

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

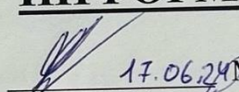
Направление подготовки: 09.04.01 «Информатика и вычислительная техника»
Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

СИСТЕМА ХРАНЕНИЯ И ОБРАБОТКИ
ИНФОРМАЦИИ

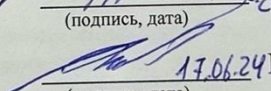
Студент

 17.06.24 Махров Антон Анатольевич
(подпись, дата)

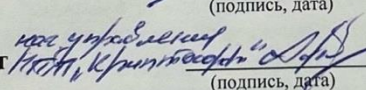
Руководитель

 17.06.24 Слепцов Николай Владимирович
(подпись, дата)

Нормоконтролёр

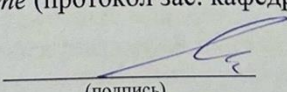
 17.06.24 Бычков Андрей Станиславович
(подпись, дата)

Рецензент

 17.06.24 М.И.О. (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 № 11)

Заведующий кафедрой


(подпись)

М. А. Митрохин

Работа защищена с отметкой отлично (протокол заседания ГЭК от

18.06.2024 № 32)

Секретарь ГЭК


(подпись)

Е.И. Гурин

Пенза 2024

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«13» сентября 2022 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Махров Антон Анатольевич гр. 22ВВИм1 факультета вычислительной техники, направления 09.04.01.

2. Тема работы «Система хранения и обработки информации»

Тема утверждена приказом ПГУ №765/о от "13" сентября 2022 г.

3. Руководитель магистерской диссертации к.т.н., доцент кафедры ВТ ПГУ Слепцов Н. В.

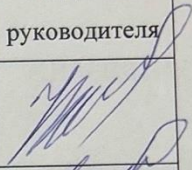
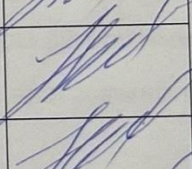
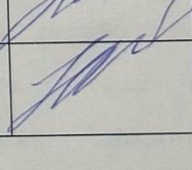
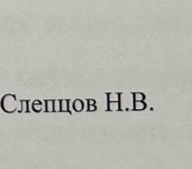
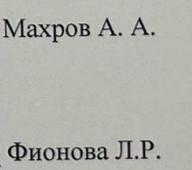
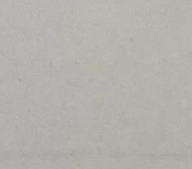
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Программное приложение должно быть реализовано в формате клиент-сервер на базе паттерна "модель-вид-контроллер". Основными функциями программного обеспечения должны быть следующие возможности: создание, удаление, обновление, чтение и запись в базу данных по маршруту клиент-сервер. Вышеописанная модель должна быть испытана на примере автоматизированной системе контроля успеваемости учащихся.

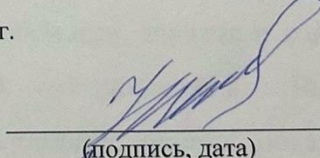
5. Перечень подлежащих разработке вопросов

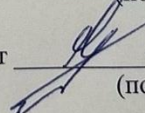
Анализ стека технологий для реализации клиент-серверных приложений и систем сбора статистики и прогнозирования. Проектирование и реализация базы данных, проектирование и реализация интерфейса, реализация бизнес-логики. Тестирование и получение результатов на базе созданного приложения.

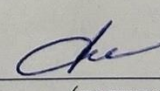
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ существующих систем хранения и обработки информации	15 %	13.09.2022-16.12.2022	
2.	Анализ средств разработки web-приложений	10 %	17.12.2022-18.02.2023	
3.	Анализ существующих СУБД и средств разработки БД	20 %	19.02.2023-31.05.2023	
4.	Анализ способов тестирования клиент-серверных приложений	15 %	1.06.2023-27.08.2023	
5.	Разработка серверной части приложения	15 %	28.08.2023-31.11.2023	
6.	Разработка клиентской части приложения	10 %	01.12.2023-31.01.2024	
7.	Тестирование системы	5 %	01.02.2024-04.04.2024	
8.	Оформление документации	10 %	05.04.24 - 31.05.24	

Дата выдачи задания "13" сентября 2022 г.

Руководитель магистерской диссертации  Слепцов Н.В.
(подпись, дата)

Задание к исполнению принял студент  17.06.2024 Махров А. А.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
(подпись, дата)
17.06.2024

РЕФЕРАТ

Отчет о практике содержит 88 листов, 27 рисунка, 14 таблиц, 6 источников

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, БАЗЫ ДАННЫХ, ХРАНЕНИЕ ИНФОРМАЦИИ, ОБРАБОТКА ИНФОРМАЦИИ

Объектом системы является система хранения и обработки информации на примере автоматизированной системы контроля успеваемости учащихся в школе.

Цель работы – разработка программного обеспечения системы контроля успеваемости учащихся в архитектуре клиент-сервер.

Разработка выполнена на языке Python.

В процессе разработки был проведён анализ требований к разрабатываемому продукту, анализ аналогичных систем, анализ алгоритмов прогнозирования, анализ сетевого взаимодействия сервисов программного приложения.

В результате выполнения магистерской работы было разработано программное обеспечение, осуществляющее выставление оценок в электронный журнал учащихся, посредством авторизации каждого преподавателя для дальнейшего просмотра, анализа и контроля выполнения домашних заданий с помощью web-приложения. Было осуществлено тестирование разработанного приложения, которое показало надёжность и правильность его работы.

					ПГУ 1.090401.14.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Махров А.А.		17.06.24			
Провер.		Слепцов Н.В.		17.06.24			
Н. Контр.		Бычков А.С.		17.06.24			
Утверд.		Митрохин М.А.		17.06.24			
					Лит.	Лист	Листов
						2	88
					Система хранения и обработки информации		
					Пояснительная записка		
					ФВТ, каф. ВТ		
					гр. 22ВВИМ1		

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	3
ВВЕДЕНИЕ	4
1 Программные средства администрирования системы контроля успеваемости учащихся.....	5
1.1 Анализ предметной области	5
1.1.1 Цели и задачи магистерской диссертации	6
1.2 Обзор технологий веб-приложений	7
1.2.1 Обзор языка программирования Python	7
1.2.2 Обзор СУБД PostgreSQL	8
1.2.3 Обзор фреймворка Django.....	10
1.2.4 Обзор библиотеки Pandas.....	17
2 Методы и средства решения задачи.....	Error! Bookmark not defined.
2.1 Требования к функциональному назначению приложения.....	Error!
Bookmark not defined.	
2.2 Анализ требований к интерфейсу приложения	Error! Bookmark not defined.
2.3 Проектирование программного обеспечения ...	Error! Bookmark not defined.
2.3.1 Проектирование структур данных	Error! Bookmark not defined.
2.3.1.1 Концептуальное проектирование.....	Error! Bookmark not defined.
2.3.2 Логическое проектирование	Error! Bookmark not defined.
2.3.3 Даталогическое проектирование..	Error! Bookmark not defined.
2.4 Проектирование пользовательского интерфейса	Error! Bookmark not defined.
3 Реализация алгоритма прогнозирования	47
ЗАКЛЮЧЕНИЕ	58
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	59
Приложение А – Листинг программы	60

Приложение Б – Презентация.....	76
---------------------------------	----

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей магистерской диссертации применяются следующие сокращения и обозначения

БД – база данных

НФ – нейронные сети

ПЗ – цифровая обработка изображений

ПИ – пользовательский интерфейс

ПК – персональный компьютер

ПО – программное обеспечение

ПЭВМ – персональная электронно-вычислительная машина

СУБД – система управления базой данных

ЭВМ – электронно-вычислительная машина

ВВЕДЕНИЕ

В современном мире информационные технологии играют ключевую роль в управлении и организации образовательного процесса. Эффективная система учёта учащихся является неотъемлемой частью образовательной инфраструктуры, позволяя обеспечить актуальность, доступность и целостность информации об учащихся. Такая система не только упростит получение всей необходимой информации только родителям учащихся и сотрудникам образовательных учреждений, но и отделам и департаментам образования.

Основными задачами магистерской диссертации являются: создание комплексной системы учёта учащихся, которая будет отвечать современным требованиям к информационной безопасности, удобству использования и функциональности. Система должна упростить заполнение электронного журнала посещаемости, предоставить возможность просмотра оценок родителям учащихся, простое и удобное управление расписанием, анализ и прогнозирование будущих оценок учеников.

Целью проектирования является разработка системы учёта учащихся, которая предполагает тесное взаимодействие с методическими и учебными отделами образовательных учреждений, что позволит учесть все необходимые требования и особенности работы с данными учащихся. Выполнив анализ аналогичных продуктовых решений, были выдвинуты требования к основному функционалу: система должна стать надёжным инструментом в руках учителей и административного персонала, способствующим повышению качества образования и оптимизации учебного процесса. Система должна автоматизировать выставление оценок в журнал, предоставить доступ ученикам к электронному дневнику, облегчить выдачу домашних заданий, обеспечить возможность просмотра расписания в удобной форме и анализировать показатели учащихся по разным предметам. При написании диссертации были учтены все функциональные требования для потенциальных пользователей системы.

1 Программные средства администрирования системы контроля успеваемости учащихся

1.1 Анализ предметной области

В современном мире, где технологии стремительно меняют нашу жизнь, образовательные учреждения также должны адаптироваться к новым реалиям. Разработка электронной системы учета учащихся является ключевым шагом на пути к цифровизации образования и предоставляет уникальные возможности для школ, стремящихся к инновациям.

Рассмотрим преимущества внедрения такой системы в образовательный процесс. Во-первых, автоматизация процессов учета экономит время и снижает вероятность ошибок. Во-вторых, доступность информации. Она доступна в любое время и из любого места, что удобно не только для учителей, но и для учеников и их родителей. В-третьих, это экологичность. Использование такой системы снизит расход бумаги, что положительно отразится на окружающей среде. В-четвертых, несанкционированный доступ к данным пользователей такой системы менее вероятен, чем к информации на бумажных носителях. В-пятых, анализировать информацию гораздо легче в электронном формате, чем на бумаге.

Теперь нужно понять, почему сторонники традиционного подхода к ведению образовательного процесса не желают переходить на новый современный электронный формат. Во-первых, необходимость обучения сотрудников школы. Это может стать достаточно большим препятствием, так как у большого количества педагогов низкий уровень владения ПК. Во-вторых, это техническая оснащенность школ. Это может быть как и неполная укомплектованность школ ПК, так работающий с низкой скоростью интернет.

Внедрение электронной системы учета учащихся открывает новые горизонты для образовательных учреждений, позволяя им быть на переднем крае образовательных инноваций. Это не только улучшает управление информацией и образовательный процесс, но и демонстрирует готовность школы к будущему, где технологии и образование идут рука об руку.

1.2 Цели и задачи магистерской диссертации

Целью магистерской диссертации является разработка программного обеспечения системы контроля успеваемости учащихся. Разрабатываемая система рассчитана, на преподавателей оценивающих учащихся по бальной системе. Им важно получать статистику успеваемости, наглядно видеть промежуточные и итоговые результаты, как всего класса, так и конкретно одного ученика. Также разрабатываемую систему могут применять учащиеся и их родители, чтобы посмотреть рейтинг успеваемости и оценить свой уровень знаний.

К основным функциям программ, реализующих работу подсистемы, администраторской части программного обеспечения администрацией учебного заведения относятся:

- регистрация школы в системе и получение личного кода доступа к функциям администрирования школой,
- авторизация,
- просмотр статистических данных школы,
- формирование списков преподавателей и привязка их к школьным предметам,
- формирование учебных классов,
- добавление учащихся,
- формирование учебных дисциплин,
- вычисление средней оценки класса,
- отслеживание динамики бальных показателей класса и отдельно взятых учащихся.

К основным функциям программ, реализующих работу подсистемы, администраторской части программного обеспечения преподавателями учебного заведения относятся:

- авторизация,
- получение выборки дисциплин, преподаваемых конкретным учителем,
- выставление оценок по своим предметам,

- выдача домашнего задания учащимся.

К основным функциям программ, реализующих просмотр успеваемости учениками, относятся:

- авторизация,
- просмотр всех учеников класса,
- получение домашних заданий,
- получение данных о преподавателях, которые ведут занятия в классе где учится ученик.

Система может эксплуатироваться только в сети Internet. Для развертывания веб-сервера необходим ПК со следующими требованиями:

- оперативная память (RAM): минимум 512 МБ, но для комфортной работы рекомендуется 2 ГБ или больше,
- жесткий диск с объемом памяти не менее 1 ГБ свободного места,
- монитор с разрешением экрана 1024x768 или выше для комфортной работы,
- процессор с тактовой частотой не ниже 2000 МГц.

1.3 Обзор технологий веб-приложений

1.3.1 Обзор языка программирования Python

Для разработки серверной части приложения был выбран язык программирования Python. Он является одним из самых популярных языков программирования для разработки веб-серверов по множеству причин. Во-первых, Python обладает простым и читаемым синтаксисом, что делает его доступным для начинающих программистов, а также удобным для опытных разработчиков. Это ускоряет процесс разработки и облегчает поддержку кода.

Богатая стандартная библиотека Python и обширное сообщество предоставляют множество готовых решений и фреймворков для веб-разработки, таких как Django, Flask и Pyramid. Эти инструменты позволяют быстро создавать надежные и масштабируемые веб-приложения.

Python также поддерживает асинхронное программирование, что является важным аспектом при создании высокопроизводительных веб-

серверов. Благодаря библиотекам, таким как `asyncio`, разработчики могут обрабатывать большое количество соединений без блокировки и с минимальной задержкой.

Интеграция с другими технологиями и системами также является сильной стороной Python. Он легко взаимодействует с базами данных, системами кэширования и другими веб-сервисами, что делает его идеальным выбором для комплексных веб-проектов.

Python активно используется в научных исследованиях, машинном обучении и анализе данных, что делает его привлекательным для проектов, требующих сложных вычислений и обработки данных. Веб-серверы, написанные на Python, могут легко интегрироваться с аналитическими системами и использовать преимущества мощных библиотек, таких как `NumPy` и `pandas`.

Наконец, Python имеет открытый исходный код и поддерживается большим сообществом разработчиков по всему миру. Это обеспечивает его постоянное развитие и обновление, а также доступность помощи и ресурсов для решения любых возникающих проблем.

Сервер, написанный на python может быть установлен на любую ОС (релизом не ранее 2016 года) и стабильно работать. Для взаимодействия с интерфейсом потребуется браузер, например, `yandex-браузер`.

В совокупности эти факторы делают Python идеальным инструментом для создания современных, эффективных и гибких веб-серверов, способных удовлетворить потребности различных проектов и организаций.

1.3.2 Обзор СУБД PostgreSQL

В качестве СУБД выбор пал на `postgresql`. PostgreSQL является одной из самых популярных систем управления базами данных (СУБД) с открытым исходным кодом, и его популярность продолжает расти благодаря множеству преимуществ, которые он предлагает для разработки и эксплуатации баз данных.

Производительность и надежность являются ключевыми характеристиками PostgreSQL. Эта СУБД обеспечивает высокую производительность даже при работе с большими и сложными базами данных, что делает её идеальной для крупных и требовательных к ресурсам проектов.

Расширяемость PostgreSQL позволяет разработчикам создавать и интегрировать свои собственные функции, операторы и типы данных, что значительно расширяет возможности стандартной СУБД. Это делает PostgreSQL гибкой и адаптируемой к различным требованиям проектов.

Соответствие стандартам SQL обеспечивает совместимость с большинством существующих приложений и упрощает миграцию из других систем. PostgreSQL строго следует стандартам SQL и часто внедряет новые стандарты раньше других СУБД.

Поддержка сложных запросов и операций, таких как подзапросы, транзакции, внешние ключи, триггеры и представления, делает PostgreSQL мощным инструментом для разработчиков, позволяя им реализовывать сложную логику на стороне сервера.

Безопасность в PostgreSQL реализована на высоком уровне с поддержкой мощных механизмов аутентификации и шифрования, что критически важно для защиты конфиденциальных данных.

Масштабируемость PostgreSQL позволяет легко расширяться как вертикально, так и горизонтально, что обеспечивает гибкость в управлении ресурсами и поддержку растущих объемов данных.

Активное сообщество разработчиков и пользователей постоянно работает над улучшением PostgreSQL, предоставляя регулярные обновления, новые функции и поддержку. Это создает динамичную и поддерживаемую среду для всех, кто использует эту СУБД. В совокупности, эти факторы делают PostgreSQL мощным, надежным и предпочтительным выбором для организаций всех размеров, которые ищут эффективное и гибкое решение для управления своими данными.

1.3.3 Обзор фреймворка Django

Фреймворк Django был разработан группой программистов из Lawrence Journal-World¹. Официально фреймворк Django был представлен и выпущен в качестве open-source проекта в июле 2005 года. Django стал быстро популярным благодаря своей простоте, мощности и модульности. Он предлагает разработчикам множество возможностей для быстрого и эффективного создания веб-приложений.

С течением времени Django стал одним из самых популярных фреймворков для веб-разработки на языке Python, и его активное сообщество разработчиков продолжает развивать и улучшать фреймворк, делая его еще более мощным и функциональным для создания различных веб-приложений. Его архитектура базируется на принципах DRY (Don't Repeat Yourself) и модели MTV (Model-Template-View), что делает разработку более эффективной и понятной. Вот его главные преимущества:

- Комплексность: Django предоставляет множество встроенных инструментов и функций, таких как аутентификация пользователей, управление сессиями, работа с базами данных и многое другое. Это упрощает разработку и ускоряет процесс создания функциональных веб-приложений.

- Богатая экосистема: Django имеет огромное сообщество пользователей, большое количество сторонних библиотек и расширений, что упрощает интеграцию с различными сервисами и расширение функциональности вашего приложения.

- Безопасность: Фреймворк Django включает множество встроенных механизмов безопасности, которые помогают защитить ваше веб-приложение от различных угроз.

- Хорошая документация: Django имеет отличную документацию, содержащую примеры кода, руководства и учебные материалы, которые помогут быстро начать работу с фреймворком и разобраться в его основах.

Рассмотрим начало работы с данным фреймворком. Для начала работы необходимо создать виртуальное окружение. Это изображено на рисунке 1.

```
1 python3 -m venv myenv
```

Рисунок 1 – создание виртуального окружения на линукс

Далее необходимо активировать виртуальное окружение. Это выполняется командой в терминале представленной на рисунке 2.

```
1 source myenv/bin/activate
```

Рисунок 2 – активация виртуального окружения

Для того, чтобы удостовериться в правильности выполненных действий в терминале должно появиться название виртуального окружения перед именем пользователя как показано на рисунке 3.

```
1 (myenv) user@host:~$
```

Рисунок 3 – пример активированного виртуального окружения

Для завершения работы с виртуальным окружением необходимо его деактивировать, как показано на рисунке 4.

```
1 deactivate
```

Рисунок 4 – деактивация виртуального окружения

После установки django в командной строке необходимо выполнить старт проекта. Таким образом будет создано базовое и готовое к работе приложение, структура которого показана на рисунке 5.

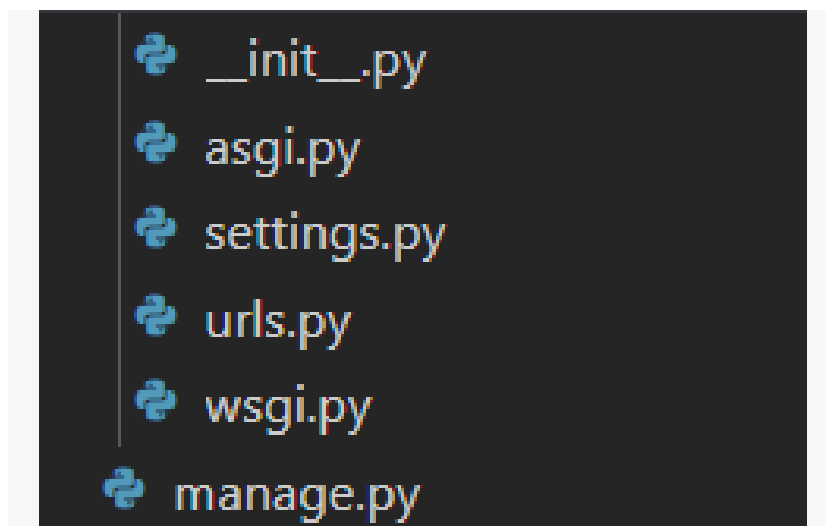


Рисунок 5 – стандартная архитектура Django-приложения

`manage.py`: Утилита командной строки, которая позволяет вам взаимодействовать с вашим проектом Django различными способами.

`settings.py`: Файл настроек для проекта Django, в котором хранится конфигурация, включая базы данных, международные настройки, статические файлы и т.д.

`urls.py`: Файл, который содержит схемы URL проекта Django для маршрутизации веб-запросов.

`wsgi.py`: Точка входа для WSGI-совместимых веб-серверов для обслуживания вашего проекта.

`asgi.py`: Точка входа для ASGI-совместимых веб-серверов, используется для асинхронного веб-сервиса.

Запуск локального сервера происходит по команде `python manage.py runserver`. Для того, чтобы удостовериться в успешном старте сервера необходимо в браузере перейти на указанный адрес и порт. Это представлено на рисунке 6.

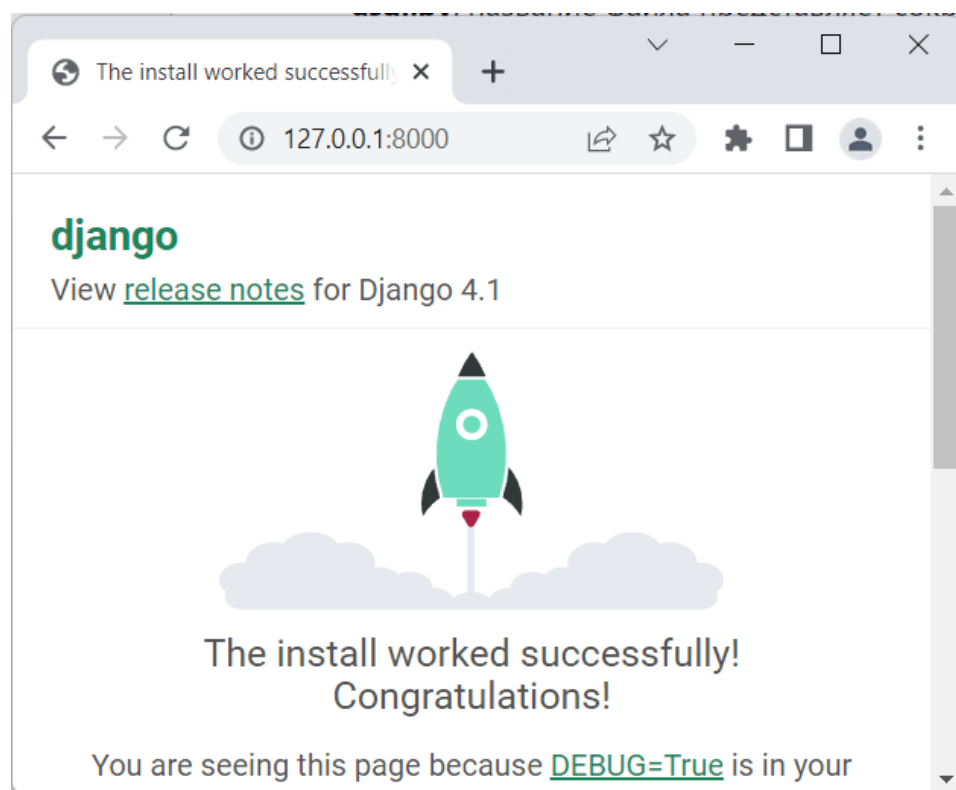


Рисунок 6 – стартовая страница запущенного сервера

Для создания собственного приложения необходимо выполнить команду `django start app app_name`. В общую директорию будет добавлен новый пакет из перечисленных выше файлов. Для того, чтобы приложение его учитывало, необходимо подключить его в общем файле настроек. Достаточно добавить название приложения в константу `installed_apps`. Это изображено на рисунке 7.

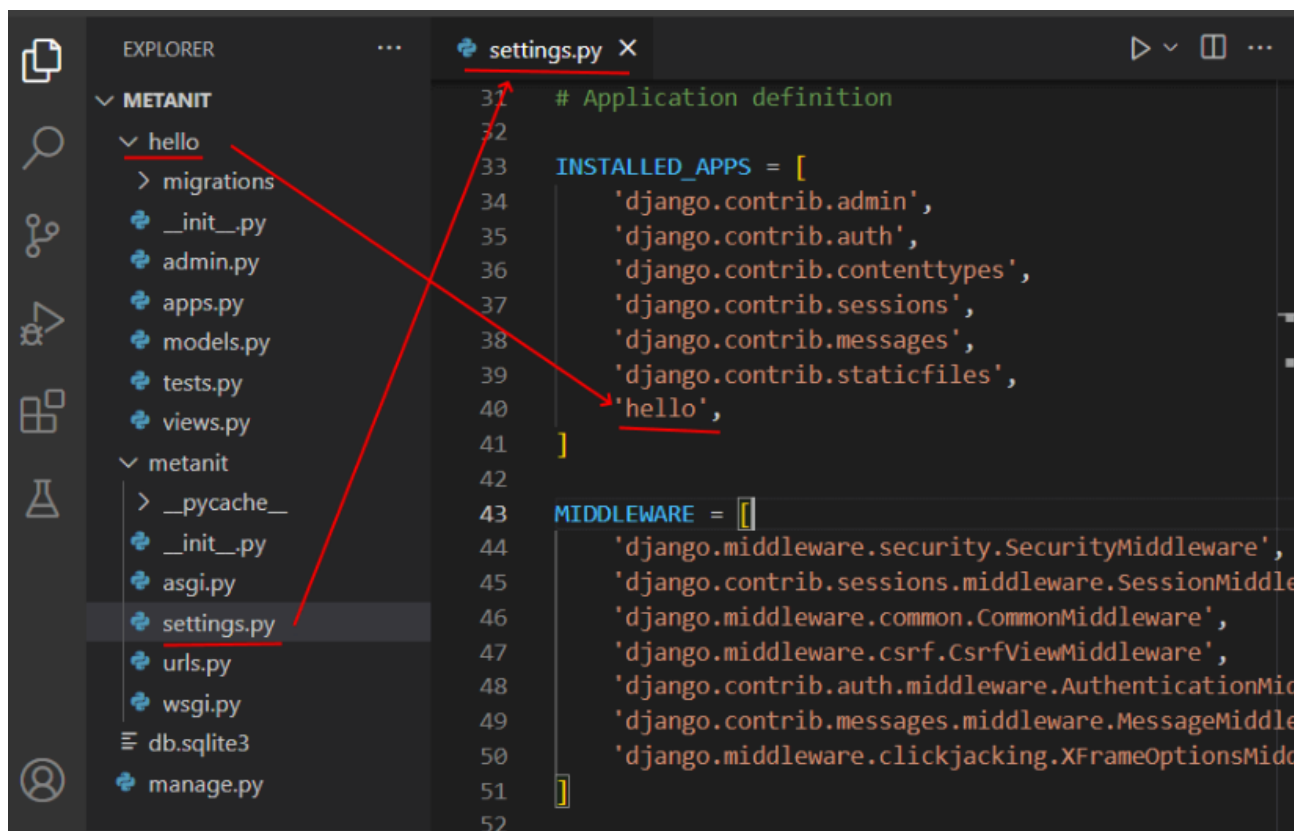


Рисунок 7 – подключение собственного приложения

Для добавления собственных конечных точек необходимо описать представления и прописать путь в файле urls.py. Входные и выходные данные для контроллера описываются в файлах views.py, как показано на рисунке 8.

```
1 from django.http import HttpResponse
2
3 def index(request):
4     return HttpResponse("<h2>Главная</h2>")
5
6 def about(request):
7     return HttpResponse("<h2>О сайте</h2>")
8
9 def contact(request):
10    return HttpResponse("<h2>Контакты</h2>")
```

Рисунок 8 – описание представлений

Результат посещения такой страницы представлен на рисунке 9.

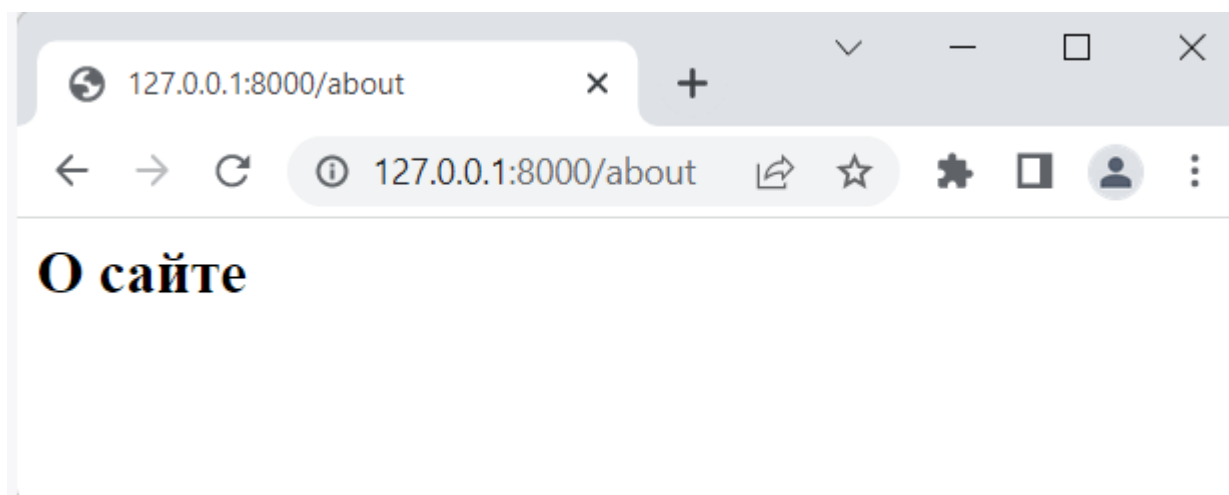


Рисунок 9 – посещение страницы контакты

Код включения пути в общий доступ представлен на рисунке 10.

```
1 from django.urls import path
2 from hello import views
3
4 urlpatterns = [
5     path("index/<int:id>", views.index),
6     path("access/<int:age>", views.access),
7 ]
```

Рисунок 10 – добавление адресов приложения

В инструментах Django имеются формы, которые легко описываются и автоматически рендерятся при html-response. Формы легко валидируются и представляют возможность сериализации и десериализации данных. Ниже показан пример простой формы на рисунке 11.

```
1 from django import forms
2
3 class UserForm(forms.Form):
4     name = forms.CharField()
5     age = forms.IntegerField()
```

Рисунок 11 – код создания Django формы

По умолчанию Django работает с mysql, однако поддерживает работу со всеми реляционными бд. Ниже представлен пример конфигурации приложения для работы с СУБД postgresql на рисунке 12.

```
1 DATABASES = {
2     'default': {
3         'ENGINE': 'django.db.backends.postgresql',
4         'NAME': 'mydatabase',
5         'USER': 'mydatabaseuser',
6         'PASSWORD': 'mypassword',
7         'HOST': '127.0.0.1',
8         'PORT': '5432',
9     }
10 }
```

Рисунок 12 – пример настроек для работы с СУБД

Таким образом, можно сделать вывод: фреймворк Django - современный и популярный инструмент для разработки веб-приложений. В его функционале заложены многие решения, о которых не приходится заботиться программисту. Его архитектура устроена таким образом, что можно легко внедрять собственные решения в разных частях приложения. Хорошо написанная документация является неотъемлемым преимуществом данного фреймворка, что позволяет быстро погрузиться в рабочий процесс.

1.3.4 Обзор библиотеки Pandas

Библиотека pandas является ключевым инструментом в мире анализа данных и науки о данных, предоставляя мощные и гибкие средства для обработки и анализа структурированных данных в языке программирования Python.

Удобство и простота использования библиотеки pandas обусловлены её высокоуровневым API, который позволяет выполнять сложные операции с данными с помощью нескольких строк кода. Это делает pandas доступной даже для тех, кто только начинает свой путь в анализе данных.

Структуры данных, такие как DataFrame и Series, являются основой pandas и предоставляют интуитивно понятные способы для хранения и манипулирования данными. DataFrame поддерживает различные типы данных и способен обрабатывать большие объемы информации, что делает его идеальным для реальных задач анализа данных.

Функциональность для чтения и записи данных в различных форматах, включая CSV, Excel, SQL и многие другие, делает pandas универсальным инструментом для взаимодействия с данными. Это позволяет аналитикам легко интегрировать pandas в свои рабочие процессы и обмениваться данными с другими приложениями.

Мощные инструменты для очистки и подготовки данных, такие как обработка пропущенных значений, фильтрация, группировка и агрегация, обеспечивают эффективную подготовку данных для анализа и моделирования.

Возможности для проведения сложного анализа данных, включая статистические расчеты, объединение данных, слияние наборов данных и многое другое, делают pandas мощным инструментом для извлечения ценных инсайтов из данных.

Визуализация данных с помощью встроенной интеграции с библиотеками matplotlib и seaborn позволяет создавать информативные графики и диаграммы непосредственно из объектов DataFrame, что упрощает интерпретацию результатов анализа.

Активное сообщество разработчиков и пользователей pandas постоянно работает над улучшением библиотеки, добавляя новые функции и улучшая существующие. Это обеспечивает постоянное развитие библиотеки и поддержку для решения новых задач анализа данных.

В совокупности, эти качества делают pandas неотъемлемым инструментом для любого аналитика данных, обеспечивая эффективную и гибкую работу с данными на всех этапах аналитического процесса.

2 Методы и средства решения задачи

2.1 Требования к функциональному назначению приложения

Программа должна поддерживать работу разных типов пользователей. Возможности, которые должны быть реализованы в зависимости от пользователя, представлены в таблице 1.

Таблица 1 – Возможности пользователей

Тип пользователя	Возможности
Администратор	Регистрирует ответственное лицо(директора), оказывает техническую поддержку по возникшим вопросам.
Директор	Формирует списки преподавателей, классов, учащихся. Связывает преподавателя с определенной дисциплиной. Привязывает к классам преподавателей.
Учитель	Выставляет оценки учащимся, просматривает списки классов, добавляет объявления для класса. Добавляет домашнее задание учащимся.
Ученик	Просматривает успеваемость (выставленные оценки), просматривает статистику успеваемости.

Доступ к возможностям системы имеют только те пользователи, которые прошли авторизацию. В соответствии с основными функциями были

разработаны диаграммы вариантов использования для каждого типа пользователей сайта, которая изображена ниже на рисунке 13.

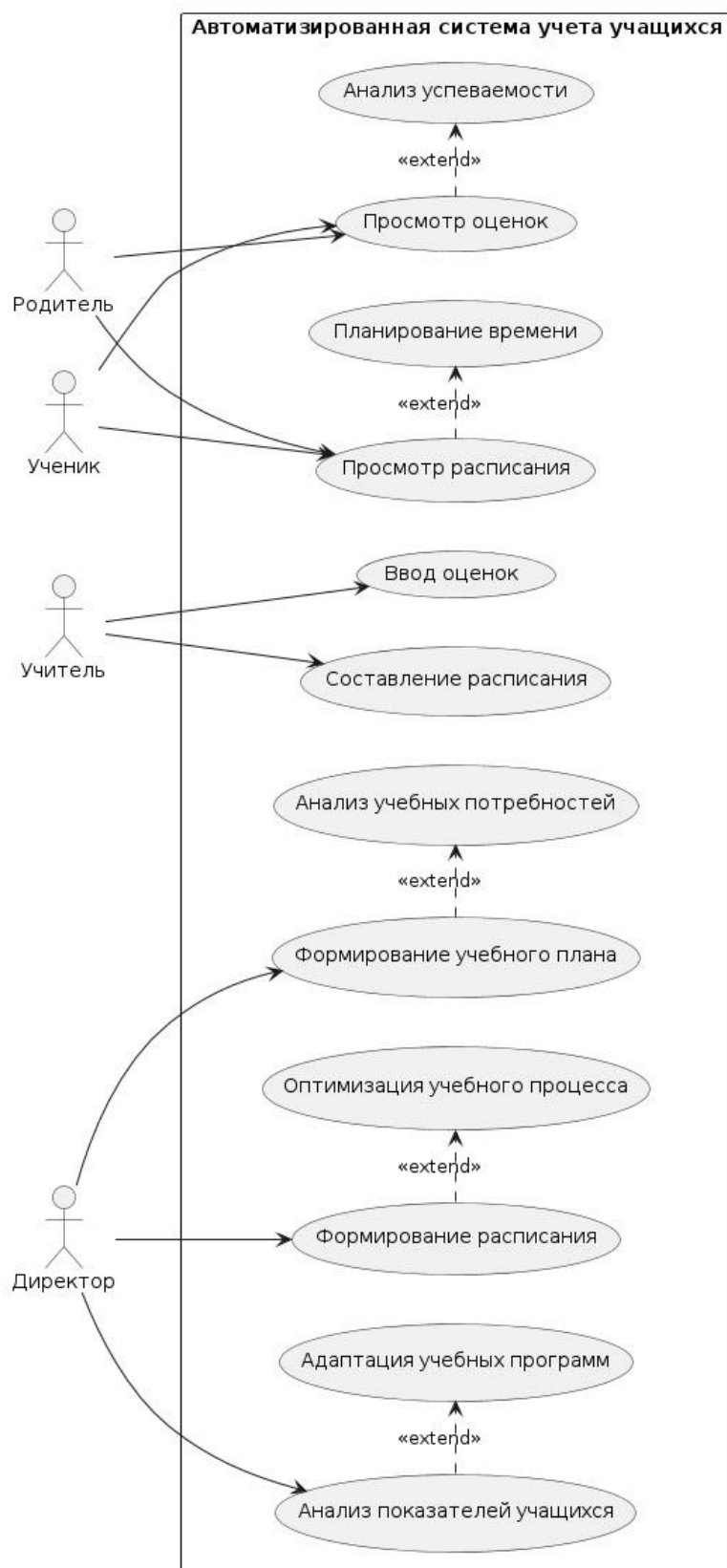


Рисунок 13 – диаграмма вариантов использования

2.2 Анализ требований к интерфейсу приложения

Формирование требований к интерфейсу приложения системы учета учащихся включает определение функциональности, которая должна быть доступна различным пользователям, а также удобства их использования. Вот основной список требований: удобство использования, интуитивно понятный интерфейс с четкой навигацией, адаптивный дизайн, подходящий для различных устройств (компьютеры, планшеты, смартфоны).

Функциональность для Родителей:

- Просмотр текущих и архивных оценок ученика,
- Просмотр расписания учебных занятий,
- Получение уведомлений о важных событиях и изменениях в расписании.

Функциональность для Учеников:

- Доступ к личному кабинету с оценками и расписанием,
- Возможность отправки обратной связи учителям и администрации.

Функциональность для Учителей:

- Ввод и редактирование оценок учеников,
- Составление и изменение расписания уроков,
- Анализ успеваемости класса и отдельных учеников.

Функциональность для Директора:

- Формирование и корректировка учебного плана,
- Анализ показателей успеваемости учащихся на уровне школы,
- Оптимизация учебного процесса на основе аналитических данных,
- Безопасность и конфиденциальность,
- Защита персональных данных согласно нормативным требованиям,
- Разграничение доступа к информации в зависимости от роли

пользователя.

Техническая поддержка:

- Наличие справочной системы по использованию приложения,
- Служба поддержки для решения технических проблем пользователей.

Эти требования помогут обеспечить, чтобы интерфейс был функциональным, безопасным и удобным для всех пользователей системы. Вы можете дополнить этот список в соответствии с конкретными потребностями вашей школы или образовательного учреждения.

2.3 Концептуальное проектирование

Процесс проектирования состоит из следующих шагов:

- определение сущностей,
- определение зависимостей между сущностями,
- задание первичных и альтернативных ключей,
- определение атрибутов сущностей.

Пользователь, работающий с программными средствами контроля успеваемости учащихся, будет добавлять информацию в базу и работать с ней. Далее в таблице 2 приведен обзор сущностей, где представлены все спроектированные сущности БД, заголовки столбцов означают следующее:

- название сущности,
- название выделенной в предметной области сущности,
- идентификация таблицы,
- физическое имя сущности,
- описание сущности.

Таблица 2 – Таблица сущностей

Название сущности	Идентификатор	Описание сущности
Школы	schools	Информация о школах
Ученики	students	Информация об учениках
Учители	teachers	Информация об учителях
Классы	school_classes	Информация о классах

Продолжение таблицы 2

Название сущности	Идентификатор	Описание сущности
Уроки	lessons	Информация об уроках
Учебные недели	learn_weeks	Информация об учебных неделях
Дисциплины	disciplines	Информация о предметах
Директоры	directors	Информация о директорах школ.
Расписания	schedules	Информация о расписаниях.
Учебные периоды	school_periods	Информация об учебных периодах
Оценки	rates	Информация об оценках
Директора	directors	Информация об директорах школ

2.4 Логическое проектирование

Логическое проектирование включает в себя следующие этапы:

- определение набора отношений исходя из концептуальной модели БД,
- проверка модели с помощью правил нормализации,
- проверка модели в отношении модели пользователей,
- создание полной атрибутивной диаграммы в рамках технологии IDEF1X,
- определение требующейся поддержки целостности.

Анализ функциональных зависимостей в процессе логического проектирования для каждого отношения приведен ниже.

Отношение: Ученики.

Схема отношения: Ученики (идентификатор ученика, фамилия ученика, имя ученика, отчество ученика, дата рождения ученика, идентификатор пользователя, идентификатор школы ученика, идентификатор учебного периода).

Альтернативный ключ: идентификатор пользователя.

Первичный ключ: идентификатор ученика.

Идентификатор ученика – идентификатор пользователя.

Отношение находится в 3НФ.

Отношение: Школы.

Схема отношения: Школа (идентификатор школы, ОГРН школы, ИНН школы, адрес школы, название школы, сайт школы, идентификатор директора).

Первичный ключ: идентификатор школы.

Альтернативный ключ: ОГРН школы, ИНН школы, название школы, адрес школы, идентификатор директора.

Отношение находится в 3НФ.

Отношение: Директоры.

Схема отношения: Директоры (идентификатор директора, фамилия директора, имя директора, отчество директора, дата рождения директора, идентификатор пользователя, идентификатор школы).

Альтернативный ключ: идентификатор пользователя.

Первичный ключ: идентификатор директора.

Идентификатор директора – идентификатор пользователя.

Отношение находится в 3НФ.

Отношение: Учители.

Схема отношения: Учителя (идентификатор учителя, фамилия учителя, имя учителя, отчество учителя, дата рождения учителя, идентификатор пользователя, идентификатор школы учителя).

Альтернативный ключ: идентификатор пользователя.

Первичный ключ: идентификатор учителя.

Идентификатор учителя – идентификатор пользователя.

Отношение находится в ЗНФ.

Отношение: Дисциплины.

Схема отношения: Дисциплины (идентификатор дисциплины, название дисциплины, идентификатор школы, идентификатор учебного периода).

Первичный ключ: идентификатор дисциплины.

Отношение находится в ЗНФ.

Отношение: Уроки.

Схема отношения: Уроки (идентификатор урока, идентификатор школы, идентификатор дисциплины, идентификатор учителя, идентификатор оценки, домашнее задание, идентификатор учебного периода).

Первичный ключ: идентификатор урока.

Отношение находится в ЗНФ.

Отношение: Пользователи.

Схема отношения: Пользователи (идентификатор пользователя, электронная почта, активность, фамилия пользователя, имя пользователя, логин, пароль).

Первичный ключ: идентификатор пользователя.

Альтернативный ключ: логин пользователя.

Идентификатор пользователя – идентификатор пользователя.

Отношение находится в ЗНФ.

Отношение: Учебный период.

Схема отношения: Учебный период (идентификатор учебного периода, начало учебного периода, конец учебного периода).

Первичный ключ: идентификатор учебного периода.

Идентификатор учебного периода: идентификатор учебного периода.

Отношение находится в 3НФ.

2.5 Даталогическое проектирование

На данном этапе разработки БД была сформирована физическая модель БД, изображенная в таблицах 3 - 10.

Таблица 3 – Отношение «Школы»

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
school_id	Идентификатор школы	int	11	-	PrimaryKey
ogrn	ОГРН школы	str	13	-	Attribute
inn	ИНН школы	str	10	-	Attribute
site	Сайт школы	str	30		Attribute
address	Адрес школы	str	50		Attribute

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
director_id	Идентификатор директора	int	11	-	ForeignKey
school_period_id	Идентификатор учебного периода	int	11	-	ForeignKey

Таблица 4 – Отношение «Ученики»

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
student_id	Идентификатор	int	11	-	PrimaryKey
last_name	Фамилия ученика	str	30	-	Attribute

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
first_name	Имя ученика	str	30	-	Attribute
patronymic	Отчество ученика	str	30		Attribute
user_id	Идентификатор пользователя	int	11	-	ForeignKey
school_id	Идентификатор школы	int	11	-	ForeignKey
class_id	Идентификатор класса	int	11	-	ForeignKey
date_of_bir th	Дата рождения ученика	date	8		Attribute

Таблица 5 – Отношение «Учители»

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
teacher_id	Идентификатор учителя	int	11	-	PrimaryKey
last_name	Фамилия учителя	str	30	-	Attribute
first_name	Имя учителя	str	30	-	Attribute
patronymic	Отчество учителя	str	30		Attribute
user_id	Индентификатор пользователя	int	11	-	ForeignKey
school_id	Идентификатор школы	int	11	-	ForeignKey
discipline_id	Идентификатор предмета	int	11	-	ForeignKey
date_of_birth	Дата рождения учителя	date	8		Attribute

Таблица 6 – отношение «Классы»

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
class_id	Идентификатор	int	11	-	Primary Key
number	Номер класса	int	11	-	Attribute
letter	Буква класса	str	1		Attribute
school_id	Идентификатор школы	int	11	-	ForeignKey
student_id	Идентификатор ученика	int	11	-	ForeignKey
school_period_id	Идентификатор школьного периода	int	11	-	ForeignKey

Таблица 7 – Отношение «Пользователи»

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
user_id	Идентификатор пользователя	int	11	-	PrimaryKey
last_name	Фамилия пользователя	str	30	-	Attribute
first_name	Имя пользователя	str	30	-	Attribute
patronymic	Отчество пользователя	str	30		Attribute
is_active	Активность пользователя	boolean	11	-	Attribute
username	Логин пользователя	str	11	-	Attribute
password	Пароль пользователя	str	8		Attribute

Таблица 8 – Отношение «Дисциплины»

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
discipline_id	Идентификатор предмета	int	11	-	PrimaryKey
name	Название дисциплины	str	30	-	Attribute
school_id	Идентификатор школы	int	11	-	ForeignKey
school_period_id	Идентификатор учебного периода	int	11	-	ForeignKey

Таблица 9 – Отношение «Уроки»

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
lesson_id	Идентификатор урока	int	11	-	PrimaryKey
theme	Тема урока	str	30	-	Attribute

Продолжение таблицы 9

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
school_period_id	Идентификатор учебного периода	int	11	-	ForeignKey
school_id	Идентификатор школы	int	11	-	ForeignKey

Таблица 10 – Отношение «Учебные периоды»

Имя столбца	Описание	Тип данных	Размерность	Возможность значения null	Роль
school_period_id	Идентификатор учебного периода	int	11	-	ForeignKey
start_date	Начало периода	date	8	-	Attribute
end_date	Конец периода	date	8	-	Attribute
name	Название периода	str	50	-	Attribute

2.6 Инфологическая модель

Инфологическая модель, созданная для школьной системы, включает в себя не только структуру данных, но и определяет правила их валидации, обработки и защиты. Важным аспектом является обеспечение целостности данных, что достигается за счет использования различных ограничений на уровне базы данных, таких как первичные ключи, внешние ключи, и проверочные ограничения (constraints).

Кроме того, модель учитывает разграничение доступа к данным различных пользователей системы. Например, ученики имеют доступ к информации о своих оценках и расписании, но не могут видеть оценки других учеников. Учителя могут вносить изменения в расписание уроков и ставить оценки, но только в рамках своих предметов. Администрация школы имеет

полный доступ ко всем данным и может управлять учетными записями пользователей.

Инфологическая модель также предусматривает возможность масштабирования системы. С учетом возможного расширения школьной системы, модель спроектирована таким образом, чтобы в будущем можно было легко добавлять новые сущности и связи между ними без необходимости кардинального изменения всей структуры базы данных.

Внедрение такой модели позволяет школе перейти на новый уровень управления данными, повысить эффективность образовательного процесса и обеспечить удобство в использовании информационной системы как для учеников, так и для персонала школы. В результате, школьная система становится более адаптивной к изменениям и способной отвечать на современные требования образовательной среды. Инфологическая модель базы данных представлена на рисунке 14.

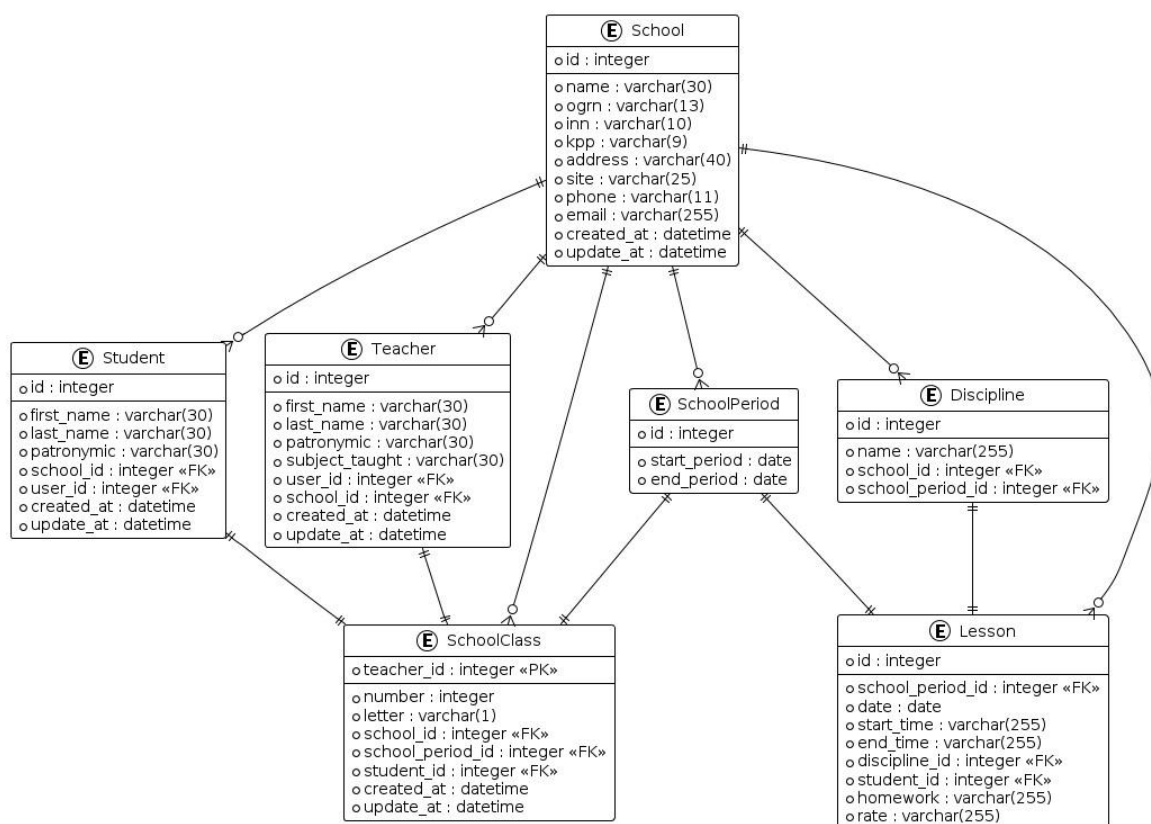


Рисунок 14 – инфологическая модель базы данных

Диаграмма коммуникаций служит для визуализации процесса взаимодействия учителя, ученика и системы прогнозирования оценок. Она отражает, как учитель вводит данные о текущих успехах ученика в систему, которая затем анализирует эту информацию и предоставляет прогноз будущих оценок. Этот прогноз помогает учителю определить, какие ученики могут нуждаться в дополнительной помощи или внимании, и планировать соответствующие меры для улучшения их успеваемости.

Система прогнозирования также отправляет данные обратно в базу данных, где они хранятся для дальнейшего использования и анализа. В конечном итоге, учитель использует полученную информацию для обратной связи с учениками, что позволяет им понять свои сильные и слабые стороны и работать над улучшением своих результатов.

Такая диаграмма становится ключевым элементом в понимании и оптимизации процесса обучения, позволяя учителям и ученикам лучше взаимодействовать и достигать высоких академических результатов. Диаграмма представлена на рисунке 15.

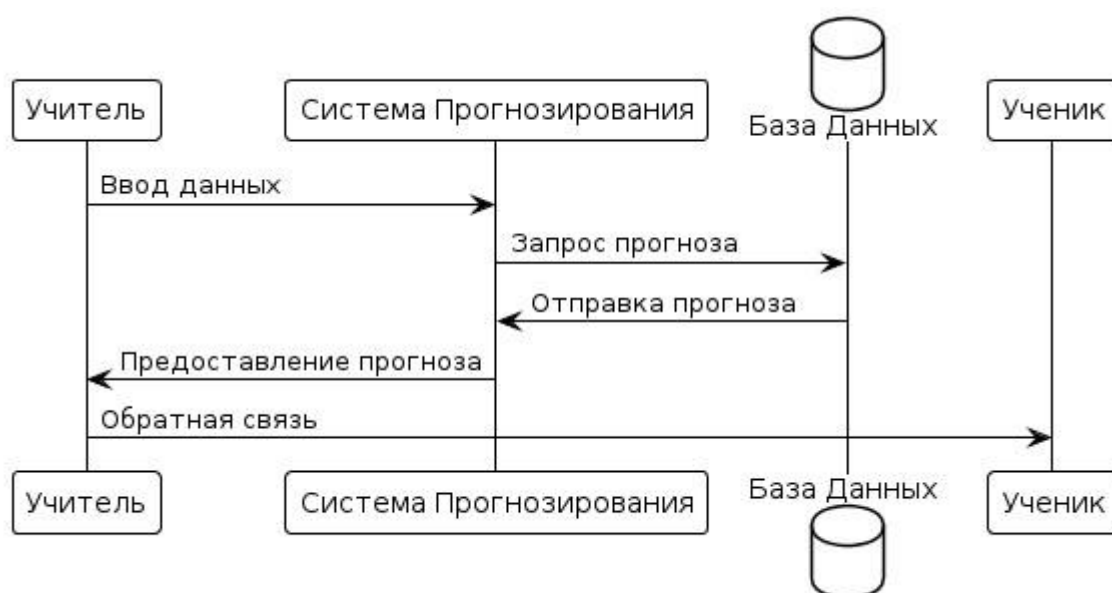


Рисунок 15 – диаграмма коммуникаций системы прогнозирования, ученика и учителя

Диаграмма последовательности, представляет собой визуальное представление шагов, которые происходят, когда учитель ставит оценку ученику в электронный журнал. Эта диаграмма важна, потому что она помогает всем участникам образовательного процесса — от разработчиков до администраторов школы — понять, как именно происходит запись оценок, какие действия предшествуют этому и какие процессы следуют за ним.

С помощью такой диаграммы можно легко идентифицировать, где именно в процессе могут возникнуть задержки или ошибки, и как эффективно их исправить. Она также служит инструментом для обучения новых сотрудников, позволяя им быстрее освоиться с процедурами школы. В конечном итоге, диаграмма последовательности улучшает общее качество управления учебным процессом и повышает прозрачность взаимодействий между учителями и учениками. Диаграмма последовательности изображена на рисунке 16.

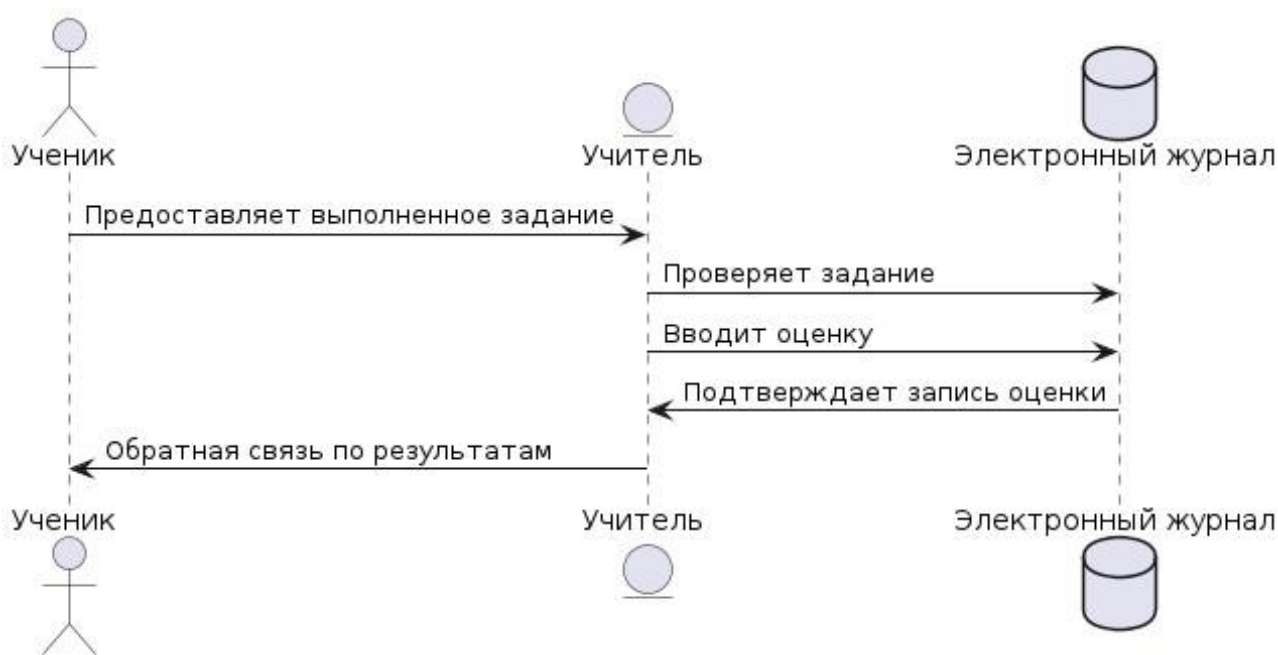


Рисунок 16 – Диаграмма последовательности выставления оценки в электронный журнал

2.7 Проектирование структуры программного обеспечения

Проектирование структуры программного обеспечения является фундаментальным этапом, который задает тон всему процессу разработки. В его основе лежит стремление к созданию архитектуры, которая не только

отвечает текущим потребностям бизнеса и пользователей, но и обладает гибкостью для адаптации к будущим изменениям. Это искусство балансировки между стабильностью и адаптивностью, между конкретными требованиями и абстрактными понятиями.

В начале процесса стоит понимание того, что каждый компонент системы должен выполнять определенную роль, будь то обработка данных, взаимодействие с пользователем или выполнение бизнес-логики. Эти компоненты связываются в единую сеть, где каждый узел важен для функционирования всей системы. Проектирование также включает в себя предвидение того, как компоненты будут взаимодействовать друг с другом и с внешним миром, обеспечивая при этом безопасность и конфиденциальность данных.

Создание прочной архитектуры требует глубокого понимания не только технических аспектов, но и бизнес-процессов, которые эта система предназначена поддерживать. Архитекторы ПО используют свой опыт и знания для создания дизайна, который будет устойчив к нагрузкам, эффективен в использовании ресурсов и легок в обслуживании.

Важно отметить, что проектирование структуры ПО не заканчивается на бумаге. Это динамичный процесс, который продолжается на протяжении всего жизненного цикла программного продукта, поскольку каждое изменение в требованиях или технологиях может потребовать пересмотра и адаптации архитектуры. Таким образом, проектирование структуры ПО — это не одноразовое действие, а непрерывный процесс, который требует внимания и усилий на всех этапах разработки программного обеспечения. Ниже на рисунке 17 представлена диаграмма компонентов.



Рисунок 17 – Диаграмма компонентов системы

2.8 Проектирование пользовательского интерфейса

На рисунке 16 показана главная страница сайта. В верхней части страницы представлена шапка, которая включает в себя приветственный текст и компоненты быстрой навигации: «Главная» - для быстрого перехода на главную страницу, «Расписание» - для доступа к странице с расписанием, «Контакты» - где представлена информация о школе, «Вход для сотрудников» - для авторизации сотрудников школы. В центральной части находится кнопка «Начать обучение», при нажатии на которую появится модальное окно с формой регистрации учащихся, изображенное на рисунке 17. Если уже пользователь авторизован, то кнопка «Вход для сотрудников» меняется на «Выход», при нажатии на которую произойдёт прерывание пользовательской сессии, что показано на рисунке 18.

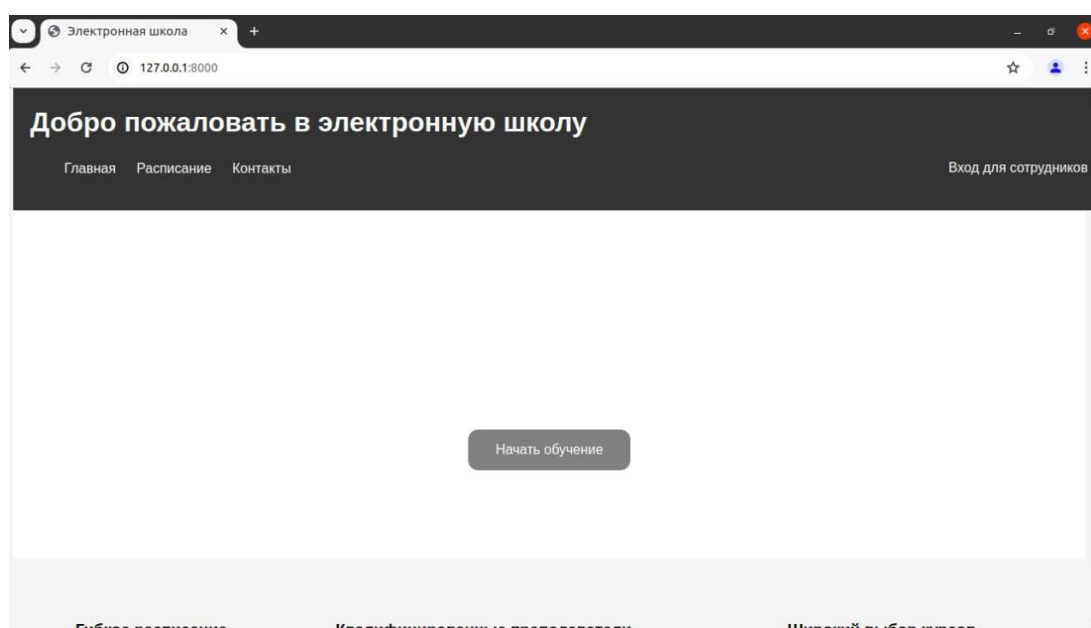


Рисунок 18 – главная страница

На модальном окне, изображенном на рисунке 19 находятся поля для регистрации. Имя, фамилия, отчество, класс, логин и пароль являются текстовыми полями для ввода текста. Название школы представлено полем выбора доступного значения из списка имеющихся зарегистрированных школ. На все поля задействована валидация, которая не позволит отправить форму с не валидными данными. При нажатии на кнопку «Начать» будет отправлена заявка на поступление в указанную на форме школу.

The image shows a web browser window with a modal registration form. The browser's address bar displays '127.0.0.1:8000'. The modal form, titled 'Имя', is centered on the screen. It contains the following fields: 'Имя' (Name) with the value 'Алексеев', 'Фамилия' (Surname) with 'Игорь', 'Отчество' (Patronymic) with 'Викторович', 'Название школы' (School Name) as a dropdown menu showing 'Школа №1', 'Класс' (Class) with '4А', 'Логин' (Login) with 'alekseev', and 'Пароль' (Password) which is masked with dots. A 'Начать' (Start) button is located at the bottom of the form. The background of the browser shows a website with a header 'Добро пожаловать в эле...' and navigation links 'Главная', 'Расписание', and 'Контакты'. A link 'Вход для сотрудников' is visible on the right side of the header.

Рисунок 19 – модальное окно с формой регистрации учащихся

Модальное окно для авторизации на сайте изображено на рисунке 20 и появляется при нажатии на кнопку «Вход для сотрудников». Название школы представляет собой поле выбора значений из списка, а логин и пароль являются полями для ввода текста.

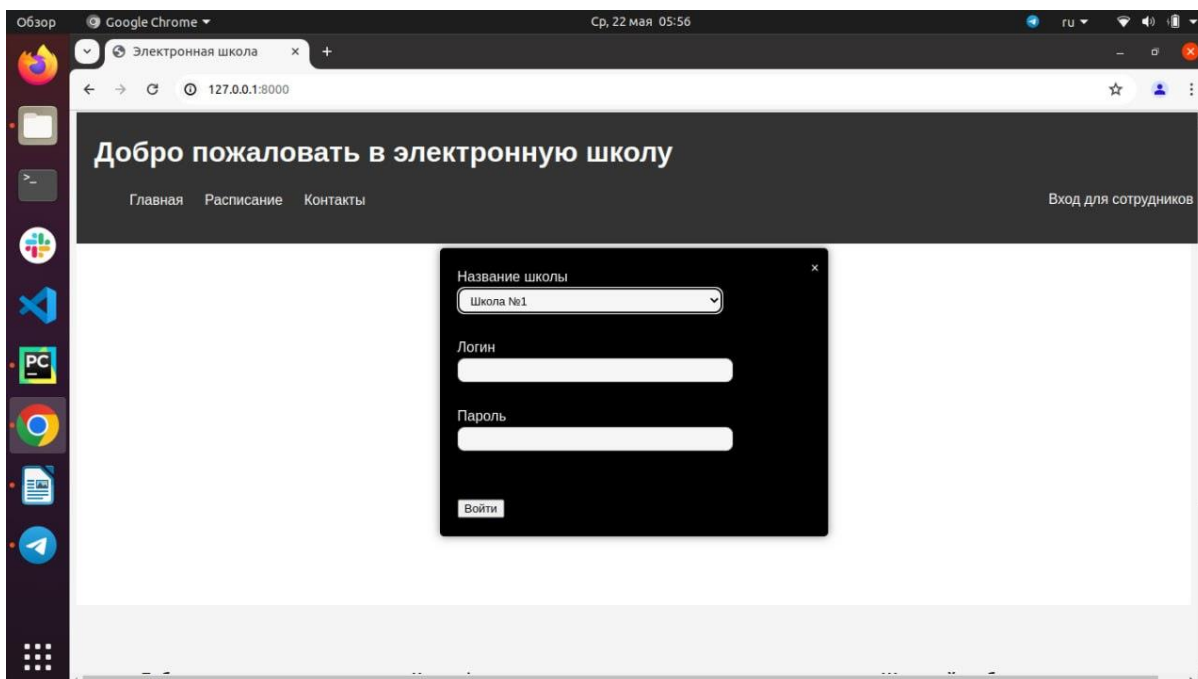


Рисунок 20 – модальное окно с формой для авторизации сотрудников

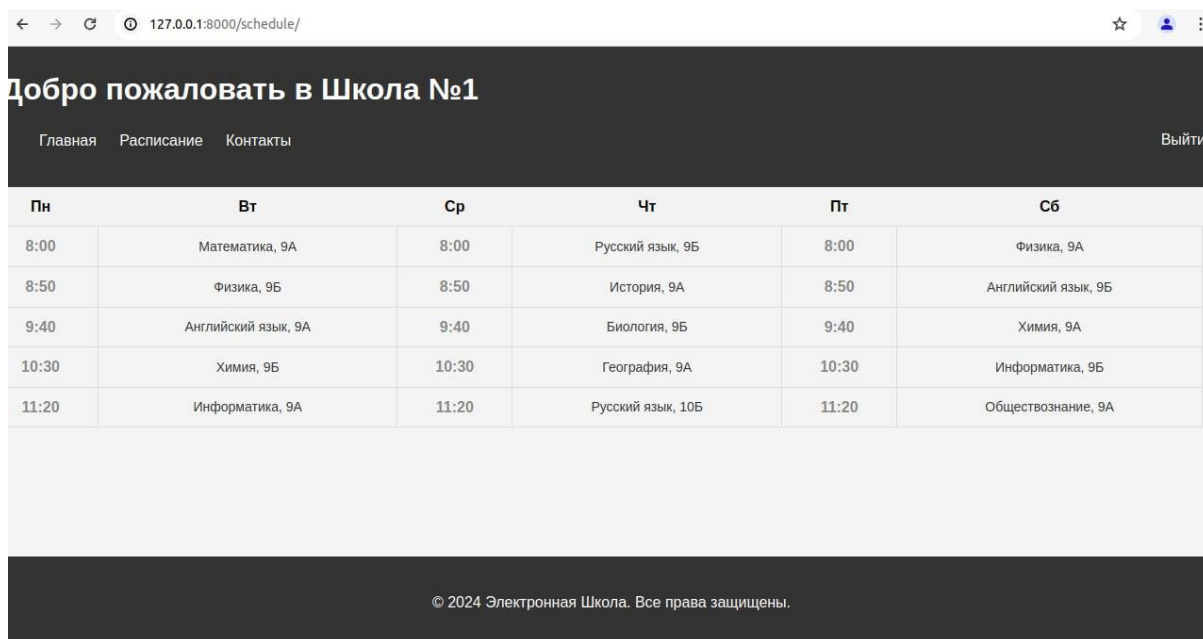
На рисунке 21 представлена страница для приема заявок учащихся на обучение в школу. Здесь отображены данные об учащихся: фамилия, имя, отчество, класс. При нажатии на кнопку «Активировать», учащимся предоставляется доступ в систему.

Добро пожаловать в Школа №1				
Главная Расписание Контакты				Выйти
Фамилия	Имя	Отчество	Класс	
Имя	Фамилия	Сергеевич	1	Активировать
Владимир	Волошин	Евгеньевич	1	Активировать
Федор	Фундуков	Алексеевич	1	Активировать
Анатолий	Соловьев	Сергеевич	1	Активировать
Сергей	Сергеевич	Сергеев	1	Активировать
Егор	Соколов	Андреевич	1	Активировать
Петр	Шилов	Сергеевич	1	Активировать

© 2024 Электронная Школа. Все права защищены.

Рисунок 21 – таблица заявок учащихся на обучение в школе

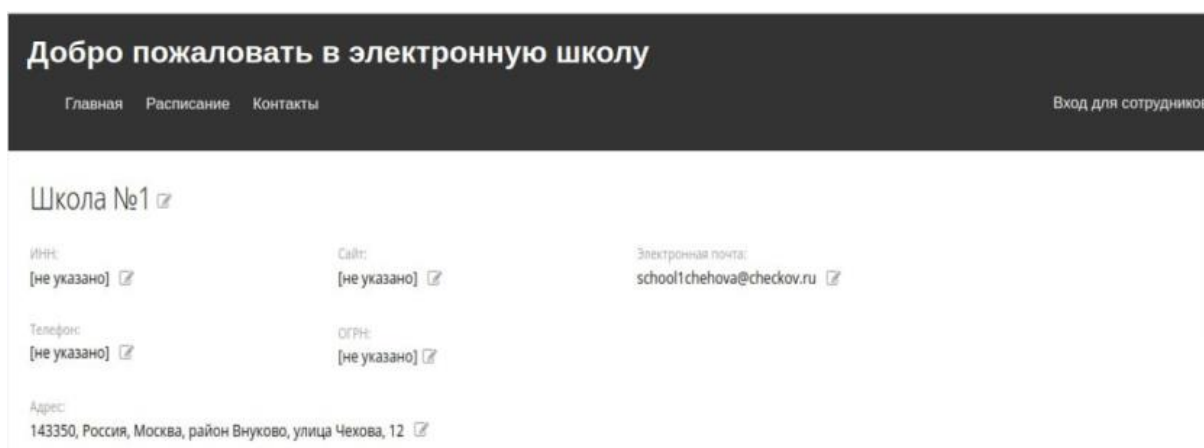
На рисунке 22 представлено расписание учащихся, где отображены: название дня недели, название предмета, класс, время начала занятия.



Пн	Вт	Ср	Чт	Пт	Сб
8:00	Математика, 9А	8:00	Русский язык, 9Б	8:00	Физика, 9А
8:50	Физика, 9Б	8:50	История, 9А	8:50	Английский язык, 9Б
9:40	Английский язык, 9А	9:40	Биология, 9Б	9:40	Химия, 9А
10:30	Химия, 9Б	10:30	География, 9А	10:30	Информатика, 9Б
11:20	Информатика, 9А	11:20	Русский язык, 10Б	11:20	Обществознание, 9А

Рисунок 22 – расписание занятий

На рисунке 23 отображена информация о школе и представлена возможность редактирования полей: название школы, ИНН, ОГРН, сайт, электронная почта, телефон, адрес.



Школа №1

ИНН: [не указано] Сайт: [не указано] Электронная почта: school1chehova@checkov.ru

Телефон: [не указано] ОГРН: [не указано]

Адрес: 143350, Россия, Москва, район Внуково, улица Чехова, 12

Рисунок 23 – карточка с информацией о школе

На рисунке 24 представлена форма создания учебного класса. На форме отображены текстовые поля «Номер класса» и «Буква класса».

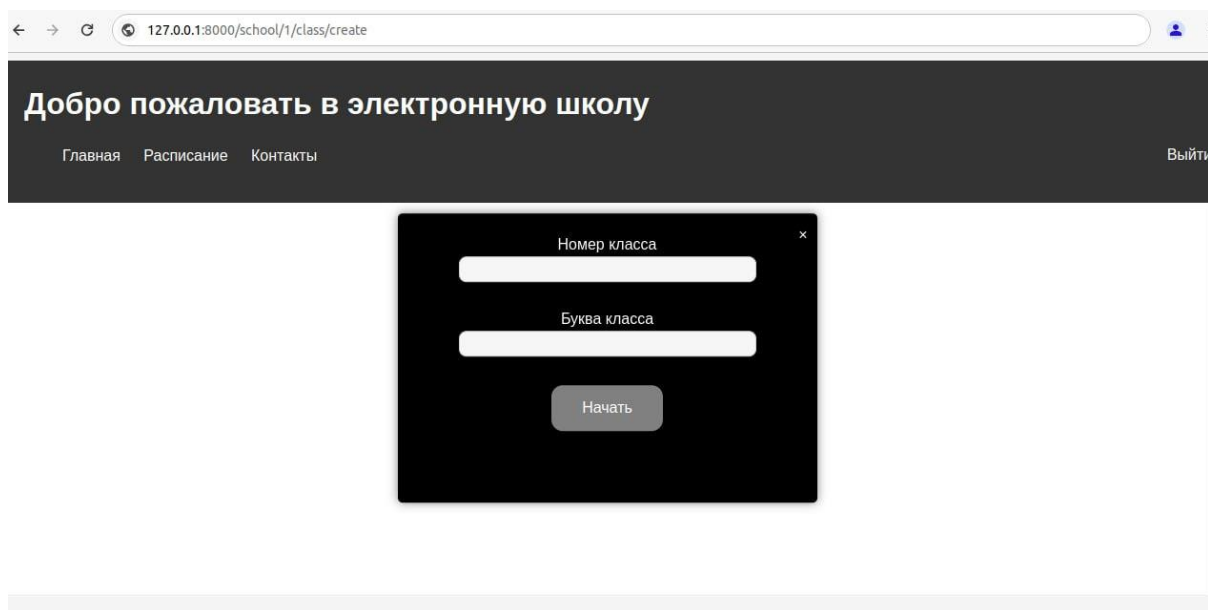


Рисунок 24 – форма создания класса

В системе имеется возможность посмотреть данные о конкретном учителе. На рисунке 25 изображена карточка учителя. Поля фамилия, имя, отчество, электронная почта, телефон, дата рождения, предмет доступны для редактирования владельцу карточки.

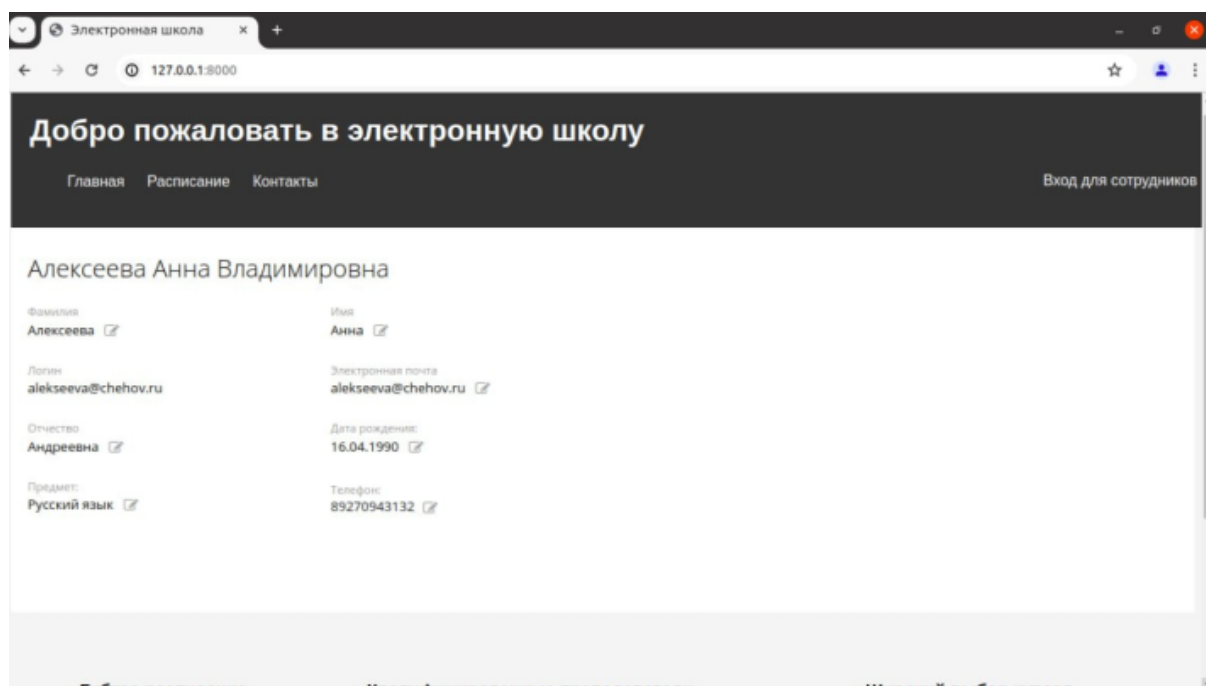


Рисунок 25 – карточка учителя

На рисунке 26 изображено модальное окно выставления оценки. Дата, предмет, ФИО ученика, класс, оценка – метки. Информация об уроках представлена на заблокированных для ввода полях, кроме поля оценки. При

нажатии на кнопку «Выставить» произойдет отправка формы и за указанный урок конкретному ученику будет присвоена оценка.

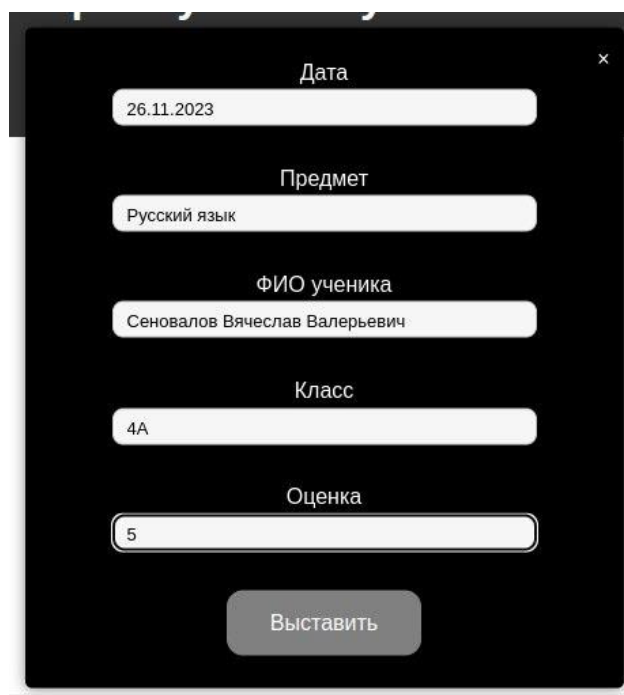
A modal window with a dark background and white text. It contains five input fields stacked vertically. The first field is labeled 'Дата' and contains '26.11.2023'. The second is labeled 'Предмет' and contains 'Русский язык'. The third is labeled 'ФИО ученика' and contains 'Сеновалов Вячеслав Валерьевич'. The fourth is labeled 'Класс' and contains '4А'. The fifth is labeled 'Оценка' and contains '5'. Below the fields is a button labeled 'Выставить'.

Рисунок 26 – модальное окно выставления оценки

На рисунке 27 представлено модальное окно для создания дисциплины. Название представляет собой поле для ввода текста.

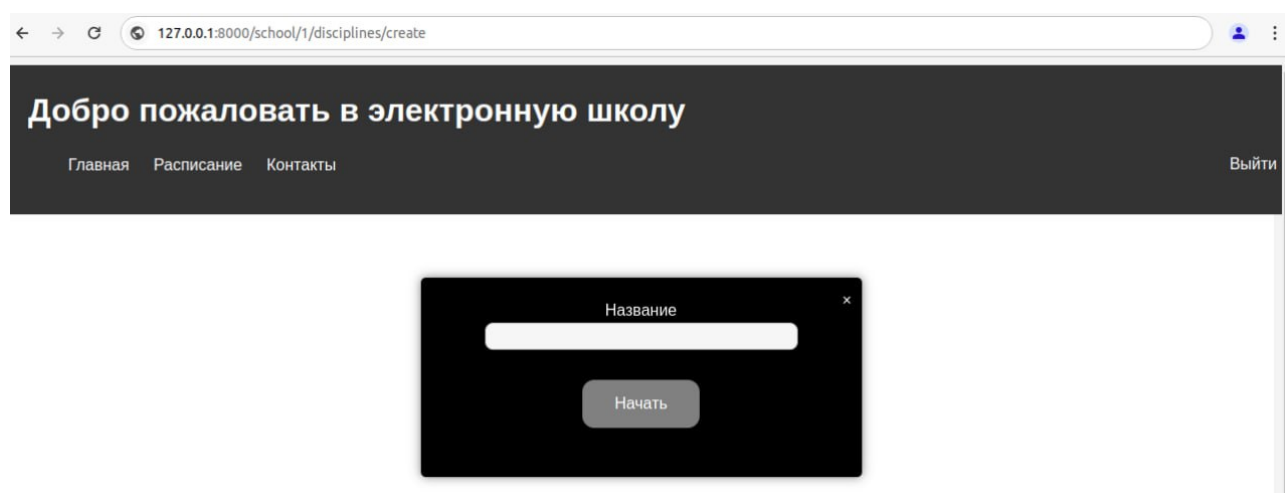
The image shows a web browser window with the address bar displaying '127.0.0.1:8000/school/1/disciplines/create'. The page has a dark header with the text 'Добро пожаловать в электронную школу' and navigation links 'Главная', 'Расписание', 'Контакты', and a 'Выйти' button. A modal window is open in the center, featuring a single text input field labeled 'Название' and a button labeled 'Начать'.

Рисунок 27 – модальное окно создания дисциплины

На рисунке 28 изображена карточка школы. Она содержит всю необходимую информацию: инн, сайт, телефон, огрн, электронную почту и адрес. Все поля на данной странице, кроме огрн и инн, редактируемые.

Школа №1

ИНН:

[не указано] 

Сайт:

[не указано] 

Электронная почта:

school1chehova@checkov.ru 

Телефон:

[не указано] 

ОГРН:

[не указано] 

Адрес:

143350, Россия, Москва, район Внуково, улица Чехова, 12 

Рисунок 28 – страница информации о школе

На рисунке 29 представлена страница контакты. Здесь потенциальный клиент может связаться и приобрести разработанный продукт. В качестве примера представлена информация о потенциальных городах клиентов, где представлен разработанное ПО, а также контакты для трудоустройства и приобретения и внедрения продукта.

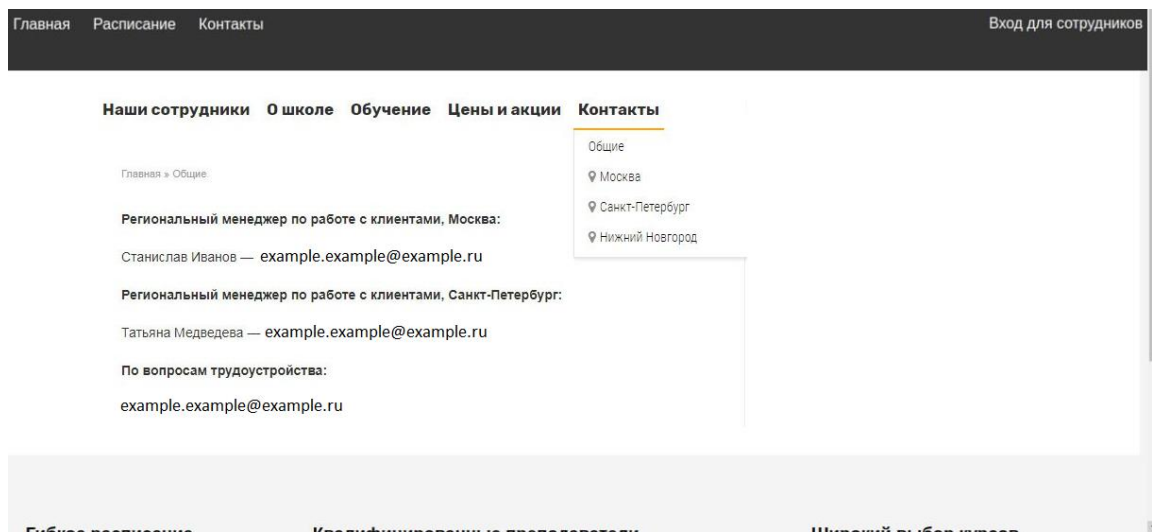


Рисунок 29 – страница контакты

Таким образом были рассмотрены основные аспекты интерфейса разрабатываемой системы. Интерфейс получился интуитивно понятным, с наличием навигации и удобными полями ввода.

3 Реализация алгоритма прогнозирования

3.1 Определение прогнозирования

В современной электронной школе важно иметь такую подсистему, которая на основе полученных данных могла бы своевременно проинформировать учителя и родителей ученика о спаде успеваемости для быстрого реагирования со стороны администрации школы. Такой подход позволит администрации школы уделять больше времени для учащихся с ухудшающимися оценками для организации индивидуальных занятий, указать родителям на важность повторного изучения школьной программы ребенка или выявить потенциальные проблемы в семье.

Прогнозирование является важнейшим инструментом принятия стратегических управленческих решений на основе анализа системы показателей: качественных (их часто называют ориентирами) и количественных (заданий). Основная задача стратегического планирования — определение путей и методов достижения поставленных целей, а целью прогноза – дать общее представление о потенциальных показателях оценок учащихся. Часто для этого достаточно нескольких ключевых показателей.

Во-первых, прогнозирование осуществляется на этапе анализа среды, перед разработкой плана. Прогнозы помогают осуществить верную постановку целей и задач, разработать правильную стратегию. В данном случае прогнозирование становится элементом планирования. Во-вторых, прогнозирование осуществляется на этапе реализации планов для оценки возможных результатов и их отклонений от плановых значений с целью организации дополнительных управляющих воздействий для ликвидации нежелательных отклонений. Такой прогноз называется оперативным. В данном случае прогнозирование становится элементом контроля. Прогнозирование – это важная функция управления, тесно связанная, в первую очередь, с планированием, во вторую – с контролем. Используемые на практике методы прогнозирования принято делить на две категории: количественные и качественные.

3.2 Количественные методы прогнозирования

Количественные методы прогнозирования — это подходы, основанные на анализе числовых данных с использованием статистических и математических моделей. Они позволяют делать предсказания о будущем, опираясь на объективную информацию и исторические данные. Рассмотрим несколько подходов.

Временные ряды — это последовательности данных, собранные или измеренные в последовательные моменты времени, часто с равными интервалами. Анализ временных рядов используется для прогнозирования будущих значений на основе прошлых наблюдений. Важными характеристиками такого метода являются тренды, сезонность, циклические колебания, случайные или нерегулярные компоненты.

Тренды — это долгосрочные движения в данных, которые показывают общее направление временного ряда. Тренд может быть восходящим, нисходящим или горизонтальным/стационарным.

Сезонность отражает повторяющиеся колебания в данных, которые происходят в определенные периоды, например, ежедневно, еженедельно, ежемесячно или ежегодно.

Циклические колебания — это волнообразные колебания вокруг тренда, которые не связаны с фиксированными календарными периодами и обычно вызваны экономическими факторами.

Случайные или нерегулярные компоненты — это непредсказуемые изменения в данных, которые не поддаются моделированию с помощью трендов, сезонности или циклов. Для анализа временных рядов используются различные методы: скользящее среднее сглаживает временной ряд, усредняя каждое значение с набором предыдущих значений, экспоненциальное сглаживание подобно скользящему среднему, но придает больший вес более недавним наблюдениям, ARIMA модель, которая объединяет авторегрессию, интегрирование для устранения трендов и скользящее среднее.

Применение моделей временных рядов требует тщательного анализа данных, проверки на стационарность и выбора подходящей модели. Это мощный инструмент для прогнозирования во многих областях, от финансов до метеорологии и маркетинга. Однако следует помнить, что прогнозы основаны на предположении, что прошлые закономерности будут продолжаться в будущем, что не всегда может быть точным.

Эконометрические модели - это статистические модели, используемые в эконометрии для анализа экономических данных и тестирования гипотез о экономических взаимосвязях. Они позволяют исследователям количественно оценивать экономические теории, прогнозировать будущие события и разрабатывать экономическую политику. Основные типы эконометрических моделей включают одновременные уравнения, где несколько уравнений моделируются вместе для учета взаимосвязей между переменными; модели временных рядов, которые анализируют данные, собранные через равные промежутки времени; и панельные данные, объединяющие временные ряды и поперечные данные. Методы оценки в эконометрике включают метод наименьших квадратов, максимальное правдоподобие и инструментальные переменные. Эконометрические модели используются для тестирования теорий, оценки эффектов экономической политики, прогнозирования экономических показателей и многое другое. Они могут быть линейными или нелинейными, статическими или динамическими, и могут включать различные типы данных и переменных. Важно правильно специфицировать модель, чтобы избежать ошибок спецификации, и провести тщательную проверку на соответствие предположениям модели. Эконометрические модели могут быть очень сложными и требуют глубоких знаний в области статистики и экономики для их корректного применения и интерпретации результатов.

Машинное обучение предоставляет множество моделей для прогнозирования, которые можно использовать в различных областях, от финансов до медицины. Основные категории включают обучение с учителем, без учителя и с подкреплением. В обучении с учителем используются такие модели, как линейная регрессия, логистическая регрессия, машины опорных векторов, нейронные сети и ансамблевые методы, включая случайный лес и бустинг. Модели без учителя, такие как кластеризация К-средних и метод главных компонент, ищут скрытые структуры в данных без явных меток. Модели с подкреплением, такие как Q-обучение и политика градиента, используются для оптимизации последовательности действий для достижения цели. Методы машинного обучения могут быть применены к временным рядам для прогнозирования будущих событий, используя прошлые данные. Глубокое обучение, подкатегория машинного обучения, использует сложные нейронные сети с множеством слоев для моделирования сложных паттернов в данных. Важно отметить, что выбор модели зависит от специфики задачи, качества и количества данных, а также от того, насколько хорошо модель может интерпретироваться и объясняться. Модели машинного обучения требуют тщательной предварительной обработки данных, включая нормализацию, обработку пропущенных значений и кодирование категориальных переменных. Также критически важно оценить модель, используя такие методы, как кросс-валидация и A/B-тестирование, чтобы убедиться, что модель обобщает и не переобучается. Машинное обучение продолжает развиваться, и появляются новые методы и алгоритмы, которые улучшают способность моделей прогнозировать и анализировать данные.

3.3 Качественные методы прогнозирования

Качественные модели прогнозирования основаны на экспертных оценках и неформализованных данных, они используются там, где количественные данные недоступны или неполны. Такие модели включают метод Дельфи, сценарное планирование, анализ влияния перекрестных факторов и мозговой

штурм. Метод Дельфи включает серию раундов опросов экспертов с последующим анализом их мнений для достижения консенсуса.

Сценарное планирование разрабатывает различные будущие сценарии на основе текущих тенденций и возможных будущих событий. Анализ влияния перекрестных факторов исследует взаимосвязи между различными элементами и их потенциальное влияние на будущее. Мозговой штурм используется для генерации идей и решений в групповой дискуссии. Эти методы полагаются на интуицию, опыт и творческий подход к прогнозированию, они часто применяются в стратегическом планировании и управлении рисками.

Анализ влияния перекрестных факторов, также известный как факторный анализ или ANOVA, исследует влияние двух или более категориальных независимых переменных на зависимую переменную. Он позволяет оценить не только основные эффекты каждого фактора, но и взаимодействие между факторами, чтобы понять, как комбинация факторов влияет на результат. Это делается путем сравнения средних значений групп, определенных каждым уровнем факторов, и анализа дисперсии, чтобы определить, есть ли статистически значимые различия между этими средними значениями. Если взаимодействие между факторами значимо, это указывает на то, что эффект одного фактора зависит от уровня другого фактора. Этот метод широко используется в экспериментальных исследованиях для определения оптимальных условий или для оценки влияния различных переменных на интересующий исход.

Мозговой штурм в прогнозировании — это творческий метод коллективной работы, при котором участники генерируют идеи и решения для предсказания будущих событий или тенденций. Принцип работы заключается в создании открытой и непредвзятой атмосферы, где каждый может свободно высказывать свои мысли и предложения без страха критики. Участники поочередно предлагают идеи, которые затем обсуждаются и развиваются группой, что позволяет расширить перспективы и учитывать различные точки

зрения. Это способствует выявлению нестандартных подходов и может привести к более точным и всесторонним прогнозам. Мозговой штурм используется для стимулирования креативности и инноваций, а также для улучшения качества прогнозов путем коллективного обсуждения и анализа возможных будущих сценариев.

Качественные модели могут быть полезны для понимания широких тенденций и формирования общего направления будущих изменений, но они не предоставляют точных численных прогнозов и могут быть субъективными из-за зависимости от мнений экспертов. Важно сочетать качественные и количественные подходы для более полного и сбалансированного прогнозирования.

3.4 Линейная регрессия

Проанализировав методы прогнозирования, был сделан вывод: в разрабатываемом ПО необходимо придерживаться комплексного подхода, который будет сочетать в себе как количественные способы прогнозирования, так и качественные.

Из категории качественного подхода будет использоваться метод Дельфи, в котором в качестве эксперта будет выступать учитель. Он будет учитывать полноту усвоения материала и выставять учащимся балльную оценку. Из количественной категории прогнозирования был выбран способ линейной регрессии. Линейная регрессия была выбрана не просто так. Безусловно, использование машинного обучения прекрасно подходит для решения поставленной задачи. Этот способ популярен и эффективен в мире, однако требует достаточно большого времени на обучение, огромной выборки данных и самого современного оборудования, что в условиях школы не всегда достижимо.

Линейная регрессия уместна в случаях, когда между переменными существует предполагаемая линейная зависимость. Это простой и понятный метод, который легко интерпретировать и использовать для прогнозирования, особенно в рамках учебного процесса. Очень часто благополучие и полнота

семей прямо зависит на успехи ребенка в школе, не стоит и не брать во внимание образование родителей. Линейная регрессия хорошо работает на больших наборах данных и требует относительно малого количества вычислительных ресурсов, что прекрасно подходит под поставленную задачу. В рамках изучаемой темы были выведены несколько переменных влияние, оценку которых относительно нетрудно вычислить и вывести результат в другой переменной. Также линейная регрессия эффективна в ситуациях, где данные распределены нормально и нет сильной мультиколлинеарности. Однако важно помнить, что линейная регрессия может давать неточные результаты, если истинная связь между переменными нелинейна или если данные содержат выбросы.

Использование регрессионной модели для прогнозирования состоит в подстановке в уравнение регрессии ожидаемых значений факторных признаков для расчета точечного прогноза результативного признака или его доверительного интервала с заданной вероятностью. У регрессионных моделей есть следующие перспективные направления использования: использование многовариантных значений внутренних и внешних факторов при разработке различных сценариев развития (в качестве внешних факторов по отношению к району могут использоваться показатели в целом по региону или стране). Простейшей системой корреляционной связи является линейная связь между двумя признаками – парная линейная корреляция. Многофакторная система требует уже не одного, а множества показателей тесноты связей, имеющих разный смысл и применение. Общий вид многофакторного уравнения регрессии следующий:

Коэффициент корреляции может принимать значения $-1 \leq r \leq 1$; по абсолютной величине $0 \leq |r| \leq 1$. Отрицательные значения свидетельствуют об обратной связи признаков y и x , положительные – о прямой связи. Обычно считают связь сильной, если $r \geq 0,7$, средней при $0,5 \leq r \leq 0,7$; слабой – при $r \leq 0,5$. Основой измерения связей является матрица парных коэффициентов корреляции. По этой матрице можно судить о тесноте связи факторов с

результативным признаком и между собой. Хотя все эти показатели относятся к парным связям, все же матрицу можно использовать для предварительного отбора факторов для включения их в уравнение регрессии. Для прогнозирования оценок по конкретному предмету будет использоваться метод линейной регрессии (Linear Regression) и его модификации.

Алгоритм линейной регрессии выглядит следующим образом (1):

$$a(x) = w_0 + \sum w_j x_j, \quad (1)$$

где w_0 - свободный коэффициент,

x_j - признаки,

w_j - их веса.

Функция потерь для линейной модели часто выбирается в виде квадрата отклонения - $(a(x) - y)$.

Функционал качества, который называется среднеквадратичной ошибкой алгоритма, задается следующим образом (2):

$$Q(a, Xl) = \frac{1}{2} \sum (a(x_i) - y_i)^2, \quad (2)$$

Так как необходимо минимизировать функционал качества для обучения модели, нужно воспользоваться численными методами оптимизации, а именно методом градиентного спуска. С помощью данного метода будут подобраны оптимальные веса для алгоритма. В случае, когда модель получается слишком сложной и недостаточно данных для точного определения параметров, модель может легко переобучиться. Решить эту проблему можно либо упростив модель, либо использовав регуляризацию – ограничение значений весов модели.

Существуют такие типы регуляризации как:

– L_2 -регуляризатор (ridge-regression): $Q(a, Xl) = \frac{1}{2} \sum (a(x_i) - y_i)^2 + \lambda \sum w_j^2 \rightarrow \min_w$,

– L_1 -регуляризатор (lasso-regression): $Q(a, Xl) = \frac{1}{2} \sum (a(x_i) - y_i)^2 + \lambda \sum |w_j| \rightarrow \min_w$.

При использовании $L2$ -регуляризации значения весов сглаживаются, а при использовании $L1$ -регуляризации веса признаков, обладающих низкой предсказательной способностью, откидываются, т.к. они становятся равны 0. Включим информацию о родителях в описанную выше регрессионную модель. Это может стать полезным способом улучшения точности прогнозов. Добавим следующие параметры:

- полнота семьи. Данный параметр может быть бинарным (1 - если ребенок живет в полной семье, 0 - если нет). Это может помочь учесть социальные и эмоциональные факторы, влияющие на успеваемость ученика,
- оценки родителей. Если доступны данные об образовании родителей, можно использовать их как количественный показатель (например, средний балл аттестата о среднем образовании или диплома о высшем образовании). Если такие данные недоступны, можно использовать категориальные переменные, такие как уровень образования (например, среднее, среднее специальное, высшее).

Для выполнения вычислений были загружены средние оценки ученика за конкретный период, полнота семьи и средние оценки родителей. Данные представлены в таблице 11.

Таблица 11 – таблица исходных данных

Номер учащегося	Полнота семьи	Образование матери	Образование отца	Средний балл
1	1	4	3	4
2	0	5	2	3
3	1	3	5	5
4	0	3	4	3
5	1	2	2	2
6	0	4	4	4
7	1	5	5	5
8	0	3	3	3

9	1	4	4	4
10	0	2	3	3

Более наглядно оценить получившиеся результаты регрессии можно в таблицах 12 – 15

Таблица 12 – определение значений показателей

Показатель	Значение
Зависимая переменная	Средний балл
R-квадрат	1.000
Скорректированный R-квадрат	1.000
F-статистика	5.950e+29
Prob (F-статистика)	8.31e-89
Log-Likelihood	322.31
Количество наблюдений	10
AIC	-636.6

Анализ полученных данных представлен в таблице 12.

Таблица 13– результат анализа исходных данных

Коэффициенты	Значение	Статистическая ошибка	T - статистика	P > t	[0.025, 0.975]
Балл ученика	3	3.94e-15	0	3	3
Образование матери	1	1.68e-15	5.96e+14	0	1
Образование отца	1	5.274e-16	2.13e-15	-4.69e-15	5.74e-15
Полнота семьи	1	3.31e-15	3.03e+14	0	1

Диагностика значений представлена в таблице 14

Таблица 14 – результаты регрессии

Диагностика	Значение
Omnibus	1.847
Prob(Omnibus)	0.397
Jarque-Bera (JB)	0.847
Prob(JB)	0.655
Skew	-0.215
Kurtosis	1.640
Cond. No.	23.9

Результаты регрессии OLS показывают, что модель идеально подходит для данных, так как R-квадрат и скорректированный R-квадрат равны 1.000, что указывает на то, что модель объясняет 100% вариации зависимой переменной "Средний балл".

F-статистика очень высока (порядка 5.950×10^{29}), и вероятность (Prob (F-statistic)) практически равна нулю, что говорит о том, что модель статистически значима. Коэффициенты для переменных "Полнота семьи" и "Образование матери" равны 1.0000, что означает, что изменение этих переменных на одну единицу приведет к изменению "Среднего балла" на одну единицу. Коэффициент для "Образование отца" статистически не значим, так как его p-значение (0.813) выше общепринятого порога 0.05, и его интервалы доверия включают ноль.

Omnibus, Durbin-Watson, Jarque-Bera и другие показатели говорят о том, что остатки модели распределены нормально, не имеют автокорреляции и соответствуют предположениям линейной регрессии.

ЗАКЛЮЧЕНИЕ

В результате выполнения магистерской диссертации был проведен анализ предметной области, и спроектировано программное обеспечение для создания автоматизированной системы учета успеваемости учащихся. В данной работе были рассмотрены основные принципы организации учебного процесса и контроля качества знаний учащихся с использованием классической базовой системы, создан удобный пользовательский интерфейс, который облегчит и ускорит рутинную работу учителя, а также позволит родителю наблюдать за успеваемостью ребенка. Рассмотрены недостатки традиционных административных процессов обучения и предложены варианты устранения имеющихся недостатков в разработанном ПО.

В процессе выполнения магистерской диссертации были реализованы следующие цели: изучены особенности организации и управления данными в информационных системах, создана возможность в разработанном программном продукте удобного вывода и манипулирования данными для формирования диаграмм и обработки статистической информации, спроектирован пользовательский web-интерфейс, реализована физическая модель базы данных с помощью языка запросов SQL, создана возможность удобного и интуитивно понятного использования системы, а также подробно изучены способы прогнозирования и проведен сравнительный анализ современных методов количественных и качественных моделей. Реализованное ПО позволит не только упростить рутинную работу сотрудников школы, но и поможет улучшить качество образования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Арженовский С.В. Статистические методы прогнозирования: Учебное пособие / С.В. Арженовский, И.Н. Молчанов. Ростов-н/Д.: Рост. гос. экон. ун-т. «РИНХ», 2001.
2. Бейдер. Д., Эймос Д., Яблонски Д., Хейслер Ф. Знакомство с Python, 2023.
3. Бхаргава А. Грожаем алгоритмы, Питер, 2017.
4. Mat Marquis. Javascript For Web Designers, БХВ, 2016.
5. Канцедал С. А. Основы статистики, ИД Форум, 2023.

Приложение А

(Обязательное)

Система хранения и обработки информации

Листинг программы

Файл school/migrations

```
import datetime
import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models

class Migration(migrations.Migration):

    initial = True

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]

    operations = [
        migrations.CreateModel(
            name='School',
            fields=[
                ('id', models.BigAutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=30)),
                ('ogrn', models.CharField(max_length=13)),
                ('inn', models.CharField(max_length=10)),
                ('kpp', models.CharField(max_length=9)),
                ('address', models.CharField(max_length=40)),
                ('site', models.CharField(max_length=25,
unique=True)),
                ('phone', models.CharField(max_length=11,
unique=True)),
                ('email', models.EmailField(max_length=254,
unique=True)),
                ('created_at',
models.DateTimeField(default=datetime.datetime.utcnow)),
                ('update_at',
models.DateTimeField(default=datetime.datetime.utcnow)),
            ],
            options={
                'verbose_name': 'Школа',
                'verbose_name_plural': 'Школы',
                'db_table': 'schools',
            },
        ),
        migrations.CreateModel(
            name='Teacher',
            fields=[
                ('id', models.BigAutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('first_name', models.CharField(max_length=30)),
                ('last_name', models.CharField(max_length=30)),
```

```

        ('patronymic', models.CharField(blank=True,
max_length=30)),
        ('subject_taught',
models.CharField(max_length=30)),
        ('created_at',
models.DateTimeField(default=datetime.datetime.utcnow)),
        ('update_at',
models.DateTimeField(default=datetime.datetime.utcnow)),
        ('school',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='schools.school')),
        ('user',
models.OneToOneField(on_delete=django.db.models.deletion.CASCADE
, to=settings.AUTH_USER_MODEL)),
    ],
    options={
        'verbose_name': 'Учитель',
        'verbose_name_plural': 'Учителя',
        'db_table': 'teachers',
    },
),
migrations.CreateModel(
    name='Student',
    fields=[
        ('id', models.BigAutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
        ('first_name', models.CharField(max_length=30)),
        ('last_name', models.CharField(max_length=30)),
        ('patronymic', models.CharField(blank=True,
max_length=30)),
        ('created_at',
models.DateTimeField(default=datetime.datetime.utcnow)),
        ('update_at',
models.DateTimeField(default=datetime.datetime.utcnow)),
        ('school',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='schools.school')),
        ('user',
models.OneToOneField(on_delete=django.db.models.deletion.CASCADE
, to=settings.AUTH_USER_MODEL)),
    ],
    options={
        'verbose_name': 'Ученик',
        'verbose_name_plural': 'Ученики',
        'db_table': 'students',
    },
),
migrations.CreateModel(
    name='SchoolClass',
    fields=[
        ('number', models.IntegerField()),
        ('letter', models.CharField(max_length=1)),

```

```

        ('teacher',
models.OneToOneField(on_delete=django.db.models.deletion.CASCADE
, primary_key=True, serialize=False, to='schools.teacher')),
        ('created_at',
models.DateTimeField(default=datetime.datetime.utcnow)),
        ('update_at',
models.DateTimeField(default=datetime.datetime.utcnow)),

, [15.06.2024 7:26]
('school',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='schools.school')),
    ],
    options={
        'verbose_name': 'Класс',
        'verbose_name_plural': 'Классы',
        'db_table': 'classes',
    },
),
]

```

Файл classes/views.py

```

from django.http import JsonResponse
from django.views import View
from .models import SchoolClass
from django.views.decorators.csrf import csrf_exempt
from django.utils.decorators import method_decorator
from django.core.exceptions import ObjectDoesNotExist

@method_decorator(csrf_exempt, name='dispatch')
class SchoolClassView(View):
    def get(self, request, *args, **kwargs):
        classes = list(SchoolClass.objects.values())
        return JsonResponse(classes, safe=False)

    def post(self, request, *args, **kwargs):
        # Здесь должен быть код для создания нового класса
        pass

    def put(self, request, class_id, *args, **kwargs):
        # Здесь должен быть код для обновления существующего
        # класса по class_id
        pass

    def delete(self, request, class_id, *args, **kwargs):
        try:
            school_class = SchoolClass.objects.get(id=class_id)
            school_class.delete()
            return JsonResponse({'message': 'Class deleted
successfully'}, status=200)
        except ObjectDoesNotExist:

```

```
        return JsonResponse({'message': 'Class not found'},
status=404)
```

Файл models.py

```
import datetime

from django.contrib.auth.models import User
from django.db import models

class School(models.Model):
    name = models.CharField(max_length=30,
verbose_name='Название')
    ogrn = models.CharField(max_length=13, verbose_name='ОГРН')
    inn = models.CharField(max_length=10, verbose_name='ИНН')
    kpp = models.CharField(max_length=9, verbose_name='КПП')
    address = models.CharField(max_length=40,
verbose_name='Адрес')
    site = models.CharField(max_length=25, unique=True,
verbose_name='Сайт')
    phone = models.CharField(max_length=11, unique=True,
verbose_name='Телефон')
    email = models.EmailField(unique=True, verbose_name='Эл.
почта')
    created_at =
models.DateTimeField(default=datetime.datetime.utcnow,
verbose_name='Создано')
    update_at =
models.DateTimeField(default=datetime.datetime.utcnow,
verbose_name='Обновлено')

    def str(self):
        return self.name

    class Meta:
        verbose_name_plural = 'Школы'
        verbose_name = 'Школа'
        db_table = 'schools'

class Student(models.Model):
    first_name = models.CharField(max_length=30,
verbose_name='Имя')
    last_name = models.CharField(max_length=30,
verbose_name='Фамилия')
    patronymic = models.CharField(max_length=30, blank=True,
verbose_name='Отчество')
    school = models.ForeignKey(School, on_delete=models.CASCADE,
verbose_name='ID школы')
    # date_of_birth = models.DateField()
    user = models.OneToOneField(User, on_delete=models.CASCADE,
verbose_name='ID пользователя')
```

```

        created_at =
models.DateTimeField(default=datetime.datetime.utcnow,
verbose_name='Создано')
        update_at =
models.DateTimeField(default=datetime.datetime.utcnow,
verbose_name='Обновлено')

        class Meta:
            verbose_name_plural = 'Ученики'
            verbose_name = 'Ученик'
            db_table = 'students'

class Teacher(models.Model):
    first_name = models.CharField(max_length=30,
verbose_name='Имя')
    last_name = models.CharField(max_length=30,
verbose_name='Фамилия')
    patronymic = models.CharField(max_length=30, blank=True,
verbose_name='Отчество')
    subject_taught = models.CharField(max_length=30,
verbose_name='Предмет')
    user = models.OneToOneField(User, on_delete=models.CASCADE,
verbose_name='ID пользователя')
    school = models.ForeignKey(School, on_delete=models.CASCADE,
verbose_name='ID школы')
    created_at =
models.DateTimeField(default=datetime.datetime.utcnow,
verbose_name='Создано')
    update_at =
models.DateTimeField(default=datetime.datetime.utcnow,
verbose_name='Обновлено')

    class Meta:
        verbose_name_plural = 'Учителя'
        verbose_name = 'Учитель'
        db_table = 'teachers'

class SchoolClass(models.Model):
    number = models.IntegerField(blank=False, null=False,
verbose_name='Номер класса')
    letter = models.CharField(max_length=1,
verbose_name='Буква')
    teacher = models.OneToOneField(Teacher,
on_delete=models.CASCADE, primary_key=True, verbose_name='ID
учителя')
    school = models.ForeignKey(School, on_delete=models.CASCADE,
verbose_name='ID школы')
    student = models.ForeignKey(Student,
on_delete=models.CASCADE, verbose_name='Класс')

```

```

        created_at =
models.DateTimeField(default=datetime.datetime.utcnow,
verbose_name='Создано')
        update_at =
models.DateTimeField(default=datetime.datetime.utcnow,
verbose_name='Обновлено')

class Meta:
    verbose_name_plural = 'Классы'
    verbose_name = 'Класс'
    db_table = 'classes'

```

Файл teachers/views.py

```

from django.http import JsonResponse
from django.views import View
from .models import Teacher
from django.views.decorators.csrf import csrf_exempt
from django.utils.decorators import method_decorator
from django.core.exceptions import ObjectDoesNotExist
import json

@method_decorator(csrf_exempt, name='dispatch')
class TeacherView(View):
    def get(self, request, *args, **kwargs):
        teachers = list(Teacher.objects.values())
        return JsonResponse(teachers, safe=False)

    def post(self, request, *args, **kwargs):
        try:
            data = json.loads(request.body)
            teacher = Teacher.objects.create(**data)
            return JsonResponse({'message': 'Teacher created successfully', 'teacher': teacher.id}, status=201)
        except Exception as e:
            return JsonResponse({'message': str(e)}, status=400)

    def put(self, request, teacher_id, *args, **kwargs):
        try:
            data = json.loads(request.body)
            teacher = Teacher.objects.get(id=teacher_id)
            for key, value in data.items():
                setattr(teacher, key, value)
            teacher.save()
            return JsonResponse({'message': 'Teacher updated successfully'}, status=200)
        except ObjectDoesNotExist:
            return JsonResponse({'message': 'Teacher not found'}, status=404)
        except Exception as e:
            return JsonResponse({'message': str(e)}, status=400)

    def delete(self, request, teacher_id, *args, **kwargs):

```

```

    try:
        teacher = Teacher.objects.get(id=teacher_id)
        teacher.delete()
        return JsonResponse({'message': 'Teacher deleted
successfully'}, status=200)
    except ObjectDoesNotExist:
        return JsonResponse({'message': 'Teacher not
found'}, status=404)

```

Файл manage.py

```

#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'schools.settings')
    try:
        from django.core.management import
execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed
and "
            "available on your PYTHONPATH environment variable?
Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if name == 'main':
    main()

```

Файл teachers/views.py

```

from django.contrib.auth import authenticate, login
from django.contrib.auth.models import User
from django.shortcuts import render
from django.views import View
from django.views.generic.edit import CreateView

from schools.models import Teacher
from teachers.forms import LoginForm

def login_teacher(request):
    form = LoginForm(request.POST)
    if form.is_valid():

```

```

        user =
User.objects.filter(username=form.data['username'])
        if user.exists():
            log_user =
authenticate(username=form.data['username'],
password=form.data['password'])
            if log_user:
                login(request, log_user)
            school =
Teacher.objects.get(user__username=form.data['username']).school
            return render(request, 'schools/home_page.html',
{'school': school})
        return render(request, 'schools/home_page.html')

```

Файл school/helper.py

```

class SchoolSystemHelper:
    @staticmethod
    def validate_teacher_data(data):
        # Проверка данных учителя перед созданием или
        # обновлением
        required_fields = ['name', 'subject', 'email']
        for field in required_fields:
            if field not in data:
                return False, f'Отсутствует обязательное поле:
{field}'
        return True, 'Данные валидны'

    @staticmethod
    def format_class_list(classes):
        # Форматирование списка классов для вывода
        return '\n'.join([f'Класс: {cls["name"]}, Учитель:
{cls["teacher"]}' for cls in classes])

# Пример использования хэлпера
data = {
    'name': 'Иван Иванов',
    'subject': 'Математика',
    'email': 'ivanov@example.com'
}

is_valid, message =
SchoolSystemHelper.validate_teacher_data(data)
if is_valid:
    print('Создание учителя...')
    # Логика создания учителя
else:
    print(message)

classes = [
    {'name': '5А', 'teacher': 'Иван Иванов'},
    {'name': '6Б', 'teacher': 'Мария Петрова'}
]

```

```
formatted_classes =
SchoolSystemHelper.format_class_list(classes)
```

Файл teachers/forms.py

```
from django import forms

from schools.models import School

class LoginForm(forms.Form):
    school = forms.ModelChoiceField(
        label='Название школы',
        queryset=School.objects.all(),
        to_field_name='name',
        widget=forms.Select(attrs={'id': 'school_name'})
    )
    username = forms.CharField(max_length=30, min_length=5,
required=True, label='Логин')
    password = forms.CharField(max_length=30, min_length=5,
required=True, label='Пароль', widget=forms.PasswordInput())
```

Файл students/views.py

```
from django.shortcuts import render, redirect
# from django.views.decorators.clickjacking import
xframe_options_deny

from schools.models import Student, School
from students.actions import StudentsAggregator
from students.exceptions import SchoolNotDefinedError,
UserExistsError
from students.forms import StudentForm

# @xframe_options_deny
def registration(request):
    if request.method == 'POST':
        form = StudentForm(data=request.POST)
        if form.is_valid():
            try:
                StudentsAggregator(data=request.POST).create()
            except SchoolNotDefinedError as error:
                return render(request, 'schools/home_page.html',
{error: error})
            except UserExistsError as error:
                render(request, 'schools/home_page.html',
{error: error})
            return render(request, 'schools/home_page.html', {})
        else:
            return render (request, 'schools/home_page.html',
{})
```

```

else:
    return render(request, 'schools/home_page.html', {})

def activation(request):
    # students =
    Student.objects.filter(school=request.user.teacher.school,
user__is_active=True)
    school = School.objects.first()
    students = Student.objects.filter(user__is_active=True)
    # school = request.user.teacher.school
    return render(request, 'students/activation.html',
{'students': students, 'school': school})

def activate(request):
    student = Student.objects.get(id=request.POST['id'])
    student.user.update(active=True)
    return redirect('/students/activation')

def card(request):
    return render(request, 'students/card.html')

```

Файл students/forms.py

```

from django import forms

from schools.models import School

class StudentForm(forms.Form):
    first_name = forms.CharField(label='Имя')
    last_name = forms.CharField(label='Фамилия')
    patronymic = forms.CharField(label='Отчество',
required=False)
    school = forms.ModelChoiceField(
        label='Название школы',
        queryset=School.objects.all(),
        to_field_name='name',
        widget=forms.Select(attrs={'id': 'school_name'})
    )
    classname = forms.CharField(label='Класс')
    username = forms.CharField(label='Логин')
    password = forms.CharField(label='Пароль',
widget=forms.PasswordInput())

```

Файл students/actions.py

```

from django.contrib.auth.hashers import make_password
from django.contrib.auth.models import User

```

```

from students.exceptions import SchoolNotDefinedError,
UserExistsError
from schools.models import Student, School

class StudentsAggregator:
    def init(self, data):
        self.data = data

    def create(self):
        school =
School.objects.filter(name=self.data['schoolName'])
        if school.exists():
            user_exists = User.objects.filter(
                username=self.data['username']
            ).exists()
            if user_exists:
                raise UserExistsError('Пользователь уже
существует')
            user = User.objects.create(
                username=self.data['username'],
                password=make_password(self.data['password']),
                first_name=self.data['firstName'],
                last_name=self.data['lastName'],
                active=False,
            )
            Student.objects.create(
                first_name=self.data['firstName'],
                last_name=self.data['lastName'],
                patronymic=self.data['middleName'],
                user_id=user.id,
                school_id=school.get().id
            )
        else:
            raise SchoolNotDefinedError('Указанная школа не
существует')

```

Файл school/urls.py

```

from django.urls import path
from .views import SchoolView, TeacherView, StudentView,
ClassView, SubjectView, ScheduleView, AttendanceView, GradeView,
EventView, LibraryView, CafeteriaView, SchoolBusView

urlpatterns = [
    # Школьные ручки
    path('schools/', SchoolView.as_view(), name='schools-list'),
    path('schools/<int:school_id>/', SchoolView.as_view(),
name='school-detail'),

    # Учителя
    path('teachers/', TeacherView.as_view(), name='teachers-
list'),

```

```

    path('teachers/<int:teacher_id>/', TeacherView.as_view(),
name='teacher-detail'),

    # Ученики
    path('students/', StudentView.as_view(), name='students-
list'),
    path('students/<int:student_id>/', StudentView.as_view(),
name='student-detail'),

    # Классы
    path('classes/', ClassView.as_view(), name='classes-list'),
    path('classes/<int:class_id>/', ClassView.as_view(),
name='class-detail'),

    # Предметы
    path('subjects/', SubjectView.as_view(), name='subjects-
list'),
    path('subjects/<int:subject_id>/', SubjectView.as_view(),
name='subject-detail'),

    path('schedule/', ScheduleView.as_view(), name='schedule-
list'),
    path('schedule/<int:schedule_id>/', ScheduleView.as_view(),
name='schedule-detail'),

    path('attendance/', AttendanceView.as_view(),
name='attendance-list'),
    path('attendance/<int:attendance_id>/',
AttendanceView.as_view(), name='attendance-detail'),

    path('grades/', GradeView.as_view(), name='grades-list'),
    path('grades/<int:grade_id>/', GradeView.as_view(),
name='grade-detail'),

    path('events/', EventView.as_view(), name='events-list'),
    path('events/<int:event_id>/', EventView.as_view(),
name='event-detail')

    path('library/', LibraryView.as_view(), name='library-
list'),
    path('library/<int:book_id>/', LibraryView.as_view(),
name='book-detail'),

    path('cafeteria/', CafeteriaView.as_view(), name='cafeteria-
list'),
    path('cafeteria/<int:menu_id>/', CafeteriaView.as_view(),
name='menu-detail'),

    path('schoolbus/', SchoolBusView.as_view(), name='schoolbus-
list'),

```

```

        path('schoolbus/<int:bus_id>/', SchoolBusView.as_view(),
name='bus-detail'),
]

```

Файл calculator/helper.py

```

import statsmodels.api as sm
import pandas as pd

# Создаем DataFrame с данными
data = {
    data
}
df = pd.DataFrame(data)

# Определяем зависимую и независимые переменные
X = df[['Полнота семьи', 'Образование матери', 'Образование
отца']]
y = df['Средний балл']

# Добавляем константу к независимым переменным
X = sm.add_constant(X)

# Строим модель
model = sm.OLS(y, X).fit()

# Выводим результаты
print(model.summary())

```

Файл disciplines/views.py

```

from django.http import JsonResponse
from django.views import View
from .models import Subject
from django.core.exceptions import ObjectDoesNotExist
from django.views.decorators.csrf import csrf_exempt
from django.utils.decorators import method_decorator

@method_decorator(csrf_exempt, name='dispatch')
class SubjectView(View):
    def get(self, request, subject_id=None):
        # Получение информации о дисциплине или списка дисциплин
        if subject_id:
            try:
                subject = Subject.objects.get(id=subject_id)
                return JsonResponse({'id': subject.id, 'name':
subject.name}, safe=False)
            except ObjectDoesNotExist:
                return JsonResponse({'error': 'Дисциплина не
найдена'}, status=404)
        else:
            subjects = list(Subject.objects.values())
            return JsonResponse(subjects, safe=False)

```

```

def post(self, request):
    # Создание новой дисциплины
    from django.http import QueryDict
    data = QueryDict(request.body)
    subject = Subject(name=data.get('name'))
    subject.save()
    return JsonResponse({'id': subject.id, 'name':
subject.name}, status=201)

def put(self, request, subject_id):
    # Обновление информации о дисциплине
    from django.http import QueryDict
    data = QueryDict(request.body)
    try:
        subject = Subject.objects.get(id=subject_id)
        subject.name = data.get('name', subject.name)
        subject.save()
        return JsonResponse({'id': subject.id, 'name':
subject.name})
    except ObjectDoesNotExist:
        return JsonResponse({'error': 'Дисциплина не
найдена'}, status=404)

def delete(self, request, subject_id):
    # Удаление дисциплины
    try:
        subject = Subject.objects.get(id=subject_id)
        subject.delete()
        return JsonResponse({'message': 'Дисциплина
удалена'})
    except ObjectDoesNotExist:
        return JsonResponse({'error': 'Дисциплина не
найдена'}, status=404)

```

Файл schedules/views.py

```

# views.py

from django.http import JsonResponse
from django.views import View
from .models import Schedule
from django.core.exceptions import ObjectDoesNotExist
from django.views.decorators.csrf import csrf_exempt
from django.utils.decorators import method_decorator

@method_decorator(csrf_exempt, name='dispatch')
class ScheduleView(View):
    def get(self, request, class_id=None):
        # Получение расписания для класса
        if class_id:
            try:

```

```

        schedule =
Schedule.objects.filter(class_id=class_id).values()
        return JsonResponse(list(schedule), safe=False,
status=200)
    except ObjectDoesNotExist:
        return JsonResponse({'error': 'Расписание не
найдено'}, status=404)
    else:
        # Получение общего расписания для всех классов
        all_schedules = Schedule.objects.all().values()
        return JsonResponse(list(all_schedules), safe=False,
status=200)

# urls.py

from django.urls import path
from .views import ScheduleView

urlpatterns = [
    # Другие URL-адреса...
    path('schedule/', ScheduleView.as_view(), name='schedule-
list'),
    path('schedule/<int:class_id>', ScheduleView.as_view(),
name='class-schedule'),
    # Другие URL-адреса...
]
```

Приложение Б

(Обязательное)

Система хранения и обработки информации

Презентация

Система хранения и обработки информации

Автор работы: студент группы 22ВВИМ1 Махров А.А.

Научный руководитель: к.т.н., доцент кафедры ВТ Слепцов Н.В.

Актуальность проекта

- Адаптация школьного журнала к современным технологиям
- Автоматизация процесса учета экономит время и снижает вероятность ошибок
- Доступность информации
- Экологичность
- Безопасность данных

Цели работы

- 1 Разработка программного обеспечения системы контроля успеваемости учащихся
- 2 Программа должна поддерживать работу разных типов пользователей
- 3 Реализовать прогнозирование успеваемости ученика на будущее время
- 4 Создать систему, которая будет хранить в себе не только информацию о ученике, но и его родителей

Анализ системных требований

- 1.Производительность:** Система должна обеспечивать быструю обработку данных и отклик на запросы пользователей, даже при большом объеме информации.
- 2.Надежность:** Система должна быть устойчивой к сбоям и обеспечивать сохранность данных в случае аварий.
- 3.Масштабируемость:** Способность системы адаптироваться к увеличению числа пользователей и объема данных без потери производительности.
- 4.Безопасность:** Защита данных от несанкционированного доступа, включая механизмы аутентификации, авторизации и шифрования.
- 5.Совместимость:** Возможность работы системы с различными операционными системами и устройствами.
- 6. Интероперабельность:** Способность системы взаимодействовать с другими программными продуктами, используемыми в учебном заведении.
- 7. Обновляемость:** Простота внесения изменений и добавления новых функций в систему.
- 8. Пользовательский интерфейс:** Интерфейс должен быть понятным и удобным для всех категорий пользователей.
- 9. Документация:** Наличие подробной и понятной документации для пользователей и разработчиков.

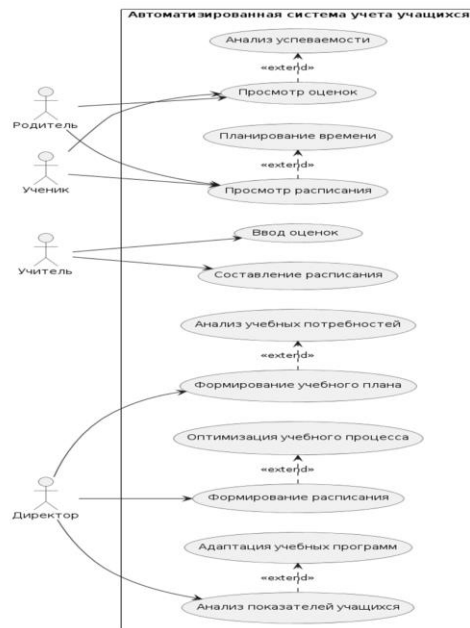
Стек технологий



Типы пользователей

Тип пользователя	Возможности
Администратор	Регистрирует ответственное лицо(директора), оказывает техническую поддержку по возникшим вопросам.
Директор	Формирует списки преподавателей, классов, учащихся. Связывает преподавателя с определенной дисциплиной. Призывает к классам преподавателей.
Учитель	Выставляет оценки учащимся, просматривает списки классов, добавляет объявления для класса. Добавляет домашнее задание учащимся.
Ученик	Просматривает успеваемость (выставленные оценки), просматривает статистику успеваемости.

Диаграмма вариантов использования



Проектирование физической модели бд

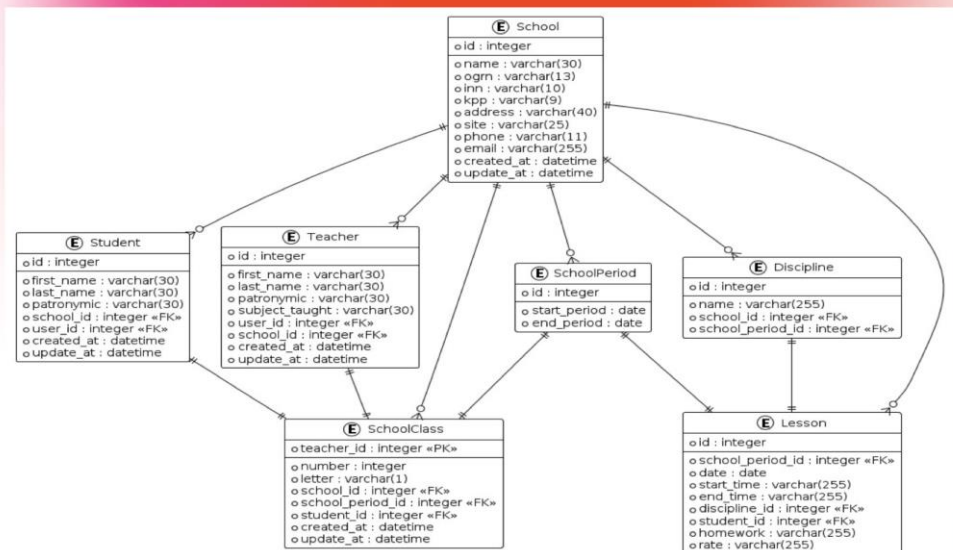


Диаграмма коммуникаций

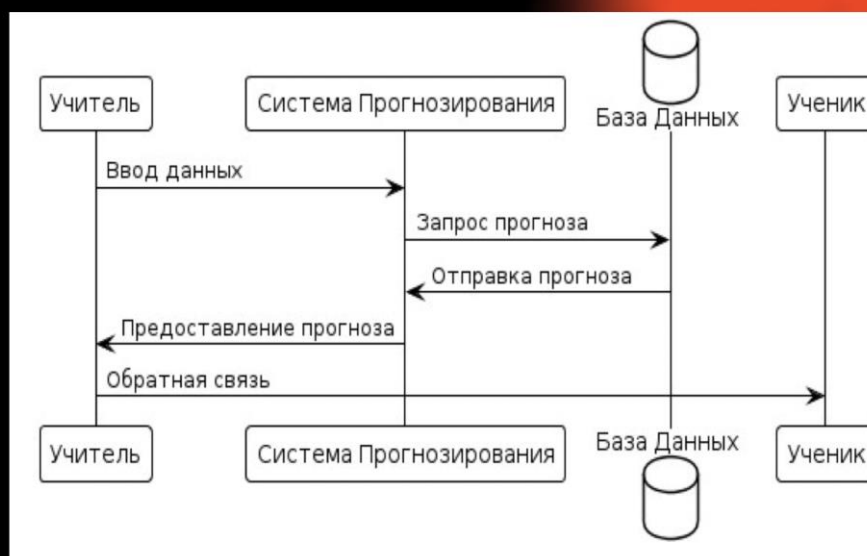
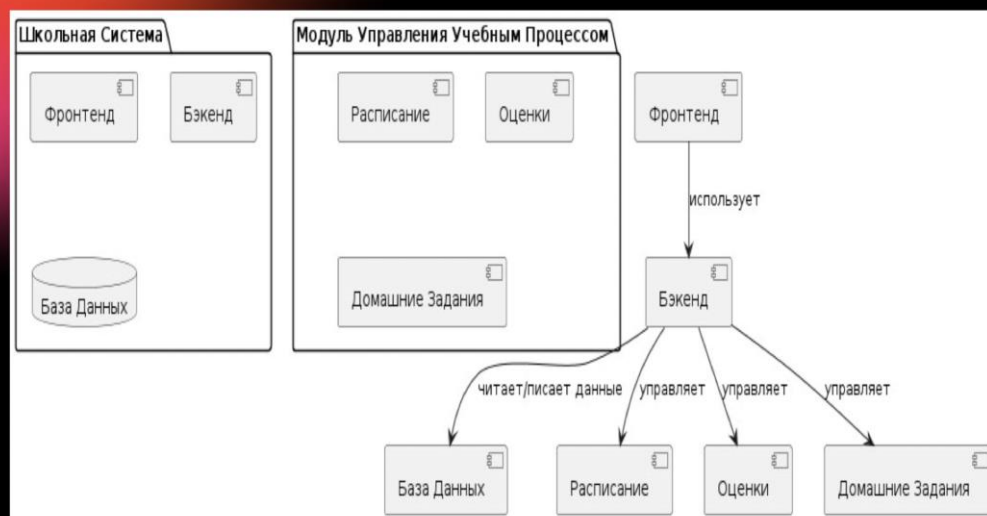


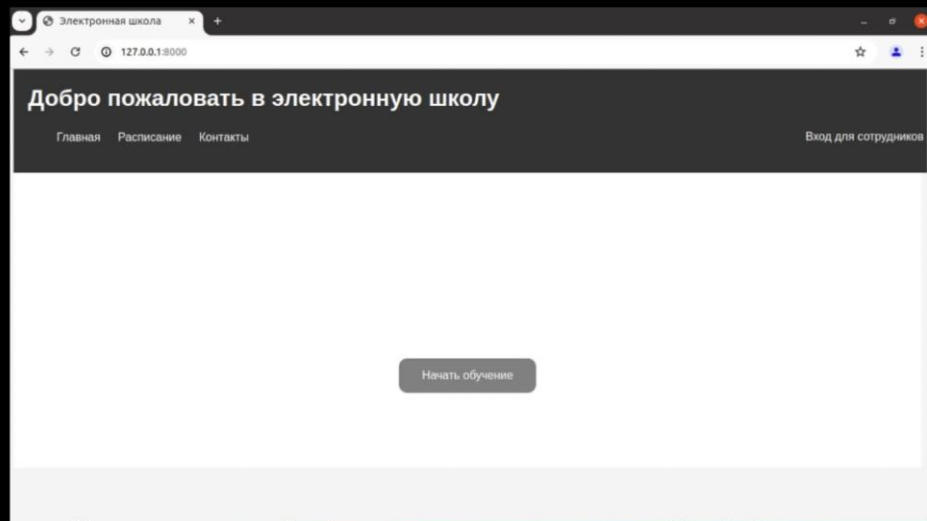
Диаграмма последовательности выставления оценки в электронный журнал



Диаграмма компонентов системы



Главная страница сайта



Форма регистрации

Добро пожаловать в эле

Главная Расписание Контакты

Вход для сотрудников

Имя: Alexsava

Фамилия: Игорь

Отчество: Викторович

Название школы: Школа №1

Класс: 4А

Логин: Alexsava

Пароль: [masked]

Начать

Форма активации учеников

Добро пожаловать в Школа №1

Главная Расписание Контакты

Выйти

Фамилия	Имя	Отчество	Класс	
Имя	Фамилия	Сергеевич	1	Активировать
Владимир	Волошин	Евгеньевич	1	Активировать
Федор	Фундуков	Алексевич	1	Активировать
Анатолий	Соловьев	Сергеевич	1	Активировать
Сергей	Сергеевич	Сергеев	1	Активировать
Егор	Соколов	Андреевич	1	Активировать
Петр	Шилов	Сергеевич	1	Активировать

© 2024 Электронная Школа. Все права защищены.

Форма входа для сотрудников

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000'. The page title is 'Добро пожаловать в электронную школу'. Below the title are navigation links: 'Главная', 'Расписание', and 'Контакты'. In the top right corner, there is a link 'Вход для сотрудников'. A modal login form is centered on the screen. It contains a dropdown menu for 'Название школы' with 'Школа №1' selected, input fields for 'Логин' and 'Пароль', and a 'Войти' button.

Страница расписания

The screenshot shows the 'Расписание' page. The header includes 'Добро пожаловать в Школа №1' and navigation links 'Главная', 'Расписание', and 'Контакты'. A 'Выйти' link is in the top right. The main content is a table with 6 columns representing days of the week (Пн, Вт, Ср, Чт, Пт, Сб) and 6 rows representing time slots (8:00, 8:50, 9:40, 10:30, 11:20). The table lists subjects and class numbers for each slot. A footer contains the copyright notice: '© 2024 Электронная Школа. Все права защищены.'

Пн	Вт	Ср	Чт	Пт	Сб
8:00	Математика, 9А	8:00	Русский язык, 9Б	8:00	Физика, 9А
8:50	Физика, 9Б	8:50	История, 9А	8:50	Английский язык, 9Б
9:40	Английский язык, 9А	9:40	Биология, 9Б	9:40	Химия, 9А
10:30	Химия, 9Б	10:30	География, 9А	10:30	Информатика, 9Б
11:20	Информатика, 9А	11:20	Русский язык, 10Б	11:20	Обществознание, 9А

Количественные методы:

- **Временные ряды:** Анализ исторических данных для выявления тенденций, циклов, сезонности и шума.
- **Экстраполяция:** Продление существующих тенденций в будущее.
- **Каузальные модели:** Использование статистических методов для определения взаимосвязей между различными переменными и их влиянием на прогнозируемую величину.

Качественные методы:

Делфи-метод: Серия раундов опросов экспертов с последующим анализом их мнений.

Метод сценариев: Разработка различных сценариев будущего на основе текущих данных и тенденций.

Интуитивные методы: Основаны на личном опыте и интуиции экспертов.

Прогнозирование

Номер учащегося	Полнота семьи	Образование матери	Образование отца	Средний балл
1	1	4	3	4
2	0	5	2	3
3	1	3	5	5
4	0	3	4	3
5	1	2	2	2
6	0	4	4	4
7	1	5	5	5
8	0	3	3	3
9	1	4	4	4
10	0	2	3	3

Прогнозирование

Показатель	Значение
Зависимая переменная	Средний балл
R-квадрат	1.000
Скорректированный R-квадрат	1.000
F-статистика	5.950e+29
Prob (F-статистика)	8.31e-89
Log-Likelihood	322.31
Количество наблюдений	10
AIC	-636.6

Анализ полученных результатов

Диагностика	Значение
Omnibus	1.847
Prob(Omnibus)	0.397
Jarque-Bera (JB)	0.847
Prob(JB)	0.655
Skew	-0.215
Kurtosis	1.640
Cond. No.	23.9

Заключение