR_MESSAGE);
                                break;
                            }
                        }
                        operationDialog.setVisible(false);
                    });
                }
                catch (IOException ex)
                {
                    logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                    SwingUtilities.invokeLater(() -> {
                        operationDialog.setVisible(false);
                    });
                    JOptionPane.showMessageDialog(studentsFrame, "Ошибка
соединения с сервером: " + ex.getMessage(), "Ошибка",
JOptionPane.ERROR_MESSAGE);
                    System.exit(1);
                }
                catch (Exception ex)
                {
                    logger.log(Level.SEVERE, "Ошибка при записи/чтении
```

```
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
            finally
            {
                reset();
            }
        });
    }

    private void usersFrameWindowClosing(java.awt.event.WindowEvent
evt) {
        WindowClosingAction(usersTable);
    }

    private void
studentsFrameWindowClosing(java.awt.event.WindowEvent evt) {
        WindowClosingAction(studentsTable);
    }

    private void
changeSubjectsButtonMouseClicked(java.awt.event.MouseEvent evt) {
        showOpDialog();
        executor.execute(() -> {
            DefaultTableModel t = new DefaultTableModel()
            {
                @Override
                public boolean isCellEditable(int row, int column)
                {
                    //Можно редактировать только класс и предмет
                    return column == 5 || column == 6;
                }
            };
            try
            {
                executorMethod("SELECT entry_ID AS 'ИД записи',
user_ID AS 'ИД учителя', sname AS 'Фамилия', name AS 'Имя', pname AS
'Отчество' ,"
                    + "class AS 'Класс', subject AS 'Предмет' FROM
users RIGHT OUTER JOIN (SELECT * FROM classes) AS s ON teacher_ID =
user_ID",
                    t,
                    subjectsTable,
                    subjectsFrame,
                    () -> {
                        try
                        {
                            if(pkt.getCode() == Opcode.SQL_RESPONSE)
                            {
                                NetPacket p =
NetPacket.FormSqlQuery("SELECT user_ID, sname, name, pname FROM
```

```
users WHERE role = 'T'");
                                niocl.write(p.Serialize());
                                niocl.read();
                                p =
NetPacket.Deserialize(niocl.peerAppData().array());
                                teachers =
Arrays.stream(p.getData()).skip(1)
                                        .map(e -> new
Pair(((TableRow)e).elems[0],
                                                String.format("%s %s
%s", ((TableRow)e).elems[1], ((TableRow)e).elems[2],
((TableRow)e).elems[3])))

.collect(Collector.of(LinkedList::new, LinkedList::add, (left,
right) -> {left.addAll(right); return left;}));
                                DefaultComboBoxModel cbmdl = new
DefaultComboBoxModel(teachers.stream()
                                        .map(e -> String.format("%s,
ИД %d", e.getValue(), e.getKey()))
                                        .toArray(String[]::new));
                                teacherComboBox.setModel(cbmdl);
                            }
                        }
                        catch (Exception ex)
                        {
                            logger.log(Level.SEVERE, "Ошибка во
вспомогательном методе: ", ex);
                            SwingUtilities.invokeLater(() -> {
                                operationDialog.setVisible(false);
                            });
                        }
                    });
                operationDialog.setVisible(false);
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }

        });
    }

    private void
subjectsConfirmButtonMouseClicked(java.awt.event.MouseEvent evt) {
         if(changes.isEmpty() && deletions.isEmpty() &&
insertions.isEmpty())
            return;
        showOpDialog();
        executor.execute(() -> {
```

```
            try
            {
                pkt = NetPacket.FormUpdateTable("classes",
"entry_ID",
                        "entry_ID", new TableRow(new Object[]
{changes, deletions, insertions}));
                niocl.write(pkt.Serialize());
                niocl.read();
                pkt =
NetPacket.Deserialize(niocl.peerAppData().array());
                if(pkt.getCode() == Opcode.ERROR)
                {
                    int err = (int)pkt.getData()[0];
                    String msg = (String)pkt.getData()[1];
                    JOptionPane.showMessageDialog(subjectsFrame,
"MySQL вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                    return;
                }
                DefaultTableModel t = new DefaultTableModel()
                {
                    @Override
                    public boolean isCellEditable(int row, int
column)
                    {
                        //Можно редактировать только класс и предмет
                        return column == 5 || column == 6;
                    }
                };
                executorMethod("SELECT entry_ID AS 'ИД записи',
user_ID AS 'ИД учителя', sname AS 'Фамилия', name AS 'Имя', pname AS
'Отчество' ,"
                    + "class AS 'Класс', subject AS 'Предмет' FROM
users RIGHT OUTER JOIN (SELECT * FROM classes) AS s ON teacher_ID =
user_ID",
                    t,
                    subjectsTable,
                    subjectsFrame,
                    () -> {
                        try
                        {
                            if(pkt.getCode() == Opcode.SQL_RESPONSE)
                            {
                                NetPacket p =
NetPacket.FormSqlQuery("SELECT user_ID, sname, name, pname FROM
users WHERE role = 'T'");
                                niocl.write(p.Serialize());
                                niocl.read();
                                p =
NetPacket.Deserialize(niocl.peerAppData().array());
                                teachers =
Arrays.stream(p.getData()).skip(1)
```

```
                                        .map(e -> new
Pair(((TableRow)e).elems[0],
                                                String.format("%s %s
%s", ((TableRow)e).elems[1], ((TableRow)e).elems[2],
((TableRow)e).elems[3])))

.collect(Collector.of(LinkedList::new, LinkedList::add, (left,
right) -> {left.addAll(right); return left;}));
                                DefaultComboBoxModel cbmdl = new
DefaultComboBoxModel(teachers.stream()
                                        .map(e -> String.format("%s,
ИД %d", e.getValue(), e.getKey()))
                                        .toArray(String[]::new));
                                teacherComboBox.setModel(cbmdl);
                            }
                        }
                        catch (Exception ex)
                        {
                            logger.log(Level.SEVERE, "Ошибка во
вспомогательном методе: ", ex);
                            SwingUtilities.invokeLater(() -> {
                                operationDialog.setVisible(false);
                            });
                        }
                    });
            }
            catch (IOException ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
                JOptionPane.showMessageDialog(subjectsFrame, "Ошибка
соединения с сервером: " + ex.getMessage(), "Ошибка",
JOptionPane.ERROR_MESSAGE);
                System.exit(1);
            }
            catch (Exception ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи/чтении
сетевого пакета: ", ex);
                SwingUtilities.invokeLater(() -> {
                    operationDialog.setVisible(false);
                });
            }
            finally
            {
                reset();
            }
        });
    }
```

```
    private void
addSubjectButtonMouseClicked(java.awt.event.MouseEvent evt) {
        SwingUtilities.invokeLater(() -> {
            addSubjectDialog.pack();
            addSubjectDialog.setLocationRelativeTo(this);
            addSubjectDialog.setVisible(true);
        });
    }

    private void
subjectsFrameWindowClosing(java.awt.event.WindowEvent evt) {
        WindowClosingAction(subjectsTable);
    }

    private void
addSubjectConfirmButtonMouseClicked(java.awt.event.MouseEvent evt) {
        String name = (String)teacherComboBox.getSelectedItem();
        String subject = subjectTextField.getText();
        String classname = classTextField.getText();
        if(name == null || name.isEmpty() || subject.isEmpty() ||
classname.isEmpty())
        {
            JOptionPane.showMessageDialog(addSubjectDialog, "Одно из
полей пустое", "Ошибка", JOptionPane.ERROR_MESSAGE);
            return;
        }
        String[] teacherdata = name.split("\\s+|,\\s*ИД\\s*");
        int teacherid = Integer.valueOf(teacherdata[3]);
        Object[] row = new Object[]{null,
            teacherid,
            teacherdata[0],
            teacherdata[1],
            teacherdata[2],
            classname,
            subject};
        TableRow trow = new TableRow(new Object[] {
            classname,
            subject,
            teacherid,
            null
        });
        insertions.add(trow);
        ((DefaultTableModel)subjectsTable.getModel()).addRow(row);
        addSubjectDialog.setVisible(false);
    }

    private void
addSubjectDialogWindowClosing(java.awt.event.WindowEvent evt) {
        subjectTextField.setText("");
        classTextField.setText("");
    }

    private void
```

```
subjectsRmMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedRow = subjectsTable.getSelectedRow();
        DefaultTableModel mdl =
(DefaultTableModel)subjectsTable.getModel();
        if(selectedRow != -1)
        {
            Object primary = mdl.getValueAt(selectedRow, 0);
            if(primary == null)
            {
                TableRow trow = new TableRow(new Object[] {
                    mdl.getValueAt(selectedRow, 5),
                    mdl.getValueAt(selectedRow, 6),
                    mdl.getValueAt(selectedRow, 1),
                    null
                });
                insertions.removeIf(e -> e.equals(trow));
            }
            else
            {
                changes.remove(primary);
                deletions.add(primary);
            }
            mdl.removeRow(selectedRow);
        }
    }

    private void tableChanged(javax.swing.event.TableModelEvent evt,
Predicate<? super Object> pred)
    {
        DefaultTableModel mdl = (DefaultTableModel)evt.getSource();
        switch(evt.getType())
        {
            case UPDATE:
            {
                int row = evt.getFirstRow();
                int col = evt.getColumn();
                Object key = mdl.getValueAt(row, 0);
                String colname = mdl.getColumnName(col);
                Object val = mdl.getValueAt(row, col);
                if(key == null)
                {
                    if(pred != null)
                        insertions.removeIf(pred);
                    TableRow trow = TableRow.FromTableModel(mdl,
row, 0, mdl.getColumnCount());
                    insertions.add(trow);
                }
                else
                {
                    try
                    {
                        changes.computeIfAbsent(key, k -> new
TreeMap<>()).put(colnames.get(colname), val);
```

```
                    }
                    catch(RuntimeException ex)
                    {
                        logger.log(Level.WARNING, "Ошибка при
формировании изменений: ", ex);
                    }
                }
                break;
            }
            case INSERT:
                break;
            case DELETE:
                break;
        }
    }

    private void
subjectTableChanged(javax.swing.event.TableModelEvent evt)
    {
        DefaultTableModel mdl = (DefaultTableModel)evt.getSource();
        switch(evt.getType())
        {
            case UPDATE:
            {
                int row = evt.getFirstRow();
                int col = evt.getColumn();
                Object key = mdl.getValueAt(row, 0);
                String colname = mdl.getColumnName(col);
                Object val = mdl.getValueAt(row, col);
                if(key == null)
                {
                    TableRow trow = new TableRow(new Object[] {
                        mdl.getValueAt(row, 5),
                        mdl.getValueAt(row, 6),
                        mdl.getValueAt(row, 1),
                        null
                    });
                    insertions.removeIf(e -> e.equals(trow));
                    insertions.add(trow);
                }
                else
                {
                    try
                    {
                        changes.computeIfAbsent(key, k -> new
TreeMap<>()).put(colnames.get(colname), val);
                    }
                    catch(RuntimeException ex)
                    {
                        logger.log(Level.WARNING, "Ошибка при
формировании изменений: ", ex);
                    }
                }
```

```
                break;
            }
            case INSERT:
                break;
            case DELETE:
                break;
        }
    }

    private void onAddAction()
    {
        addUserDialog.pack();
        addUserDialog.setLocationRelativeTo(this);
        addUserDialog.setVisible(true);
    }

    private void executorMethod(String query, DefaultTableModel t,
JTable tbl, JDialog frame, Runnable aux)
            throws IOException, ClassNotFoundException, Exception
    {
        pkt = NetPacket.FormSqlQuery(query);
        niocl.write(pkt.Serialize());
        niocl.read();
        pkt = NetPacket.Deserialize(niocl.peerAppData().array());
        if(aux != null) aux.run();
        SwingUtilities.invokeLater(() -> {
            switch (pkt.getCode()) {
                case SQL_RESPONSE:
                {
                    formTable(t);
                    formColnames();

t.addTableModelListener((javax.swing.event.TableModelEvent tevt) ->
{
                        subjectTableChanged(tevt);
                    });
                    tbl.setModel(t);
                    SwingUtilities.invokeLater(() -> {
                        frame.pack();
                        frame.setLocationRelativeTo(this);
                        frame.setVisible(true);
                    });
                    break;
                }
                case ERROR:
                {
                    int err = (int)pkt.getData()[0];
                    String msg = (String)pkt.getData()[1];
                    JOptionPane.showMessageDialog(frame, "MySQL
вернул ошибку: " + msg + "\nКод: " + err, "Ошибка",
JOptionPane.ERROR_MESSAGE);
                    break;
                }
```

```
                default:
                {
                    JOptionPane.showMessageDialog(frame,
"Неизвестная ошибка", "Ошибка", JOptionPane.ERROR_MESSAGE);
                    break;
                }
            }
            operationDialog.setVisible(false);
        });
    }

    private void printRoles(DefaultTableModel t)
    {
        for(int i = 0; i < t.getRowCount(); i++)
        {
            String newrole = null;
            char role = t.getValueAt(i, 7).toString().charAt(0);
            switch(role)
            {
                case 'S':
                    newrole = "Обучающийся";
                    break;
                case 'T':
                    newrole = "Учитель";
                    break;
                case 'A':
                    newrole = "Администратор";
                    break;
            }
            t.setValueAt(newrole, i, 7);
        }
    }

    // Variables declaration - do not modify
    private javax.swing.JButton addButton;
    private javax.swing.JMenuItem addMenuItem;
    private javax.swing.JButton addSubjectButton;
    private javax.swing.JButton addSubjectConfirmButton;
    private javax.swing.JDialog addSubjectDialog;
    private javax.swing.JButton addUserConfirmButton;
    private javax.swing.JDialog addUserDialog;
    private javax.swing.JButton changePwdConfirmButton;
    private javax.swing.JDialog changePwdDialog;
    private javax.swing.JPasswordField changePwdField1;
    private javax.swing.JPasswordField changePwdField2;
    private javax.swing.JMenuItem changePwdMenuItem;
    private javax.swing.JButton changeStudentsButton;
    private javax.swing.JButton changeSubjectsButton;
    private javax.swing.JButton changeUsersButton;
    private javax.swing.JTextField classTextField;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel12;
```

```
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JTextField lTextField;
    private javax.swing.JTextField nTextField;
    private javax.swing.JTextField pTextField;
    private javax.swing.JPasswordField pwdField1;
    private javax.swing.JPasswordField pwdField2;
    private javax.swing.JMenuItem rmMenuItem;
    private javax.swing.JComboBox<String> roleComboBox;
    private javax.swing.JTextField sTextField;
    private javax.swing.JButton studentsConfirmButton;
    private javax.swing.JDialog studentsFrame;
    private javax.swing.JTable studentsTable;
    private javax.swing.JTextField subjectTextField;
    private javax.swing.JButton subjectsConfirmButton;
    private javax.swing.JDialog subjectsFrame;
    private javax.swing.JPopupMenu subjectsPopupMenu;
    private javax.swing.JMenuItem subjectsRmMenuItem;
    private javax.swing.JTable subjectsTable;
    private javax.swing.JComboBox<String> teacherComboBox;
    private javax.swing.JButton usersConfirmButton;
    private javax.swing.JDialog usersFrame;
    private javax.swing.JPopupMenu usersPopupMenu;
    private javax.swing.JTable usersTable;
    private org.jdesktop.beansbinding.BindingGroup bindingGroup;
    // End of variables declaration
}
```

Файл «*DbInit.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dbinit;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import my.*;
import java.sql.*;
import java.text.MessageFormat;
```

```
//left for debug
import java.util.Scanner;

/**
 *
 * @author Admin
 */
public class DbInit {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        if(args.length < 2)
        {
            System.out.println("Мало входных аргументов");
            return;
        }
        MessageFormat fmt = new MessageFormat("CREATE DATABASE IF
NOT EXISTS {0};USE {0};CREATE TABLE IF NOT EXISTS users(user_ID
INTEGER UNSIGNED AUTO_INCREMENT NOT NULL,login CHAR(15) NOT NULL,pwd
CHAR(128) NOT NULL,salt CHAR(32) NOT NULL,name VARCHAR(25),sname
VARCHAR(25),pname VARCHAR(25),role ENUM(''S'', ''T'', ''A'') NOT
NULL,PRIMARY KEY(user_ID),UNIQUE(login));CREATE TABLE IF NOT EXISTS
classes(class VARCHAR(5),subject VARCHAR(15) NOT NULL,teacher_ID
INTEGER UNSIGNED NOT NULL,entry_ID INTEGER UNSIGNED AUTO_INCREMENT
NOT NULL,FOREIGN KEY(teacher_ID) REFERENCES users(user_ID) ON UPDATE
CASCADE ON DELETE CASCADE,PRIMARY KEY(entry_ID),CONSTRAINT entry
UNIQUE(class, subject, teacher_ID) );CREATE TABLE IF NOT EXISTS
student_info(student_ID INTEGER UNSIGNED NOT NULL,class
VARCHAR(5),FOREIGN KEY(student_ID) REFERENCES users(user_ID) ON
UPDATE CASCADE ON DELETE CASCADE,PRIMARY KEY(student_ID));CREATE
TABLE IF NOT EXISTS homework(\n" +
"class VARCHAR(5) NOT NULL,subject VARCHAR(15) NOT NULL,hw_date DATE
NOT NULL,homework TEXT,entry_ID INTEGER UNSIGNED AUTO_INCREMENT NOT
NULL,PRIMARY KEY(entry_ID),FOREIGN KEY fk(class, subject) REFERENCES
classes(class, subject) ON UPDATE CASCADE ON DELETE
CASCADE,CONSTRAINT entry UNIQUE(class, subject, hw_date));CREATE
TABLE IF NOT EXISTS marks(student_ID INTEGER UNSIGNED NOT
NULL,subject VARCHAR(15) NOT NULL,m_date DATE NOT NULL,mark
ENUM(''1'', ''2'', ''3'', ''4'', ''5'', ''н''),entry_ID INTEGER
UNSIGNED AUTO_INCREMENT NOT NULL,PRIMARY KEY(entry_ID),FOREIGN
KEY(student_ID) REFERENCES users(user_ID) ON UPDATE CASCADE ON
DELETE CASCADE);\n" +
"CREATE PROCEDURE InsertMark(IN p_sname VARCHAR(25),IN p_name
VARCHAR(25),IN p_pname VARCHAR(25),IN p_subject VARCHAR(15),IN
p_date VARCHAR(10),IN p_mark CHAR(1),IN p_class VARCHAR(5))\n" +
"BEGIN\n" +
"DECLARE stID INTEGER UNSIGNED;\n" +
"DECLARE varmark CHAR(1);\n" +
"DECLARE EXIT HANDLER FOR SQLEXCEPTION\n" +
"BEGIN\n" +
```

```
"ROLLBACK;\n" +
"RESIGNAL;\n" +
"END;\n" +
"START TRANSACTION;\n" +
"SELECT user_ID FROM users INNER JOIN (SELECT student_ID FROM
student_info WHERE class = p_class) AS s ON s.student_ID =
users.user_ID WHERE name = p_name AND pname = p_pname AND sname =
p_sname INTO stID;\n" +
"IF(stID IS NOT NULL) THEN\n" +
"SELECT NULLIF(p_mark, ''-'') INTO varmark;\n" +
"INSERT INTO marks VALUES (stID, p_subject, STR_TO_DATE(p_date,
''%d.%m.%Y''), varmark, NULL);\n" +
"END IF;\n" +
"COMMIT;\n" +
"END;\n" +
"CREATE USER IF NOT EXISTS ''observer''@''localhost'' IDENTIFIED
WITH caching_sha2_password BY ''eKZrt55NCgvPTUtbQ'' PASSWORD EXPIRE
NEVER;\n" +
"GRANT SELECT ON {0}.users TO ''observer''@''localhost'';\n" +
"CREATE USER IF NOT EXISTS ''student''@''localhost'' IDENTIFIED WITH
caching_sha2_password BY ''fg9tzCcET9shMFEtg'' PASSWORD EXPIRE
NEVER;\n" +
"GRANT SELECT ON {0}.users TO ''student''@''localhost'';\n" +
"GRANT UPDATE (pwd, salt) ON {0}.users TO
''student''@''localhost'';\n" +
"GRANT SELECT ON {0}.student_info TO ''student''@''localhost'';\n" +
"GRANT SELECT ON {0}.marks TO ''student''@''localhost'';\n" +
"GRANT SELECT ON {0}.homework TO ''student''@''localhost'';\n" +
"GRANT SELECT ON {0}.classes TO ''student''@''localhost'';\n" +
"CREATE USER IF NOT EXISTS ''teacher''@''localhost'' IDENTIFIED WITH
caching_sha2_password BY ''tNJFURkgDJQfbNxKL'' PASSWORD EXPIRE
NEVER;\n" +
"GRANT SELECT ON {0}.users TO ''teacher''@''localhost'';\n" +
"GRANT UPDATE (pwd, salt) ON {0}.users TO
''teacher''@''localhost'';\n" +
"GRANT SELECT ON {0}.classes TO ''teacher''@''localhost'';\n" +
"GRANT SELECT ON {0}.student_info TO ''teacher''@''localhost'';\n" +
"GRANT SELECT, UPDATE, INSERT, DELETE ON {0}.homework TO
''teacher''@''localhost'';\n" +
"GRANT SELECT, UPDATE, INSERT, DELETE ON {0}.marks TO
''teacher''@''localhost'';\n" +
"GRANT EXECUTE ON {0}.* TO ''teacher''@''localhost'';\n" +
"CREATE USER IF NOT EXISTS ''admin''@''localhost'' IDENTIFIED WITH
caching_sha2_password BY ''kQ5E3QUyWcVG68wjc'' PASSWORD EXPIRE
NEVER;\n" +
"GRANT ALL ON {0} TO ''admin''@''localhost'';\n" +
"GRANT ALL ON {0}.* TO ''admin''@''localhost'';");
        String query = fmt.format(args);
        Connection con = null;
        System.out.println("Введите пароль пользователя root: ");
        char[] pwd = System.console().readPassword();
        //for debug
        //String pwd = new Scanner(System.in).nextLine();
```

```
        Statement stmt = null;
        PreparedStatement pstmt = null;
        try
        {
            con =
DriverManager.getConnection("jdbc:mysql://localhost:" + args[1] +
"?allowMultiQueries=true", "root", new String(pwd));
            con.setAutoCommit(false);
            stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
            stmt.execute(query);
            pstmt = con.prepareStatement("INSERT INTO users
VALUES(NULL, ?, ?, ?, ?, ?, ?, ?);");
            pstmt.setString(1, "Admin");
            pstmt.setString(4, "Admin");
            pstmt.setNull(5, java.sql.Types.VARCHAR);
            pstmt.setNull(6, java.sql.Types.VARCHAR);
            pstmt.setString(7, "A");
            byte[] salt = PwdFunctions.getSalt();
            String hash =
PwdFunctions.hashPwd("Admin".toCharArray(), salt);
            pstmt.setString(2, hash);
            pstmt.setString(3, PwdFunctions.toHex(salt));
            pstmt.executeUpdate();
            con.commit();
            System.out.println("База данных успешно создана");
        }
        catch(InvalidKeySpecException | NoSuchAlgorithmException |
SQLException ex)
        {
            try
            {
                if(stmt != null)
                    stmt.close();
                if(pstmt != null)
                    pstmt.close();
                if(con != null)
                    con.rollback();
            }
            catch (SQLException ex1) {
                ex1.printStackTrace(System.out);
            }
            ex.printStackTrace(System.out);
        }
        finally
        {
            try {
                if(con != null)
                    con.close();
            }
            catch (SQLException ex)
            {
```

```
                ex.printStackTrace(System.out);
            }
        }
    }
}
```

Файл «*MyFormatter.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package my;

/**
 *
 * @author Admin
 */
import java.util.Date;
import java.util.logging.Formatter;
import java.util.logging.LogRecord;

public class MyFormatter extends Formatter {

    @Override
    public String format(LogRecord record) {
        String fmt = "[%1$tc] %2$s::%3$s%n%4$s: %5$s%n%6$s%n";
        return String.format(fmt, new Date(record.getMillis()),
                record.getSourceClassName(),
record.getSourceMethodName(),
                record.getLevel(), record.getMessage(),
record.getThrown());
    }

}
```

Файл «*NetPacket.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package my;
import java.io.*;
import java.util.*;
import java.sql.*;

/**
 *
 * @author Admin
 */
```

```
public class NetPacket extends SerBase {
    // Код операции
    public enum Opcode
    {
        ERROR,
        FETCH_PASSWORD,
        FETCH_PASSWORD_RESPONSE,
        LOGIN_SUCCESS,
        LOGIN_RESPONSE,
        DISCONNECT_REQUEST,
        SQL_QUERY,
        SQL_RESPONSE,
        SQL_UPDATE_TABLE,
        SQL_UPDATE_BATCH,
        SQL_UPDATE_RESPONSE
    };
    private Opcode code;
    private Object[] data;

    private NetPacket(){}

    public static NetPacket FormFetchPassword(String login) throws
Exception
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.FETCH_PASSWORD;
        ret.FormData(login);
        if(ret.data == null)
            throw new Exception("NetPacket data is null");
        return ret;
    }

    public static NetPacket FormDefaultError()
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.ERROR;
        return ret;
    }

    public static NetPacket FormError(SQLException ex)
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.ERROR;
        ret.FormData(ex.getErrorCode(), ex.getMessage());
        return ret;
    }

    public static NetPacket FormFetchPasswordResponse(ResultSet rs)
throws SQLException
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.FETCH_PASSWORD_RESPONSE;
        ret.FormData(rs.getString(1), rs.getString(2));
```

```
        return ret;
    }

    public static NetPacket FormLoginSuccess(String login) throws
Exception
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.LOGIN_SUCCESS;
        ret.FormData(login);
        if(ret.data == null)
            throw new Exception("NetPacket data is null");
        return ret;
    }

    public static NetPacket FormLoginResponse(ResultSet rs) throws
SQLException
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.LOGIN_RESPONSE;
        ret.FormData(rs.getString(1).charAt(0), rs.getString(2),
rs.getString(3), rs.getString(4), rs.getInt(5));
        return ret;
    }

    public static NetPacket FormSqlQuery(String query,
Object...args)
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.SQL_QUERY;
        LinkedList<Object> L = new
LinkedList<>(Arrays.asList(args));
        L.addFirst(query);
        ret.FormData(L.toArray());
        return ret;
    }

    public static NetPacket FormUpdateTable(String tablename, String
updateKey, String deleteKey, TableRow args)
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.SQL_UPDATE_TABLE;
        ret.FormData(tablename, updateKey, deleteKey, args);
        return ret;
    }

    public static NetPacket FormUpdateBatch(String query, TableRow[]
args)
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.SQL_UPDATE_BATCH;
        ret.FormData(query, args);
        return ret;
    }
```

```
    public static NetPacket FormSqlUpdateResponse()
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.SQL_UPDATE_RESPONSE;
        return ret;
    }

    public static NetPacket FormSqlResponse(ResultSet rs) throws
SQLException
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.SQL_RESPONSE;
        ResultSetMetaData rsmd;
        int cols;
        List<Object> L = new LinkedList<>();
        rsmd = rs.getMetaData();
        cols = rsmd.getColumnCount();
        String[] collabels = new String[cols], colnames = new
String[cols];
        int[] coltypes = new int[cols];
        for(int i = 1; i <= cols; i++)
        {
            colnames[i - 1] = rsmd.getColumnName(i);
            collabels[i - 1] = rsmd.getColumnLabel(i);
            coltypes[i - 1] = rsmd.getColumnType(i);
        }
        L.add(new ResMetaData(collabels, colnames, coltypes));
        while(rs.next())
        {
            Object[] arr = new Object[cols];
            for(int i = 1; i <= cols; i++)
            {
                arr[i - 1] = rs.getObject(i);
            }
            L.add(new TableRow(arr));
        }
        ret.data = L.toArray();
        return ret;
    }

    public static NetPacket FormDisconnectRequest()
    {
        NetPacket ret = new NetPacket();
        ret.code = Opcode.DISCONNECT_REQUEST;
        return ret;
    }

    public byte[] Serialize() throws IOException
    {
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        ObjectOutputStream out = new ObjectOutputStream(bos);
        out.writeObject(this);
        out.flush();
```

```
        return bos.toByteArray();
    }

    public static NetPacket Deserialize(byte[] in) throws
IOException, ClassNotFoundException
    {
        ByteArrayInputStream bis = new ByteArrayInputStream(in);
        ObjectInputStream ois = new ObjectInputStream(bis);
        return (NetPacket)ois.readObject();
    }

    private void FormData(Object arg)
    {
        data = new Object[] {arg};
    }

    private void FormData(Object...args)
    {
        data = args;
    }
    public Opcode getCode()
    {
        return code;
    }
    public Object[] getData()
    {
        return data;
    }
}
```

Файл «*NioSslPeer.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package my;
import java.nio.*;
import java.io.*;
import javax.net.ssl.*;
import java.util.concurrent.*;
import java.nio.channels.*;
import javax.net.ssl.SSLEngineResult.*;
import java.security.*;

/**
 *
 * @author Admin
 */
public abstract class NioSslPeer {
    protected ByteBuffer myAppData;
    protected ByteBuffer myNetData;
```

```
    protected ByteBuffer peerAppData;
    protected ByteBuffer peerNetData;
    /**
     * Will be used to execute tasks that may emerge during
handshake in parallel with the server's main thread.
     */
    protected ExecutorService executor =
Executors.newSingleThreadExecutor();
    protected abstract void write(SocketChannel socketChannel,
SSLEngine engine, byte[] message) throws Exception;
    protected boolean doHandshake(SocketChannel socketChannel,
SSLEngine engine) throws IOException {

        SSLEngineResult result;
        HandshakeStatus handshakeStatus;

        // NioSslPeer's fields myAppData and peerAppData are
supposed to be large enough to hold all message data the peer
        // will send and expects to receive from the other peer
respectively. Since the messages to be exchanged will usually be
less
        // than 16KB long the capacity of these fields should also
be smaller. Here we initialize these two local buffers
        // to be used for the handshake, while keeping client's
buffers at the same size.
        int appBufferSize =
engine.getSession().getApplicationBufferSize();
        ByteBuffer lmyAppData = ByteBuffer.allocate(appBufferSize);
        ByteBuffer lpeerAppData =
ByteBuffer.allocate(appBufferSize);
        myNetData.clear();
        peerNetData.clear();
        handshakeStatus = engine.getHandshakeStatus();
        while (handshakeStatus !=
SSLEngineResult.HandshakeStatus.FINISHED && handshakeStatus !=
SSLEngineResult.HandshakeStatus.NOT_HANDSHAKING) {
            switch (handshakeStatus) {
            case NEED_UNWRAP:
                if (socketChannel.read(peerNetData) < 0) {
                    if (engine.isInboundDone() &&
engine.isOutboundDone()) {
                        return false;
                    }
                    try {
                        engine.closeInbound();
                    } catch (SSLException e) {
                        e.printStackTrace();
                    }
                    engine.closeOutbound();
                    // After closeOutbound the engine will be set to
WRAP state, in order to try to send a close message to the client.
                    handshakeStatus = engine.getHandshakeStatus();
                    break;
```

```
                }
                peerNetData.flip();
                try {
                    result = engine.unwrap(peerNetData,
lpeerAppData);
                    peerNetData.compact();
                    handshakeStatus = result.getHandshakeStatus();
                } catch (SSLException sslException) {
                    sslException.printStackTrace();
                    engine.closeOutbound();
                    handshakeStatus = engine.getHandshakeStatus();
                    break;
                }
                switch (result.getStatus()) {
                case OK:
                    break;
                case BUFFER_OVERFLOW:
                    // Will occur when lpeerAppData's capacity is
smaller than the data derived from peerNetData's unwrap.
                    lpeerAppData = enlargeApplicationBuffer(engine,
lpeerAppData);
                    break;
                case BUFFER_UNDERFLOW:
                    // Will occur either when no data was read from
the peer or when the peerNetData buffer was too small to hold all
peer's data.
                    peerNetData = handleBufferUnderflow(engine,
peerNetData);
                    break;
                case CLOSED:
                    if (engine.isOutboundDone()) {
                        return false;
                    } else {
                        engine.closeOutbound();
                        handshakeStatus =
engine.getHandshakeStatus();
                        break;
                    }
                default:
                    throw new IllegalStateException("Invalid SSL
status: " + result.getStatus());
                }
                break;
            case NEED_WRAP:
                myNetData.clear();
                try {
                    result = engine.wrap(lmyAppData, myNetData);
                    handshakeStatus = result.getHandshakeStatus();
                } catch (SSLException sslException) {
                    sslException.printStackTrace();
                    engine.closeOutbound();
                    handshakeStatus = engine.getHandshakeStatus();
                    break;
```

```
                }
                switch (result.getStatus()) {
                case OK:
                    myNetData.flip();
                    while (myNetData.hasRemaining()) {
                        socketChannel.write(myNetData);
                    }
                    break;
                case BUFFER_OVERFLOW:
                    // Will occur if there is not enough space in
myNetData buffer to write all the data that would be generated by
the method wrap.
                    // Since myNetData is set to session's packet
size we should not get to this point because SSLEngine is supposed
                    // to produce messages smaller or equal to that,
but a general handling would be the following:
                    myNetData = enlargePacketBuffer(engine,
myNetData);
                    break;
                case BUFFER_UNDERFLOW:
                    throw new SSLException("Buffer underflow occured
after a wrap. I don't think we should ever get here.");
                case CLOSED:
                    try {
                        myNetData.flip();
                        while (myNetData.hasRemaining()) {
                            socketChannel.write(myNetData);
                        }
                        // At this point the handshake status will
probably be NEED_UNWRAP so we make sure that peerNetData is clear to
read.
                        peerNetData.clear();
                    } catch (IOException e) {
                        e.printStackTrace();
                        handshakeStatus =
engine.getHandshakeStatus();
                    }
                    break;
                default:
                    throw new IllegalStateException("Invalid SSL
status: " + result.getStatus());
                }
                break;
            case NEED_TASK:
                Runnable task;
                while ((task = engine.getDelegatedTask()) != null) {
                    executor.execute(task);
                }
                handshakeStatus = engine.getHandshakeStatus();
                break;
            case FINISHED:
                break;
```

```
            default:
                throw new IllegalStateException("Invalid SSL status:
" + handshakeStatus);
            }
        }

        return true;

    }

    protected ByteBuffer enlargeBuffer(ByteBuffer buffer, int
sessionProposedCapacity) {
        if (sessionProposedCapacity > buffer.capacity()) {
            buffer = ByteBuffer.allocate(sessionProposedCapacity);
        } else {
            buffer = ByteBuffer.allocate(buffer.capacity() * 2);
        }
        return buffer;
    }

    protected ByteBuffer enlargeApplicationBuffer(SSLEngine engine,
ByteBuffer buffer) {
        return enlargeBuffer(buffer,
engine.getSession().getApplicationBufferSize());
    }

    protected ByteBuffer handleBufferUnderflow(SSLEngine engine,
ByteBuffer buffer) {
        if (engine.getSession().getPacketBufferSize() <
buffer.limit()) {
            return buffer;
        } else {
            ByteBuffer replaceBuffer = enlargePacketBuffer(engine,
buffer);
            buffer.flip();
            replaceBuffer.put(buffer);
            return replaceBuffer;
        }
    }

    protected ByteBuffer enlargePacketBuffer(SSLEngine engine,
ByteBuffer buffer) {
        return enlargeBuffer(buffer,
engine.getSession().getPacketBufferSize());
    }

    protected void closeConnection(SocketChannel socketChannel,
SSLEngine engine) throws IOException  {
        engine.closeOutbound();
        doHandshake(socketChannel, engine);
        socketChannel.close();
    }
```

```
    protected void handleEndOfStream(SocketChannel socketChannel,
SSLEngine engine) throws IOException  {
        try {
            engine.closeInbound();
        } catch (SSLException e) {
            e.printStackTrace();
        }
        closeConnection(socketChannel, engine);
    }

    protected KeyManager[] createKeyManagers(String filepath, String
keystorePassword, String keyPassword) throws Exception {
        KeyStore keyStore = KeyStore.getInstance("JKS");
        try (InputStream keyStoreIS = new FileInputStream(filepath))
{
            keyStore.load(keyStoreIS,
keystorePassword.toCharArray());
        }
        KeyManagerFactory kmf =
KeyManagerFactory.getInstance("SunX509");
        kmf.init(keyStore, keyPassword.toCharArray());
        return kmf.getKeyManagers();
    }

    protected TrustManager[] createTrustManagers(String filepath,
String keystorePassword) throws Exception {
        KeyStore trustStore = KeyStore.getInstance("JKS");
        try (InputStream trustStoreIS = new
FileInputStream(filepath)) {
            trustStore.load(trustStoreIS,
keystorePassword.toCharArray());
        }
        TrustManagerFactory trustFactory =
TrustManagerFactory.getInstance("SunX509");
        trustFactory.init(trustStore);
        return trustFactory.getTrustManagers();
    }

}
```

Файл «*PwdFunctions.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package my;

import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;
```

```
/**
 *
 * @author Admin
 */
public final class PwdFunctions {
    public static final int ITERATIONS = 1000;
    public static final int KEY_SIZE = 512;
    private static final char[] HEX_ARRAY =
"0123456789ABCDEF".toCharArray();
    private PwdFunctions(){};
//    public static String hashPwd(char[] password, final String
salt) throws NoSuchAlgorithmException, InvalidKeySpecException
//    {
//        return hashPwd(password, fromHex(salt));
//    }
    public static String hashPwd(char[] password, final byte[] salt)
throws NoSuchAlgorithmException, InvalidKeySpecException
    {
        PBEKeySpec spec = new PBEKeySpec(password, salt, ITERATIONS,
KEY_SIZE);
        SecretKeyFactory skf =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
        byte[] hash = skf.generateSecret(spec).getEncoded();
        spec.clearPassword();
        return toHex(hash);
    }
    public static boolean validatePassword(char[] origPwd, String
storedPwd, String salt) throws NoSuchAlgorithmException,
InvalidKeySpecException
    {
        PBEKeySpec spec = new PBEKeySpec(origPwd, fromHex(salt),
ITERATIONS, KEY_SIZE);
        SecretKeyFactory skf =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
        byte[] hash = skf.generateSecret(spec).getEncoded(),
            storedHash = fromHex(storedPwd);
        spec.clearPassword();
        int diff = hash.length ^ storedHash.length;
        for(int i = 0; i < hash.length && i < storedHash.length;
i++)
        {
            diff |= hash[i] ^ storedHash[i];
        }
        return diff == 0;
    }

    public static byte[] getSalt() throws NoSuchAlgorithmException
    {
        SecureRandom sr = new SecureRandom();
        byte[] salt = new byte[16];
        sr.nextBytes(salt);
        return salt;
    }
```

```
    private static byte[] fromHex(String hex)
    {
        int len = hex.length();
        byte[] data = new byte[len / 2];
        for (int i = 0; i < len; i += 2) {
            data[i / 2] = (byte) ((Character.digit(hex.charAt(i),
16) << 4)
                                 + Character.digit(hex.charAt(i+1),
16));
        }
        return data;
    }

    public static String toHex(byte[] array)
    {
        char[] hexChars = new char[array.length * 2];
        for (int j = 0; j < array.length; j++) {
            int v = array[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }

    public static boolean equals(char[] pwd1, char[] pwd2)
    {
        int diff = pwd1.length ^ pwd2.length;
        for(int i = 0; i < pwd1.length && i < pwd2.length; i++)
        {
            diff |= pwd1[i] ^ pwd2[i];
        }
        return diff == 0;
    }
}
```

Файл «*ResMetaData.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package my;

/**
 *
 * @author Admin
 */
public class ResMetaData extends SerBase {
    public final String[] colLabels;
    public final String[] colNames;
    public final int[] coLtypes;
    public ResMetaData(String[] collabels, String[] colnames, int[]
```

```
typesarr)
    {
        colLabels = collabels;
        colNames = colnames;
        coLtypes = typesarr;
    }
}
```

Файл «*SerBase.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package my;
import java.io.*;

/**
 *
 * @author Admin
 */
public abstract class SerBase implements Serializable {
    private static final long serialVersionUID =
0x2415F3DE383B315FL;
    //private static final long serialVersionUID =
0x7D6C63085EB973AFL;
}
```

Файл «*TableRow.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package my;
import java.util.Arrays;
import javax.swing.table.TableModel;

/**
 *
 * @author Admin
 */
public class TableRow extends SerBase {
    public final Object[] elems;
    public TableRow(Object[] arr)
            throws NullPointerException
    {
        if(arr == null)
            throw new NullPointerException();
        elems = arr;
    }
```

```
    public static TableRow FromTableModel(TableModel tbl, int row,
int colstart, int colnum, Object...additional)
            throws NullPointerException, IllegalArgumentException
    {
        if(tbl == null)
            throw new NullPointerException();
        if(row < 0 || colstart >= colnum)
            throw new IllegalArgumentException();
        Object[] arr = new Object[colnum - colstart +
additional.length];
        for(int i = 0; colstart < colnum; colstart++, i++)
        {
            arr[i] = tbl.getValueAt(row, colstart);
        }
        colstart--;
        for(int i = 0; i < additional.length; colstart++, i++)
        {
            arr[colstart] = additional[i];
        }
        return new TableRow(arr);
    }

    @Override
    public boolean equals(Object obj)
    {
        if(!(obj instanceof TableRow))
            return false;
        return Arrays.deepEquals(elems, ((TableRow)obj).elems);
    }

    @Override
    public int hashCode() {
        int hash = 5;
        hash = 13 * hash + Arrays.deepHashCode(this.elems);
        return hash;
    }
}
```

Файл «*NioSslServer.java*»

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package server;
import my.NioSslPeer;
import java.nio.channels.*;
import java.io.*;
import java.nio.*;
import java.net.*;
import java.security.*;
import java.sql.*;
```

```
import java.util.*;
import javax.net.ssl.*;
import java.nio.channels.spi.*;
import java.util.concurrent.*;
import java.util.logging.Level;
import java.util.stream.Stream;
import my.*;
import static server.Server.logger;

/**
 *
 * @author Admin
 */
public class NioSslServer extends NioSslPeer {

    private final Map<Channel, Connection> conmap;
    private final String dbname;
    private final int max_connections;
    private final int dbport;
    private boolean active;
    private final Connection con;
    private final SSLContext context;
    private final Selector selector;
    private final ExecutorService service;
    private class Worker implements Runnable {

        private final byte[] buf;
        private NetPacket pkt;
        private ResultSet res;
        private final SocketChannel channel;
        private final SSLEngine eng;
        private PreparedStatement pstmt;

        public Worker(ByteBuffer peerData, SocketChannel sc,
SSLEngine engine)
        {
            buf = new byte[peerData.limit()];
            System.arraycopy(peerData.array(), 0, buf, 0,
buf.length);
            pkt = null;
            res = null;
            pstmt = null;
            channel = sc;
            eng = engine;
            peerData.clear();
        }

        private boolean FetchPassword() throws SQLException
        {
            String login = (String)pkt.getData()[0];
            pstmt = QueryDB(con, "SELECT pwd, salt FROM users WHERE
login = ?", login);
            return res.next();
```

```
        }

        private void FetchUserData() throws SQLException
        {
            String login = (String)pkt.getData()[0];
            pstmt = QueryDB(con, "SELECT role, name, sname, pname,
user_ID FROM users WHERE login = ?", login);
            //Проверка не нужна: подразумевается,
            //что эта функция будет вызвана после FetchPassword
            res.next();
        }

        private int[] UpdateTable(Connection c, String tablename,
String updateKey, Map<Object, Map<String, Object>> changes) throws
SQLException
        {
            LinkedList<Integer> L = new LinkedList<>();
            changes.forEach((row, map) -> {
                map.entrySet().stream()
                    .filter(e -> !e.getKey().contains(";"))
                    .forEach(e -> {
                        String updateQuery = String.format("UPDATE
%s SET %s = ? WHERE %s = ?", tablename, e.getKey(), updateKey);
                        PreparedStatement stmt = null;
                        Object val = e.getValue();
                        try
                        {
                            stmt = c.prepareStatement(updateQuery,
ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);
                            if(val instanceof String &&
((String)val).isEmpty())
                                stmt.setNull(1,
java.sql.Types.VARCHAR);
                            else
                                stmt.setObject(1, val);
                            stmt.setObject(2, row);
                            L.add(stmt.executeUpdate());
                        }
                        catch (SQLException ex)
                        {
                            logger.log(Level.SEVERE,
String.format("Ошибка при добавлении изменений, код: %d",
ex.getErrorCode()), ex);
                        }
                        finally
                        {
                            try
                            {
                                if(stmt != null)
                                    stmt.close();
                            }
                            catch (SQLException ex)
                            {
```

```
                                    logger.log(Level.SEVERE,
String.format("Ошибка при работе с базой данных, код: %d",
ex.getErrorCode()), ex);
                                }
                            }
                        });
            });
            return L.stream().mapToInt(i -> i).toArray();
        }

        private int[] DeleteFromTable(Connection c, String
tablename, String deleteKey, Set<Object> deletions) throws
SQLException
        {
            int[] ret;
            String deleteQuery = String.format("DELETE FROM %s WHERE
%s = ?", tablename, deleteKey);
            try (PreparedStatement stmt =
c.prepareStatement(deleteQuery, ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY))
            {
                deletions.stream().forEach(e -> {
                    try
                    {
                        stmt.setObject(1, e);
                        stmt.addBatch();
                    }
                    catch (SQLException ex)
                    {
                        logger.log(Level.SEVERE,
String.format("Ошибка при удалении записей, код: %d",
ex.getErrorCode()), ex);
                    }
                });
                ret = stmt.executeBatch();
            }
            return ret;
        }

        private int[] InsertIntoTable(Connection c, String
tablename, Set<TableRow> insertions) throws SQLException
        {
            int[] ret;
            String insertQuery = String.format("INSERT IGNORE INTO
%s VALUES (", tablename);
            final int width =
insertions.iterator().next().elems.length;
            for(int i = 1; i < width; i++)
            {
                insertQuery += "?, ";
            }
            insertQuery += "?)";
            try (PreparedStatement stmt =
```

```
c.prepareStatement(insertQuery, ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY))
            {
                Iterator<TableRow> it = insertions.iterator();
                while(it.hasNext())
                {
                    TableRow row = it.next();
                    for(int i = 1; i <= width; i++)
                    {
                        Object e = row.elems[i - 1];
                        if(e == null)
                            stmt.setNull(i, java.sql.Types.INTEGER);
                        else if(e instanceof String &&
((String)e).isEmpty())
                            stmt.setNull(i, java.sql.Types.VARCHAR);
                        else
                            stmt.setObject(i, e);
                    }
                    stmt.addBatch();
                }
                ret = stmt.executeBatch();
            }
            return ret;
        }

        private int[][] ModifyTable(Connection c, String tablename,
String updateKey, String deleteKey, TableRow args) throws
QueryFormatException, SQLException
        {
            int[][] ret = null;
            Map<Object, Map<String, Object>> changes = (Map<Object,
Map<String, Object>>)args.elems[0];
            Set<Object> deletions = (Set<Object>)args.elems[1];
            Set<TableRow> insertions = (Set<TableRow>)args.elems[2];
            if(tablename != null && tablename.contains(";") ||
                    updateKey != null && updateKey.contains(";") ||
                    deleteKey != null && deleteKey.contains(";"))
            {
                throw new QueryFormatException("Неверный формат
запроса");
            }
            c.setAutoCommit(false);
            try
            {
                ret = new int[3][];
                if(changes != null) ret[0] = changes.isEmpty() ?
null : UpdateTable(c, tablename, updateKey, changes);
                if(deletions != null) ret[1] = deletions.isEmpty() ?
null : DeleteFromTable(c, tablename, deleteKey, deletions);
                if(insertions != null) ret[2] = insertions.isEmpty()
? null : InsertIntoTable(c, tablename, insertions);
                c.commit();
            }
```

```
            catch (SQLException ex)
            {
                logger.log(Level.SEVERE, String.format("Ошибка при
модификации таблицы, код: %d", ex.getErrorCode()), ex);
                c.rollback();
                throw ex;
            }
            finally
            {
                    c.setAutoCommit(true);
            }
            return ret;
        }

        private int[] UpdateBatch(Connection c, String query,
TableRow[] args) throws SQLException
        {
            int[] ret;
            c.setAutoCommit(false);
            try(PreparedStatement stmt = c.prepareStatement(query,
ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE))
            {
                Arrays.stream(args).forEach(e -> {
                    Object[] vals = e.elems;
                    try
                    {
                        for(int i = 1; i <= vals.length; i++)
                        {
                            stmt.setObject(i, vals[i - 1]);
                        }
                         stmt.addBatch();
                    }
                    catch (SQLException ex)
                    {
                        logger.log(Level.SEVERE,
String.format("Ошибка при работе с базой данных, код: %d",
ex.getErrorCode()), ex);
                    }
                });
                ret = stmt.executeBatch();
                c.commit();
            }
            catch(SQLException ex)
            {
                c.rollback();
                throw ex;
            }
            finally
            {
                c.setAutoCommit(true);
            }
            return ret;
```

```
        }

        private PreparedStatement QueryDB(Connection c, String
query, Object...args) throws SQLException
        {
            PreparedStatement stmt = c.prepareStatement(query,
ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
            if(args != null)
            {
                for(int i = 1; i <= args.length; i++)
                {
                    stmt.setObject(i, args[i - 1]);
                }
            }
            res = stmt.executeQuery();
            return stmt;
        }

        //На случай застревания клиента в read
        private void writeDefaultError()
        {
            NetPacket p = NetPacket.FormDefaultError();
            try
            {
                write(channel, eng, p.Serialize());
            }
            catch (IOException ex)
            {
                logger.log(Level.SEVERE, "Ошибка при записи в сокет:
", ex);
            }
        }

        @Override
        public void run()
        {
            try
            {
                pkt = NetPacket.Deserialize(buf);
                switch(pkt.getCode())
                {
                case FETCH_PASSWORD:
                {
                    pkt = FetchPassword() ?
NetPacket.FormFetchPasswordResponse(res) :
                            NetPacket.FormDefaultError();
                    write(channel, eng, pkt.Serialize());
                    break;
                }
                case LOGIN_SUCCESS:
                {
                    FetchUserData();
                    char role = res.getString(1).charAt(0);
```

```
                    Connection dbcon = null;
                    switch(role)
                    {
                        case 'S':
                            dbcon =
DriverManager.getConnection(String.format("jdbc:mysql://localhost:%d
/%s", dbport, dbname), "student", "fg9tzCcET9shMFEtg");
                            break;
                        case 'T':
                            dbcon =
DriverManager.getConnection(String.format("jdbc:mysql://localhost:%d
/%s", dbport, dbname), "teacher", "tNJFURkgDJQfbNxKL");
                            break;
                        case 'A':
                            dbcon =
DriverManager.getConnection(String.format("jdbc:mysql://localhost:%d
/%s", dbport, dbname), "admin", "kQ5E3QUyWcVG68wjc");
                            break;
                    }
                    conmap.put(channel, dbcon);
                    pkt = NetPacket.FormLoginResponse(res);
                    write(channel, eng, pkt.Serialize());
                    break;
                }
                case SQL_QUERY:
                {
                    String query = (String)pkt.getData()[0];
                    Object[] args = null;
                    //Если есть аргументы, переводим их в массив
                    if(pkt.getData().length - 1 != 0)
                        args =
Stream.of(pkt.getData()).skip(1).toArray(Object[]::new);
                    pstmt = QueryDB(conmap.get(channel), query,
args);
                    pkt = NetPacket.FormSqlResponse(res);
                    write(channel, eng, pkt.Serialize());
                    break;
                }
                case SQL_UPDATE_TABLE:
                {
                    String tablename = (String)pkt.getData()[0];
                    String updateKey = (String)pkt.getData()[1];
                    String deleteKey = (String)pkt.getData()[2];
                    TableRow args = (TableRow)pkt.getData()[3];
                    int[][] modres =
ModifyTable(conmap.get(channel), tablename, updateKey, deleteKey,
args);
                    pkt = NetPacket.FormSqlUpdateResponse();
                    write(channel, eng, pkt.Serialize());
                    break;
                }
                case SQL_UPDATE_BATCH:
                {
```

```
                    String query = (String)pkt.getData()[0];
                    TableRow[] args = (TableRow[])pkt.getData()[1];
                    int[] modres = UpdateBatch(conmap.get(channel),
query, args);
                    pkt = NetPacket.FormSqlUpdateResponse();
                    write(channel, eng, pkt.Serialize());
                    break;
                }
                case DISCONNECT_REQUEST:
                {
                    unmap(channel);
                    break;
                }
                }
            }
            catch(SQLException ex)
            {
                logger.log(Level.SEVERE, String.format("Ошибка при
работе с базой данных, код: %d", ex.getErrorCode()), ex);
                pkt = NetPacket.FormError(ex);
                try
                {
                    write(channel, eng, pkt.Serialize());
                }
                catch (IOException ex1)
                {
                    logger.log(Level.SEVERE, "Ошибка при записи в
сокет: ", ex1);
                }
            }
            catch(IOException | ClassNotFoundException ex)
            {
                logger.log(Level.SEVERE, "Ошибка при вводе-выводе
сетевого пакета: ", ex);
                writeDefaultError();
            }
            catch(RuntimeException ex)
            {
                logger.log(Level.WARNING, "Runtime exception: ",
ex);
                writeDefaultError();
            }
            catch (QueryFormatException ex)
            {
                logger.log(Level.SEVERE, "Ошибка при форматировании
запроса. Попытка инъекции?\n", ex);
                writeDefaultError();
            }
            finally
            {
                try
                {
```

```
                    //PreparedStatement закрываем вручную,
                    //т.к. иначе ResultSet закрывается до чтения
данных из него
                    if(pstmt != null)
                        pstmt.close();
                }
                catch (SQLException ex)
                {
                    logger.log(Level.SEVERE, String.format("Ошибка
при закрытии PreparedStatement, код: %d", ex.getErrorCode()), ex);
                }
            }
        }


    }

    public NioSslServer(String protocol, int port, String db, int
dbp, String keystorepwd, String keypwd) throws Exception {
        context = SSLContext.getInstance(protocol);

context.init(super.createKeyManagers("../resources/KeyStore.jks",
keystorepwd, keypwd),

super.createTrustManagers("../resources/KeyStore.jks", keystorepwd),
new SecureRandom());

        SSLSession dummySession =
context.createSSLEngine().getSession();
        myAppData =
ByteBuffer.allocate(dummySession.getApplicationBufferSize());
        myNetData =
ByteBuffer.allocate(dummySession.getPacketBufferSize());
        peerAppData =
ByteBuffer.allocate(dummySession.getApplicationBufferSize());
        peerNetData =
ByteBuffer.allocate(dummySession.getPacketBufferSize());
        dummySession.invalidate();

        selector = SelectorProvider.provider().openSelector();
        ServerSocketChannel serverSocketChannel =
ServerSocketChannel.open()
                .setOption(StandardSocketOptions.SO_REUSEADDR,
true);
        serverSocketChannel.configureBlocking(false);
        serverSocketChannel.socket().bind(new
InetSocketAddress((InetAddress)null, port));
        serverSocketChannel.register(selector,
SelectionKey.OP_ACCEPT);

        int cores = Runtime.getRuntime().availableProcessors();
        service = Executors.newFixedThreadPool(cores - 1);
```

```
        dbname = db;
        dbport = dbp;
        con = DriverManager.getConnection("jdbc:mysql://localhost:"
+ dbp + "/" + db, "observer", "eKZrt55NCgvPTUtbQ");
        PreparedStatement stmt = con.prepareStatement("SELECT
@@max_connections;");
        ResultSet rs = stmt.executeQuery();
        rs.next();
        max_connections = rs.getInt(1);
        conmap = new HashMap<>(max_connections, 1.0f);
        active = true;
    }

    public void start()
    {
        while (isActive()) {
            SelectionKey key = null;
            try
            {
                selector.select();
                Iterator<SelectionKey> selectedKeys =
selector.selectedKeys().iterator();
                while (selectedKeys.hasNext()) {
                    key = selectedKeys.next();
                    selectedKeys.remove();
                    if (!key.isValid()) {
                        continue;
                    }
                    if (key.isAcceptable()) {
                        accept(key);
                    } else if (key.isReadable()) {
                        try
                        {
                            boolean cont = read((SocketChannel)
key.channel(), (SSLEngine) key.attachment());
                            if(cont)
                                service.submit(new
Worker(peerAppData, (SocketChannel) key.channel(), (SSLEngine)
key.attachment()));
                        }
                        catch(IOException ex)
                        {
                            closeConnection((SocketChannel)
key.channel(), (SSLEngine) key.attachment());
                            key.cancel();
                            logger.log(Level.SEVERE, "Ошибка чтения
с сокета: ", ex);
                        }
                    }
                }
            }
            catch(Exception ex)
            {
```

```
                if(key != null)
                {
                    key.cancel();
                    try
                    {
                        if(key.channel() instanceof SocketChannel)
                            unmap((SocketChannel) key.channel());
                    }
                    catch (IOException | SQLException ex1)
                    {
                        logger.log(Level.SEVERE, "Ошибка удаления
записи соединения: ", ex1);
                    }
                }
                logger.log(Level.SEVERE, "Ошибка при приёме
соединений: ", ex);
            }
        }
    }
    public void stop() {
        active = false;
        executor.shutdown();
        selector.wakeup();
        service.shutdown();
    }

     private void accept(SelectionKey key) throws Exception {
        SocketChannel socketChannel = ((ServerSocketChannel)
key.channel()).accept();
        socketChannel.configureBlocking(false);

        SSLEngine engine = context.createSSLEngine();
        engine.setUseClientMode(false);
        engine.beginHandshake();

        if (doHandshake(socketChannel, engine)) {
            socketChannel.register(selector, SelectionKey.OP_READ,
engine);
        } else {
            socketChannel.close();
        }
        String addr =
((InetSocketAddress)socketChannel.getRemoteAddress()).getAddress().g
etHostAddress();
        System.out.println("Принято новое соединение с адреса " +
addr);
        logger.log(Level.INFO, "Принято новое соединение с адреса
{0}", addr);
    }

    protected boolean read(SocketChannel socketChannel, SSLEngine
engine) throws IOException {
        peerNetData.clear();
```

```
        int bytesRead = socketChannel.read(peerNetData);
        if (bytesRead > 0) {
            peerNetData.flip();
            while (peerNetData.hasRemaining()) {
                peerAppData.clear();
                SSLEngineResult result = engine.unwrap(peerNetData,
peerAppData);
                switch (result.getStatus()) {
                case OK:
                    peerAppData.flip();
                    break;
                case BUFFER_OVERFLOW:
                    peerAppData = enlargeApplicationBuffer(engine,
peerAppData);
                    break;
                case BUFFER_UNDERFLOW:
                    peerNetData = handleBufferUnderflow(engine,
peerNetData);
                    break;
                case CLOSED:
                    String addr =
((InetSocketAddress)socketChannel.getRemoteAddress()).getAddress().g
etHostAddress();
                    closeConnection(socketChannel, engine);
                    logger.log(Level.INFO, "Клиент {0} отключился",
addr);
                    return false;
                default:
                    throw new IllegalStateException("Invalid SSL
status: " + result.getStatus());
                }
            }
        } else if (bytesRead < 0) {
            handleEndOfStream(socketChannel, engine);
        }
        return true;
    }

    @Override
    protected synchronized void write(SocketChannel socketChannel,
SSLEngine engine, byte[] message) throws IOException {
        myAppData.clear();
        myAppData.put(message);
        myAppData.flip();
        while (myAppData.hasRemaining()) {
            // The loop has a meaning for (outgoing) messages larger
than 16KB.
            // Every wrap call will remove 16KB from the original
message and send it to the remote peer.
            myNetData.clear();
            SSLEngineResult result = engine.wrap(myAppData,
myNetData);
```

```
            switch (result.getStatus()) {
            case OK:
                myNetData.flip();
                while (myNetData.hasRemaining()) {
                    socketChannel.write(myNetData);
                }
                break;
            case BUFFER_OVERFLOW:
                myNetData = enlargePacketBuffer(engine, myNetData);
                break;
            case BUFFER_UNDERFLOW:
                throw new SSLException("Buffer underflow occured
after a wrap. I don't think we should ever get here.");
            case CLOSED:
                String addr =
((InetSocketAddress)socketChannel.getRemoteAddress()).getAddress().g
etHostAddress();
                closeConnection(socketChannel, engine);
                logger.log(Level.INFO, "Клиент {0} отключился",
addr);
                return;
            default:
                throw new IllegalStateException("Invalid SSL status:
" + result.getStatus());
            }
        }
    }

    private boolean isActive() {
        return active;
    }

    @Override
    protected void closeConnection(SocketChannel socketChannel,
SSLEngine engine) throws IOException  {
        super.closeConnection(socketChannel, engine);
    }

    private void unmap(Channel k) throws SQLException, IOException
    {
        String addr =
((InetSocketAddress)((SocketChannel)k).getRemoteAddress()).getAddres
s().getHostAddress();
        conmap.get(k).close();
        conmap.remove(k);
        logger.log(Level.INFO, "БД-соединение с адреса {0} закрыто",
addr);
    }
}
```

Файл «*QueryFormatException.java*»

```
/*
 * To change this license header, choose License Headers in Project
```

```
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package server;

/**
 *
 * @author Admin
 */
public class QueryFormatException extends Exception {
    public QueryFormatException(String msg)
    {
        super(msg);
    }
}
```

Файл «*Server.java*»

```
package server;

import java.io.*;
import java.util.logging.*;
import my.MyFormatter;

/**
 *
 * @author Admin
 */
public class Server {
    public static final Logger logger =
Logger.getLogger(Logger.GLOBAL_LOGGER_NAME);
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int dbport = 0, port = 3333;
        String dbname = "", keystorepwd = "", keypwd = "";
        logger.setLevel(Level.ALL);
        try
        {
            Handler fileHandler = new FileHandler("serverlog", 200 *
1024 * 1024, 5);
            fileHandler.setFormatter(new MyFormatter());
            logger.addHandler(fileHandler);
        }
        catch(IOException ex)
        {
            ex.printStackTrace(System.out);
            return;
        }
        try(BufferedReader br = new BufferedReader(new
FileReader("../resources/sconfig.cfg"))) {
            for(String line; (line = br.readLine()) != null; ) {
```

```
                String[] kv = line.split("\\s*=\\s*");
                switch(kv[0])
                {
                    case "port":
                        port = Integer.parseInt(kv[1]);
                        break;
                    case "dbport":
                        dbport = Integer.parseInt(kv[1]);
                        break;
                    case "dbname":
                        dbname = kv[1];
                        break;
                    case "keystorepwd":
                        keystorepwd = kv[1];
                        break;

                    case "keypwd":
                        keypwd = kv[1];
                        break;
                }
            }
            final NioSslServer S = new NioSslServer("TLSv1.2", port,
dbname, dbport, keystorepwd, keypwd);
            Runtime.getRuntime().addShutdownHook(new Thread()
            {
                @Override
                public void run()
                {
                    S.stop();
                    logger.log(Level.INFO, "Сервер остановлен");
                }
            });
            System.out.println("Сервер запущен");
            logger.log(Level.INFO, String.format("Сервер запущен на
порте %d. База данных: %s, порт СУБД: %d", port, dbname, dbport));
            S.start();
        }
        catch(IOException ex)
        {
            logger.log(Level.SEVERE, "Ошибка при чтении файла: ",
ex);
        }
        catch(Exception ex)
        {
            logger.log(Level.SEVERE, "Ошибка при запуске сервера: ",
ex);
            System.exit(1);
        }
    }
}
```

Приложение В

(обязательное)

Программный сервис ведения учебного процесса

Презентация

Программный сервис ведени
учебного процесса

студент группы 19ВВВ1 Кочетов Денис Виталье
ый руководитель:
к.т.н., доцент, Дубравин Алексей Викторович

Цель работы и работы

ь работы – разработать систему программ кли
зерной архитектуры для ведения учеб
цесса.

ект работы – приложения клиента и серв
олняющие следующие функции:

озможность ведения успеваемости обучающих

озможность учёта посещения занятий;

озможность информирования самих обучающ
их родителей (законных представителей) о ход
езультатах учебного процесса.

Постановка задачи

роцессе разработки должны быть реше
дующие задачи:

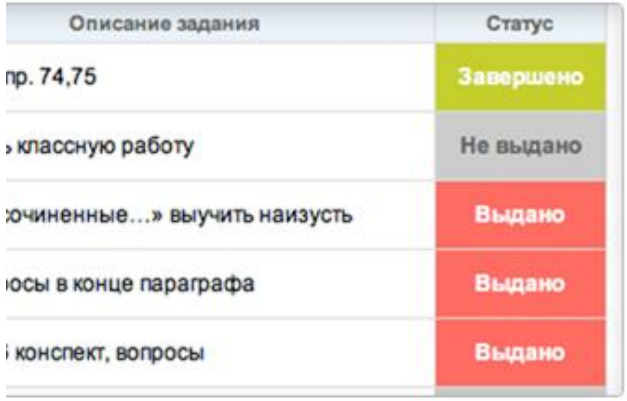
утентификация пользователей;

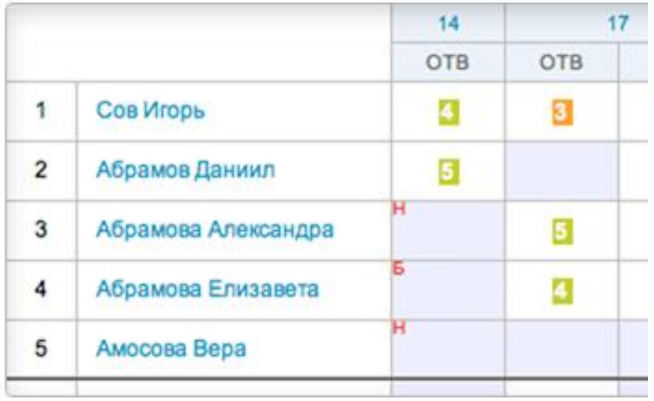
рганизация разделения ролей;

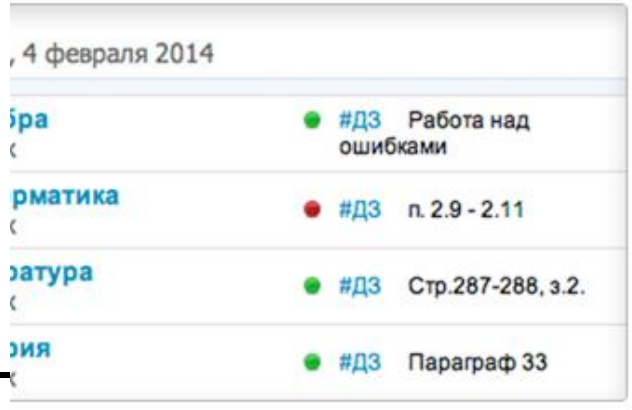
рганизация хранения данных;

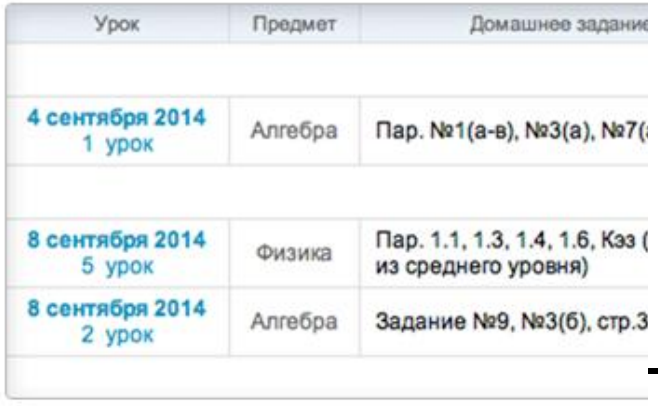
беспечение функционала клиентск
риложения согласно роли пользователя

Аналоги

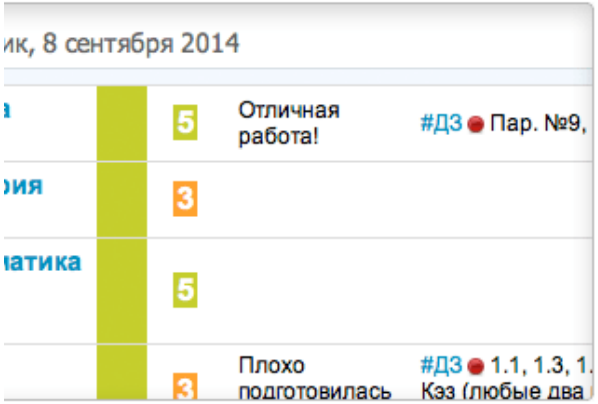

Описание задания
Статус
пр. 74,75
Завершено
классную работу
Не выдано
очиненные…» выучить наизусть
Выдано
осы в конце параграфа
Выдано
конспект, вопросы
Выдано

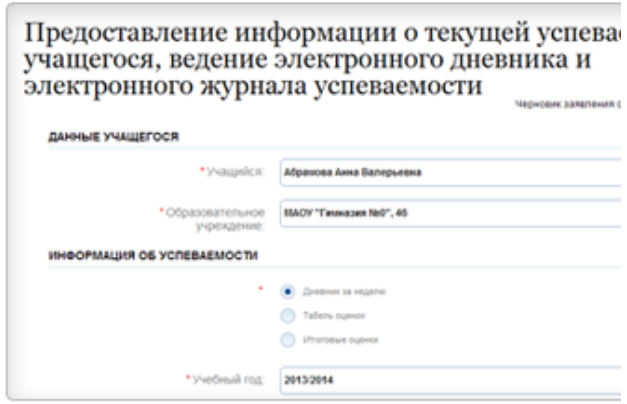

14
17
ОТВ
ОТВ
1 Сов Игорь 4 3
2 Абрамов Даниил 5
3 Абрамова Александра Н 5
4 Абрамова Елизавета Б 4
5 Амосова Вера Н

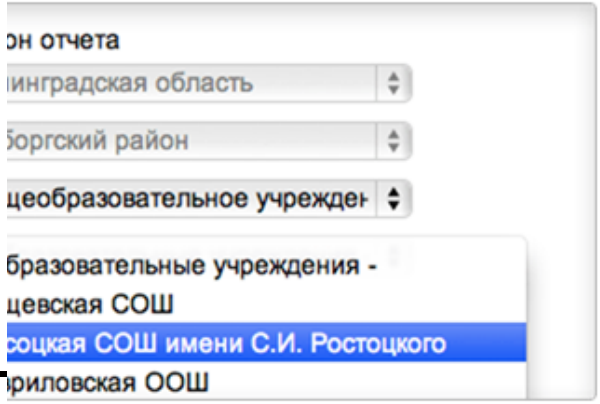

4 февраля 2014
#ДЗ Работа над ошибками
#ДЗ п. 2.9 - 2.11
#ДЗ Стр.287-288, з.2.
#ДЗ Параграф 33

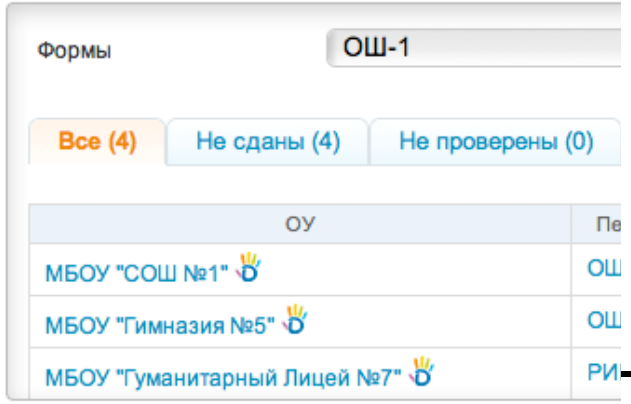

Урок
Предмет
Домашнее задание
4 сентября 2014 1 урок
Алгебра
Пар. №1(а-в), №3(а), №7(
8 сентября 2014 5 урок
Физика
Пар. 1.1, 1.3, 1.4, 1.6, Кэз (
из среднего уровня)
8 сентября 2014 2 урок
Алгебра
Задание №9, №3(б), стр.3

Аналоги

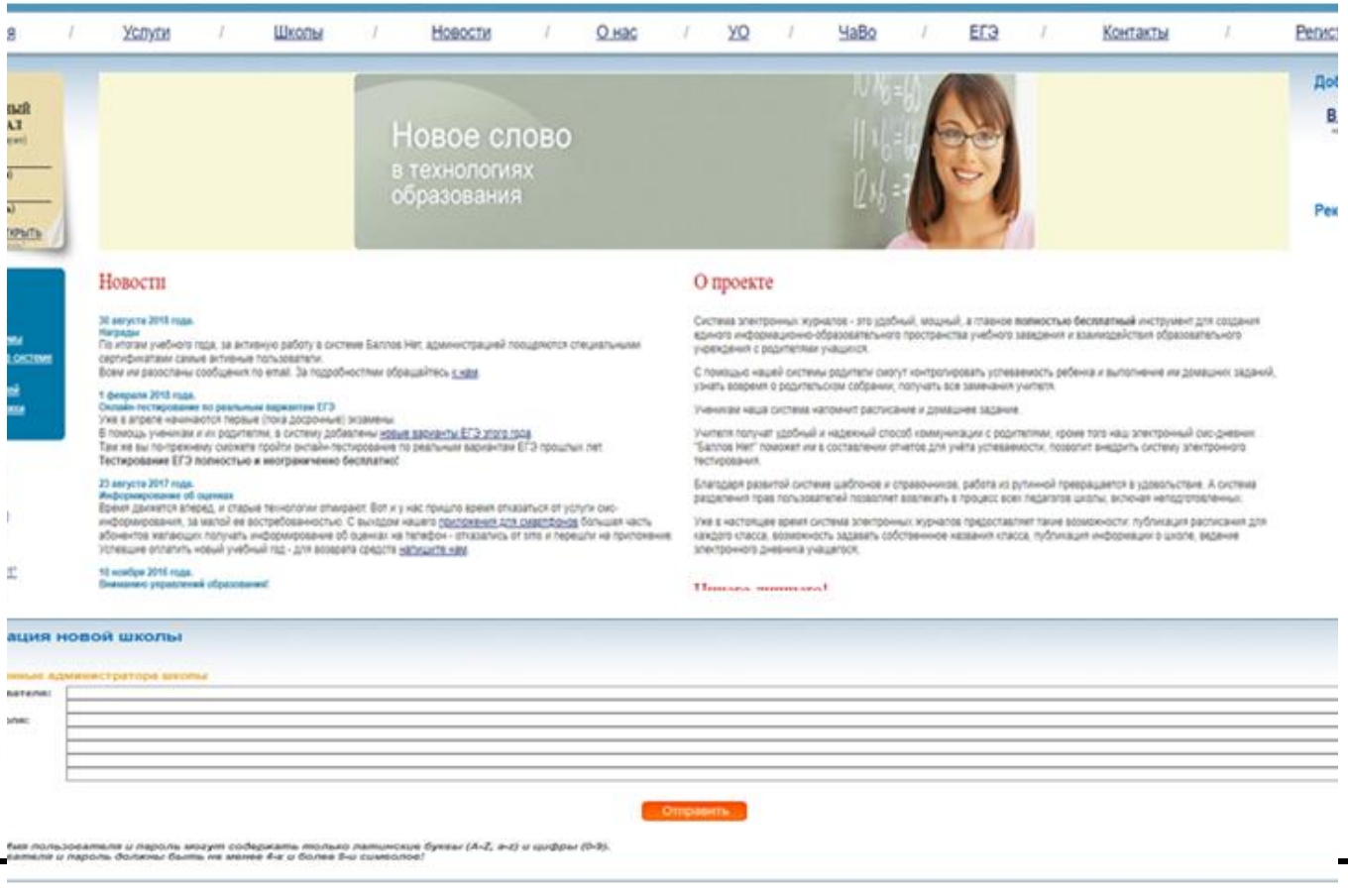

ик, 8 сентября 2014
5
Отличная работа!
#ДЗ Пар. №9,
ия
3
атика
5
3
Плохо подготовилась
#ДЗ 1.1, 1.3, 1.
Кэз (любые два


Предоставление информации о текущей успева
учащегося, ведение электронного дневника и
электронного журнала успеваемости
ДАННЫЕ УЧАЩЕГОСЯ
Учащийся:
Образовательное учреждение:
ИНФОРМАЦИЯ ОБ УСПЕВАЕМОСТИ
Дневник за неделю
Табель оценок
Итоговые оценки
Учебный год:
2013/2014

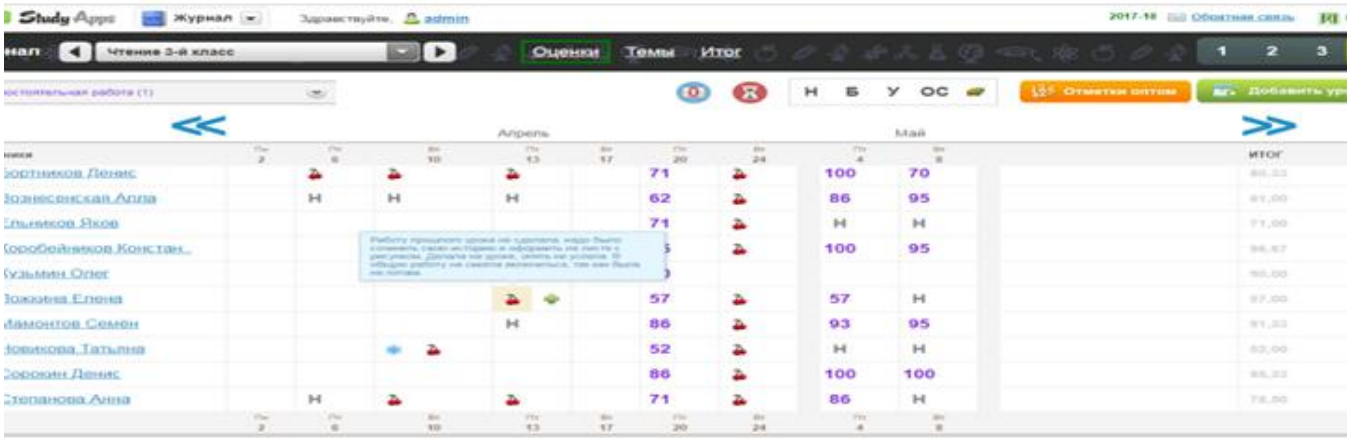

он отчета
инградская область
оргский район
цеобразовательное учрежде
бразовательные учреждения -
цевская СОШ
соцкая СОШ имени С.И. Ростоцкого
риловская ООШ

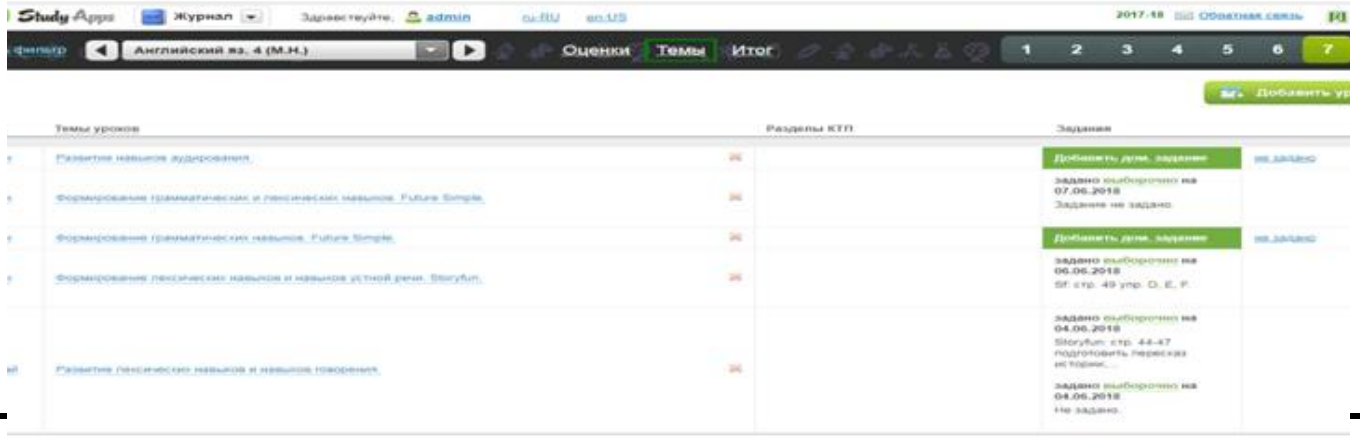

Формы
ОШ-1
Все (4)
Не сданы (4)
Не проверены (0)
ОУ
МБОУ "СОШ №1"
МБОУ "Гимназия №5"
МБОУ "Гуманитарный Лицей №7"

Аналоги


Услуги
Школы
Новости
О нас
УО
ЧаВо
ЕГЭ
Контакты
Новое слово
в технологиях
образования
Новости
О проекте
Отправить

Аналоги


Study Apps
Журнал
Здравствуйте, admin
Чтение 3-й класс
Оценки
Темы
Итог
Апрель
Май


Study Apps
Журнал
Здравствуйте, admin
Английский яз. 4 (М.Н.)
Оценки
Темы
Итог
Темы уроков
Разделы КТП
Задания

Стек технологий

Архитектура «клиент-сервер»
Манипуляции с БД
Программа на сервере
Запрос к базе данных
Ответ

ротокол сетевого взаимодействи

це работы был
аботан протокол
вого
модействия,
оказано на UML-
рамме
едовательностей

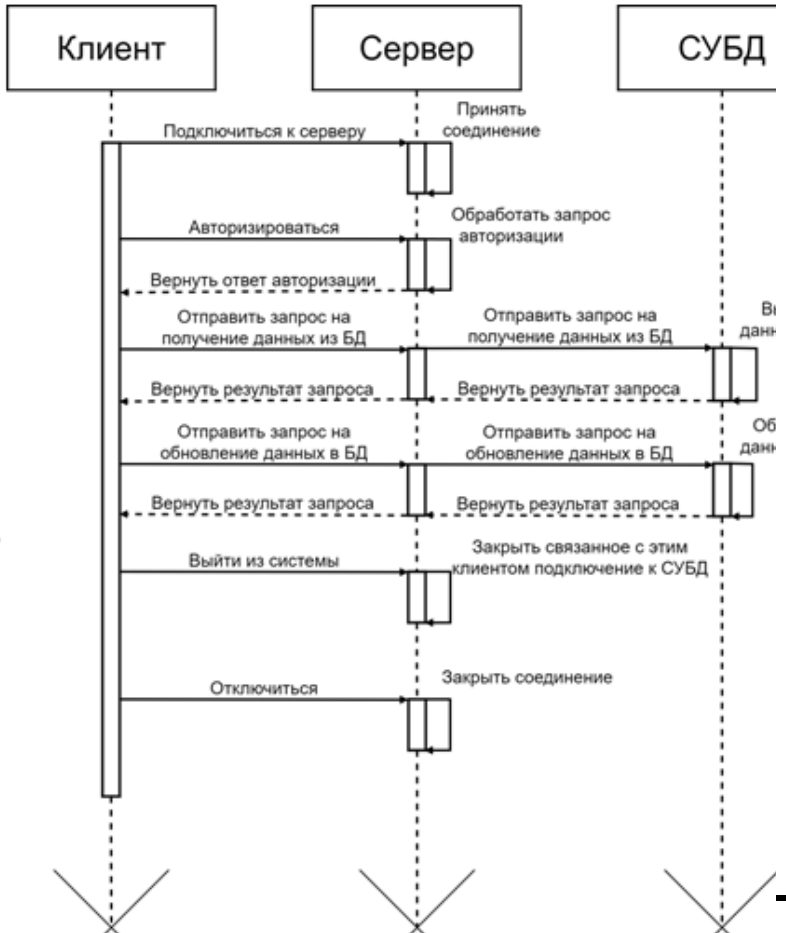


Диаграмма прецедентов

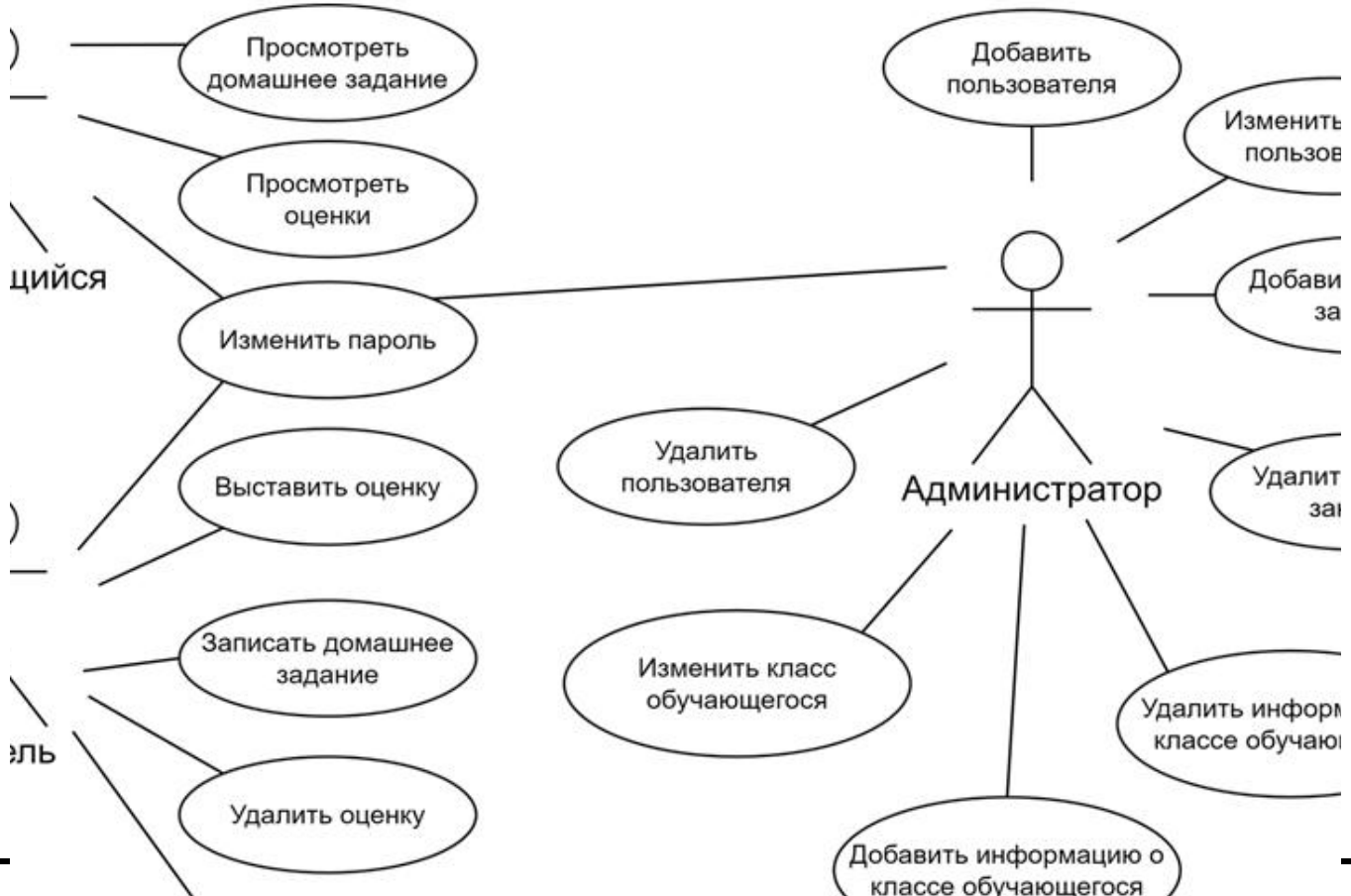


Диаграмма «сущность-связь» логическая

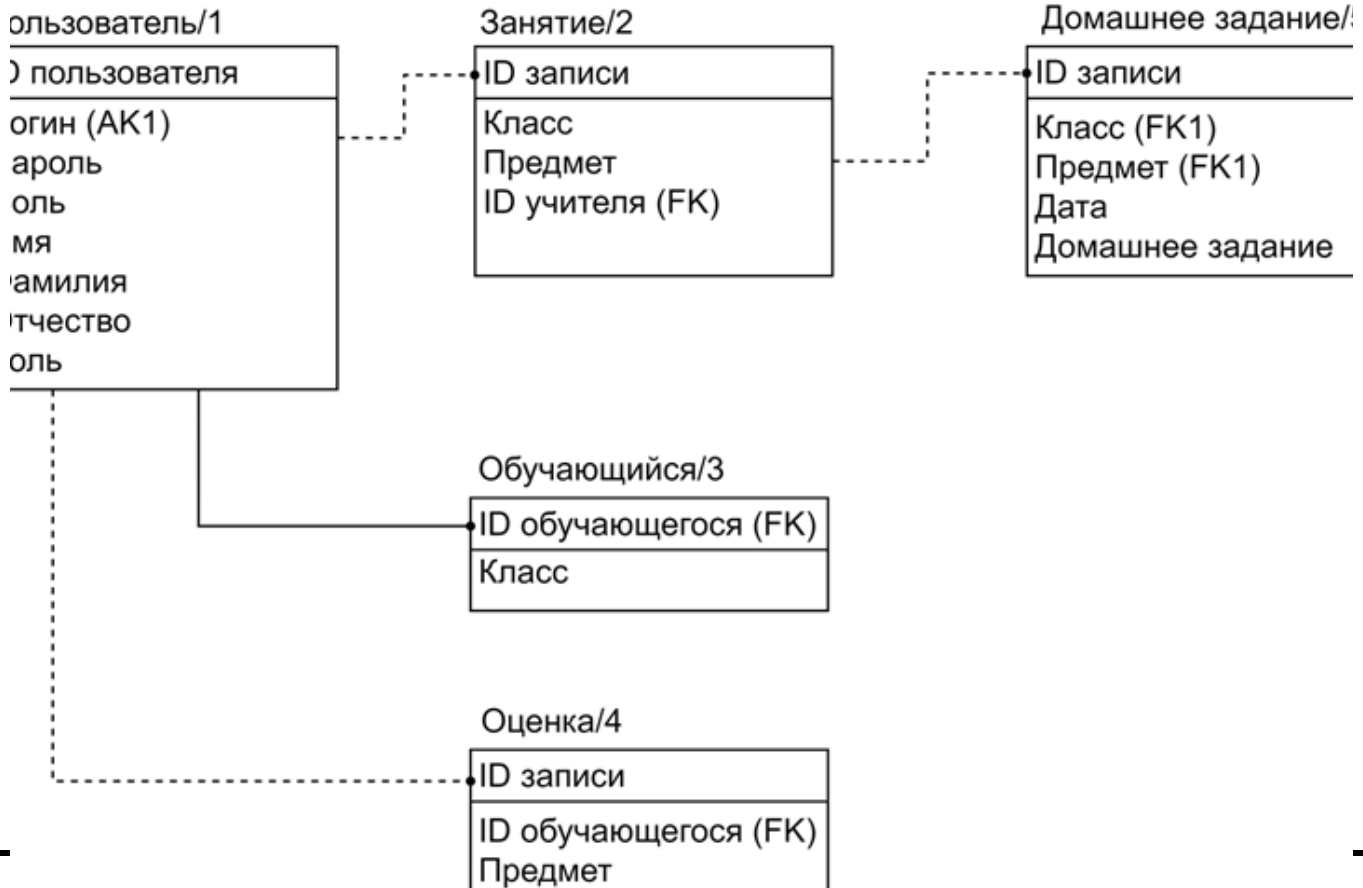
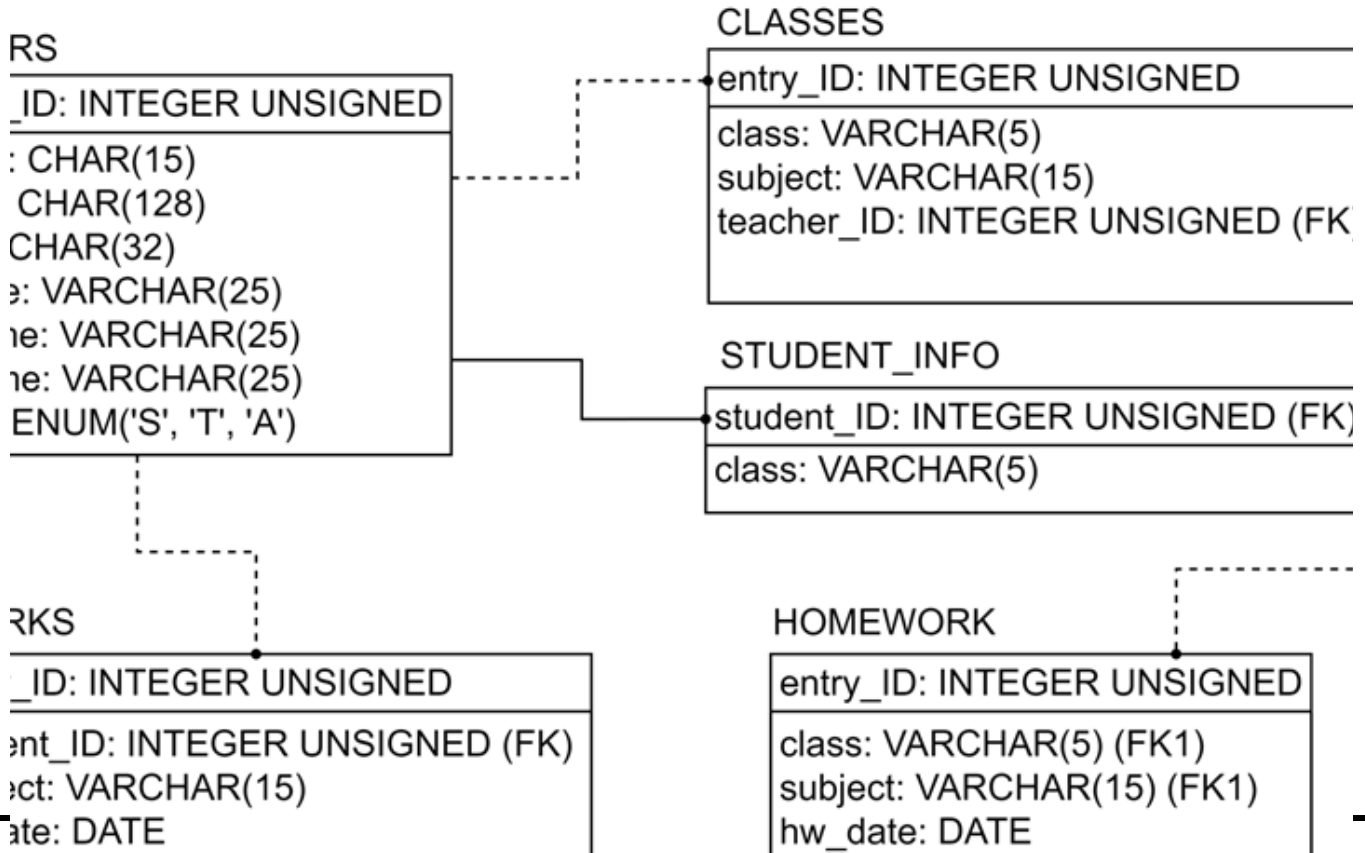
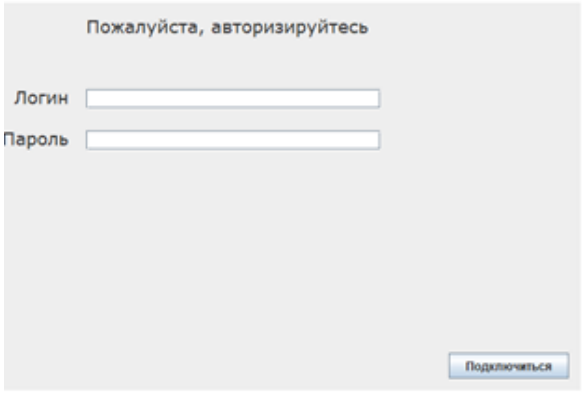


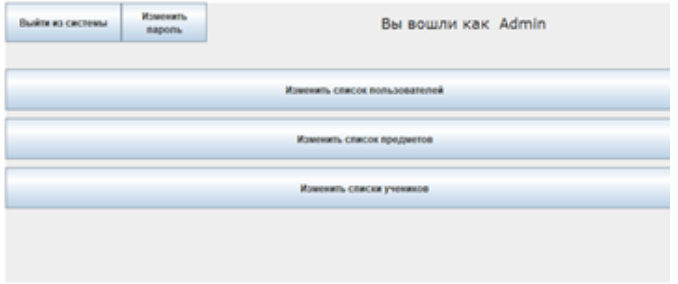
Диаграмма «сущность-связь» физическая



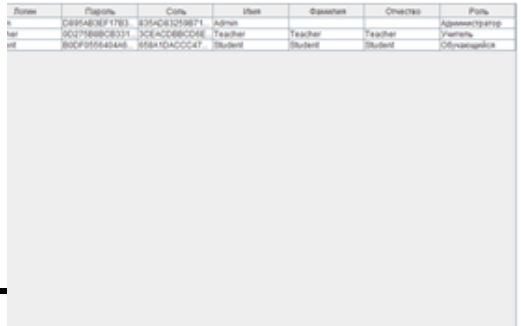
ид окон клиентского приложени



Форма LoginForm



Форма AdminForm



Диалоговое окно changePwdDialog

Заключение

це выполнения выпускной квалификационной рабо
разработан программный сервис ведения учебного
есса школы на языке Java. Графический интерфейс
изован с помощью библиотеки Swing.

танная система позволяет:

- учёт результатов учебного есса, посещаемости ающихся;
- ить информацию о шнем задании;
- елять роли и тифицировать зователей.
- информацию прикладной сти хранить на сервере БД.

Возможно усовершенствовать разработанный сервис, добавив:

- возможность создавать статис отчёты и проводить аналитику
- улучшенный графический инте возможностью работать с неск таблицами сразу;
- возможность получать доступ сервису с помощью веб-интер есть, интернет-браузера;
- интеграцию с различными сто информационными сервисами например, электронной почто

Спасибо за внимание!