МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ПЛАНИРОВАНИЯ УЧЕБНОЙ НАГРУЗКИ ПРЕПОДАВАТЕЛЕЙ

Студент _____ *31.05.2022* Милованов А.С.
(подпись, дата)

Руководитель _____ *31.05.2022* Карамышева Н.С.
(подпись, дата)

Нормоконтролёр _____ *31.05.2022* А.С. Бычков
(подпись, дата)

Работа допущена к защите (протокол заседания кафедры от 02.06.2022 г. №13)

Заведующий кафедрой _____ М.А. Митрохин
(подпись)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 7.06.22 № 2)

Секретарь ГЭК _____ Е.И. Гурин
(подпись)

Пенза 2022

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» апреля 2022г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Милованов Антон Сергеевич гр.18ВВ2 факультета вычислительной техники.

2.Тема работы: «Автоматизированная система планирования учебной нагрузки преподавателей»

Тема утверждена приказом ПГУ №357/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы к.т.н., доцент, каф. ВТ ПГУ Карамышева Н.С.

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

Разработать приложение «Автоматизированная система планирования учебной нагрузки преподавателей», позволяющее автоматизировать ручной процесс планирования учебной нагрузки преподавателей. Приложение должно обладать графическим и интуитивно понятным интерфейсом, для взаимодействия с пользователем. Система должна предоставлять возможность для взаимодействия с информацией о преподавателях, учебных группах, нормах времени. Разрабатываемое приложение также должно выполнять задачи по формированию электронных таблиц.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить существующие автоматизированные системы, выделить их достоинства и недостатки. Выбрать язык программирования, который подойдет для создания графического интерфейса. Определить архитектуру приложения и вспомогательные средства разработки. Изучить функционал библиотек для взаимодействия с файлами формата *xlsx*.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	26.04.2022 – 28.05.2022	
2	Анализ требований	32	29.04.2022 – 30.05.2022	
3	Создание макета графического интерфейса	30	1.04.2022 – 3.05.2022	
4	Выбор технологий, применяемых при проектировании и разработке	40	4.05.2022 – 8.05.2022	
5	Реализация графического интерфейса	66	9.05.2022 – 15.05.2022	
6	Разработка основной части программы	70	16.05.2022 – 23.05.2022	
7	Оформление пояснительной записки	54	24.05.2022 – 31.05.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы _____ 26.04.2022 Карамышева Н.С.
(подпись, дата)

Задание к исполнению принял студент _____ 26.04.2022 Милованов А.С.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 26.05.2022 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 116 листов, 38 рисунков 1 таблицу, 7 источников, 31 приложение.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА, РАСЧЕТ НАГРУЗКИ ПРЕПОДАВАТЕЛЕЙ УНИВЕРСИТЕТА, MVVM, PRISM, WPF.

Цель работы – разработать систему, позволяющую автоматизировать процесс планирования учебной нагрузки преподавателей.

Объектом разработки является приложение импортирующее информацию из расчетного листа кафедральной нагрузки и экспортирующие преподавательскую нагрузку в файл формата xlsx. Приложение должно обладать графическим пользовательским интерфейсом.

Разработка проводилась в среде программирования Visual Studio 2019. В качестве языка программирования был выбран C#.

В результате проведенной работы разработано приложение, удовлетворяющее поставленной задаче, а именно: удобный пользовательский интерфейс, возможность взаимодействия с информацией о преподавателях и учебных группах, автоматическое генерирование таблиц.

Созданное приложение может быть внедрено в Пензенский государственный университет для автоматизации планирования преподавательской нагрузки.

Содержание

Перечень сокращений и обозначений 5
Введение 6
1 Автоматизированная система планирования учебной нагрузки преподавателей 7
1.1 Постановка задачи 7
1.2 Анализ существующих аналогов 10
1.2.1 Автоматизированная система «НАГРУЗКА ВУЗА» 10
1.2.2 Автоматизированная система на базе платформы «1С» 12
1.2.3 Подсистема «Педагогическая нагрузка» 14
1.3 Выбор языка программирования 15
1.3.1 Язык программирования *C++* 15
1.3.2 Язык программирования *Java* 15
1.3.3 Язык программирования *Python* 16
1.3.4 Язык программирования *C#* 16
1.4 Выводы 17
1.5 Выбор инструментов разработки 17
1.5.1 Выбор среды разработки 17
1.5.2 Выбор платформы для реализации графического интерфейса 17
1.5.3 Выбор вспомогательного фреймворка 19
1.5.4 Выбор библиотеки для взаимодействия с таблицами *Excel* 21
1.6 Архитектура приложения 21
1.6.1 Паттерн *MVVM* 21
1.6.2 Архитектура 22
1.7 Разработка приложения 23
1.7.1 Разработка графического интерфейса 23
1.7.1.1 Создание стилей *WPF* 25
1.7.1.2 Диалоговые окна 26
1.7.2 Разработка моделей 29

1.7.2.1 Класс *Teacher* 29
1.7.2.2 Класс *Group* 30
1.7.2.3 Класс *CalculationSheet* 30
1.7.2.4 Класс *CalculationSheetDiscipline* 31
1.7.2.5 Класс *TeacherLoadDiscipline* 31
1.7.2.6 Отношения между объектами классов 32
1.8 Описание приложения 33
1.8.1 Руководство пользователя 33
2 Охрана труда 43
2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения 43
2.1.1 Оборудование рабочего места 43
2.1.2 Освещенность рабочего места 45
2.1.3 Пожарная безопасность 47
2.1.4 Режим труда и отдыха 48
Заключение 50
Список используемых источников 51
Приложение А – UML – диграммы 52
Приложение Б – Листинг программы 58
Приложение В – Презентация 96

Перечень сокращений и обозначений

WPF (*Windows Presentation Foundation*) – система для построения клиентских приложений *Windows* с визуальными возможностями взаимодействия с пользователем

MVVM (*Model View ViewModel*) – шаблон проектирования архитектуры приложения

XLSX – формат файлов табличного типа

UML (*Unified Modeling Language*) – унифицированный язык моделирования

ГОСТ – региональный стандарт, принятый Межгосударственным советом по стандартизации, метрологии и сертификации Содружества Независимых Государств

ПЭВМ – персональная электронно-вычислительная машина

ПК – персональный компьютер

XAML – декларативный язык разметки

.*NET* – кроссплатформенная платформа для создания различных типов приложений

Фреймворк – программная платформа, которая упрощает разработку программного продукта, определяет структуру проекта

АС – автоматизированная система

Введение

В настоящее время трудно представить нашу жизнь без информационных технологий. Вряд ли можно найти организацию, которая в той или иной степени не была бы оснащена средствами приёма, хранения, передачи, обработки информации.

Информация является одним из основных ресурсов роста производительности. Одним из видов программного обеспечения являются автоматизированные системы. Автоматизированная система состоит из компонентов, предназначенных для выполнения набора задач, которые были предварительно настроены. Функциональные и утомительные задачи становятся менее обременительными и делают рабочий процесс проще и легче.

Целью выпускной работы является разработка автоматизированной системы для планирования учебной нагрузки преподавателей.

В ходе данной работы необходимо разработать приложение для автоматизации процесса планирования преподавательской нагрузки. Приложение должно иметь удобный графический интерфейс, для визуализации всей необходимой информации для планирования нагрузки. Входным файлом является расчётный лист кафедральной нагрузки в формате *xlsx*, из него необходимо импортировать в приложение всю информацию об учебных дисциплинах. Результатом работы приложения является файл, содержащий в себе таблицы с информацией о нагрузке всех преподавателей.

1 Автоматизированная система планирования учебной нагрузки преподавателей

Актуальность данной темы, прежде всего, обусловлена отсутствием собственной автоматизированной системы расчета учебной нагрузки преподавателей Пензенского государственного университета.

Так как расчеты производятся вручную, присутствует большая вероятность появления различных ошибок расчета, за которыми необходимо следить и исправлять их.

Большинство аналогов данной системы могут не подходить для использования по различным параметрам.

Некоторые системы не могут работать с форматом подписного расчетного листа кафедральной нагрузки нашего университета. Данные расчетные листы хранятся в файлах формата *xlsx*, из-за этого импорт данных в систему может быть некорректен или невозможен. А результат по расчету учебной нагрузке преподавателей тоже должен быть экспортирован в определенном формате.

1.1 Постановка задачи

Требуется разработать приложение, которое должно советовать следующим целям:

– автоматизирование процесса распределения учебной нагрузки между преподавателями кафедры, в соответствии с нормами времени для расчета объема учебной работы;

– возможность импорта информации из файлов формата *xlsx*, и экспорта информации в файлы формата *xlsx*. Формат *xlsx* используется для хранения электронных таблиц;

– удобный и интуитивно понятный графический интерфейс для работы пользователя с приложением;

– расположение графических элементов должно предоставлять пользователю быстрое ориентирование в интерфейсе;

– возможность масштабирования интерфейса в процессе работы;

В процессе разработки должны быть решены следующие задачи:

– импорт данных в приложение из расчетных листов кафедральной нагрузки, которые хранятся в файлах в формате *xlsx*. Формат данных представлен на рисунке 1.1;

– отображение импортированной информации из расчетного листа кафедральной нагрузки.

– экспорт учебной нагрузки в отдельный файл в формате *xlsx*, в этот файл экспортируется информация об итоговой учебной нагрузке для всех преподавателей и учебная нагрузка для конкретного преподавателя в отдельный лист. Результат экспорта должен советовать таблицам, представленным на рисунках 1.2, 1.3.

– автоматическая расстановка основных расчетных формул в советующие ячейки таблицы;

– автоматическое изменение стилей отдельных ячеек.

– наличие всплывающих подсказок у графических элементов.

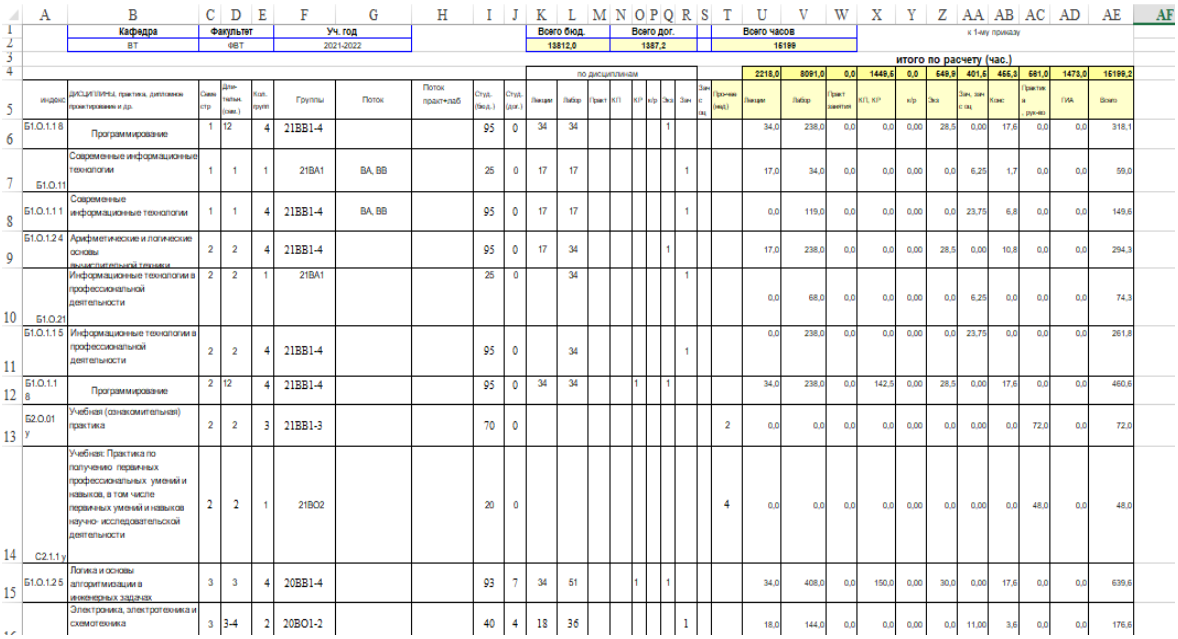
– добавление, изменение, хранение основных данных о преподавателях;

– добавление, изменение, хранение основных данных об учебных группах;

– блокировать некорректный ввод данных пользователем;

– возможность ручной корректировки значений расчетов пользователем;

– возможность изменения значений норм времени, в случае их изменений, так как автоматизированный расчет учебной нагрузки преподавателей происходит в соответствии с ними.



Кафедра	Факультет	Уч. год	Всего бюд.	Всего дог.	Всего часов
ВТ	ФВТ	2021-2022	13812,0	1587,2	16199

к 1-ому приказу

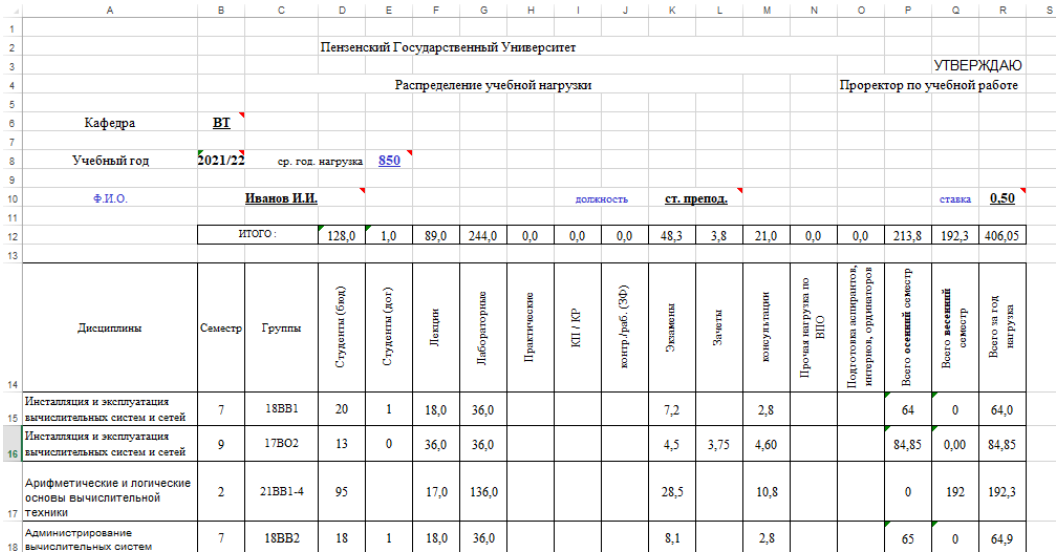
итого по расчету (час.)

Индекс	Дисциплины, практики, дипломное проектирование и др.	Семестр	Для-тельн. (сем.)	Кол. групп	Группы	Поток	Поток практ+лаб	Студ. (бюд.)	Студ. (дог.)	по дисциплинам: Лекции	Лабор	Практ	КП	КР	к/р	Экз	Зач	Зач. с оц.	Трудоемкость (ед.)	Лекции	Лабор	Практ занятия	КП, КР	к/р	Экз	Зач. зач с оц.	Конс	Практика и руководство	ГИА	Всего
																				2218,0	8091,0	0,0	1449,5	0,0	549,5	401,5	455,5	681,0	1473,0	16199,2
Б1.О.1.18	Программирование	1	12	4	21ВВ1-4			95	0	34	34					1				34,0	238,0	0,0	0,0	0,00	28,5	0,00	17,6	0,0	0,0	318,1
Б1.О.11	Современные информационные технологии	1	1	1	21ВА1	ВА, ВВ		25	0	17	17						1			17,0	34,0	0,0	0,0	0,00	0,0	6,25	1,7	0,0	0,0	59,0
Б1.О.1.11	Современные информационные технологии	1	1	4	21ВВ1-4	ВА, ВВ		95	0	17	17						1			0,0	119,0	0,0	0,0	0,00	0,0	23,75	6,8	0,0	0,0	149,6
Б1.О.1.24	Арифметические и логические основы вычислительной техники	2	2	4	21ВВ1-4			95	0	17	34					1				17,0	238,0	0,0	0,0	0,00	28,5	0,00	10,8	0,0	0,0	294,3
Б1.О.21	Информационные технологии в профессиональной деятельности	2	2	1	21ВА1			25	0		34						1			0,0	68,0	0,0	0,0	0,00	0,0	6,25	0,0	0,0	0,0	74,3
Б1.О.1.15	Информационные технологии в профессиональной деятельности	2	2	4	21ВВ1-4			95	0		34						1			0,0	238,0	0,0	0,0	0,00	0,0	23,75	0,0	0,0	0,0	261,8
Б1.О.1.18	Программирование	2	12	4	21ВВ1-4			95	0	34	34			1		1				34,0	238,0	0,0	142,5	0,00	28,5	0,00	17,6	0,0	0,0	460,6
Б2.О.01 у	Учебная (ознакомительная) практика	2	2	3	21ВВ1-3			70	0										2	0,0	0,0	0,0	0,0	0,00	0,0	0,00	0,0	72,0	0,0	72,0
С2.1.1 у	Учебная: Практика по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков научно- исследовательской деятельности	2	2	1	21ВО2			20	0										4	0,0	0,0	0,0	0,0	0,00	0,0	0,00	0,0	48,0	0,0	48,0
Б1.О.1.25	Логика и основы алгоритмизации в инженерных задачах	3	3	4	20ВВ1-4			93	7	34	51			1		1				34,0	408,0	0,0	150,0	0,00	30,0	0,00	17,6	0,0	0,0	639,6
	Электроника, электротехника и схемотехника	3	3-4	2	20ВО1-2			40	4	18	36						1			18,0	144,0	0,0	0,0	0,00	0,0	11,00	3,6	0,0	0,0	176,6

Рисунок 1.1 – Входные данные

	B	C	D	E	F	G	H	I	J	K
1										
2			Пензенский Государственный Университет							
3										
4										
5										
6	ВТ									
7										
8	2021/22		850							
9										
10	ставки по приказу	1	3,75	8,25	2,00	0,00	2,00	15,25	12707,50	
11	ставки по факту	1,00	3,75	8,25	1,00		2,50	16,00	12651,40	2057,75
12	Ф.И.О.	зав.каф	проф.	доц.	ст. преп.	преп.	ассистент	ставки	учебная нагрузка	лекционная нагрузка
13	Бикташев Р.А.			0,75				0,75	640,90	142,00
14	Бычков А.С.			1,00				1,00	860,30	158,00
15	Гурин Е.И.		1,00					1,00	850,05	191,00
16	Дорошенко И.Н.			0,25				0,25	216,05	68,00
17	Дубинин В.Н.		1,00					1,00	770,60	193,00
18	Дубравин А.В.			0,25				0,25	221,30	68,00
19	Егоров В.Ю.			0,25				0,25	223,85	68,00
20	Зинкин С.А.		1,00					1,00	847,40	198,00
21	Калиниченко Е.И.				0,50			0,50	406,05	89,00
22	Карамышева Н.С.			1,00				1,00	826,35	70,00
23	Коннова Л.Г.						0,50	0,50	154,00	0,00

Рисунок 1.2 – Итоговый лист



Пензенский Государственный Университет

УТВЕРЖДАЮ

Распределение учебной нагрузки

Проректор по учебной работе

Кафедра ВТ

Учебный год 2021/22 ср. год. нагрузка 850

Ф.И.О. Иванов И.И. должность ст. препод. ставка 0,50

ИТОГО :			128,0	1,0	89,0	244,0	0,0	0,0	0,0	48,3	3,8	21,0	0,0	0,0	213,8	192,3	406,05
Дисциплины	Семестр	Группы	Студенты (бюдж)	Студенты (дог)	Лекции	Лабораторные	Практические	КП / КР	контр./раб. (ЗФ)	Экзамены	Зачеты	консультации	Прочая нагрузка по ВПО	Подготовка аспирантов, интернов, ординаторов	Всего осенний семестр	Всего весенний семестр	Всего за год нагрузка
Инсталляция и эксплуатация вычислительных систем и сетей	7	18ВВ1	20	1	18,0	36,0				7,2		2,8			64	0	64,0
Инсталляция и эксплуатация вычислительных систем и сетей	9	17ВО2	13	0	36,0	36,0				4,5	3,75	4,60			84,85	0,00	84,85
Арифметические и логические основы вычислительной техники	2	21ВВ1-4	95		17,0	136,0				28,5		10,8			0	192	192,3
Администрирование вычислительных систем	7	18ВВ2	18	1	18,0	36,0				8,1		2,8			65	0	64,9

Рисунок 1.3 – Таблица для отдельного преподавателя

1.2 Анализ существующих аналогов

Среди аналогов автоматизированных систем, позволяющих выполнять распределение и формирование учебной нагрузки можно выделить:

– АС «НАГРУЗКА ВУЗА»;

– АВТОМАТИЗИРОВАННАЯ СИСТЕМА РАСЧЕТА И РАСПРЕДЕЛЕНИЯ УЧЕБНОЙ НАГРУЗКИ ПРЕПОДАВАТЕЛЕЙ КАФЕДРЫ ВУЗА НА БАЗЕ ПЛАТФОРМЫ «1С»;

– Подсистема «Педагогическая нагрузка».

1.2.1 Автоматизированная система «НАГРУЗКА ВУЗА»

Графический интерфейс приложения «АС «НАГРУЗКА ВУЗА» представлен на рисунках 1.4 и 1.5.

Согласно описанию автоматизированной системы [1], из основных особенностей данного аналога можно выделить систему доступов, которая определяет доступный функционал пользователю. Всего имеется три уровня доступов.

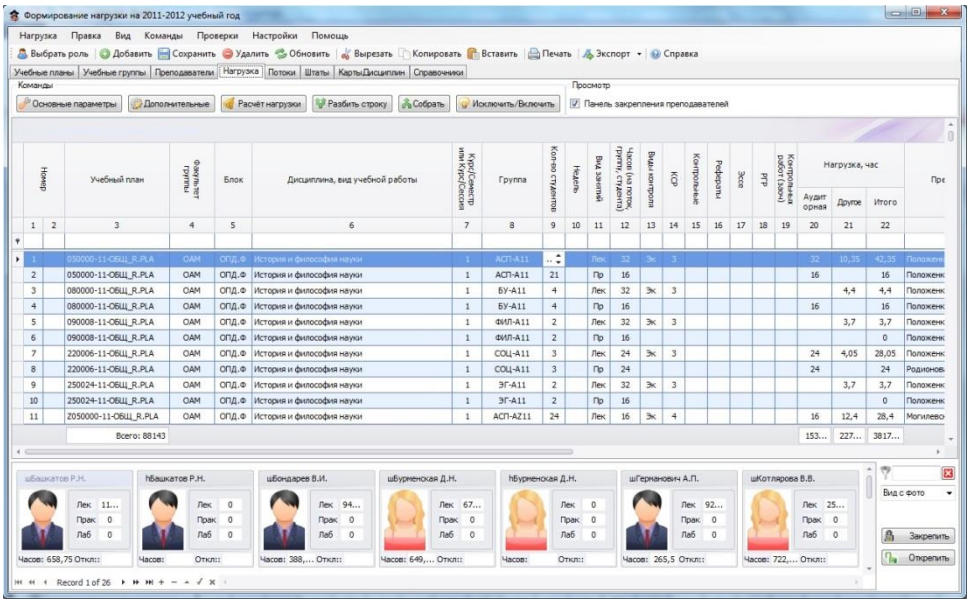


Рисунок 1.4 – Главный экран приложения

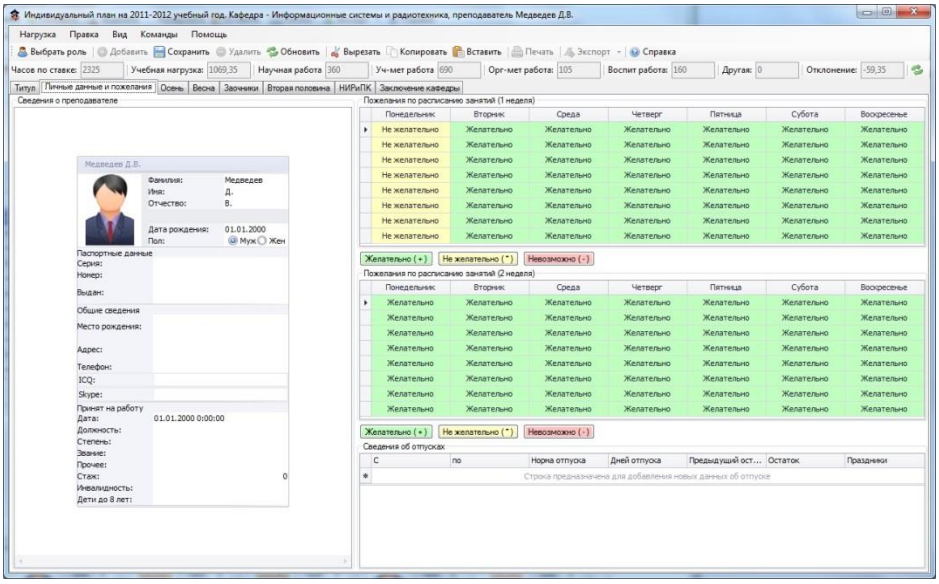


Рисунок 1.5 – Информация о преподавателе

Администратор системы, который имеет возможность проверять учебные планы, создавать список учебных групп, задавать нормы времени на проведение лекций, лабораторных и практических занятий, для приема зачетов и экзаменов, проверок дипломных работ, курсовых. Также администратору доступно формирование учебной нагрузки.

Заведующему кафедрой доступно закрепление учебной нагрузки за преподавателями, учет данных о распределении учебной нагрузки предыдущего года, проверка корректности и соответствия распределения учебной нагрузки.

Преподавателям доступна информация по учебной нагрузке, заполнение количество часов необходимо на подготовку к занятиям. Также преподавателям доступно ведение учета собственных пожеланий по графику работы в течение недели.

Из плюсов данного приложения можно выделить следующее:

– наличие уровней доступа, которые позволяют разграничить взаимодействие с учебной нагрузкой;

– большой функционал;

– наличие пожеланий преподавателя по расписанию занятий;

– наличие карт дисциплин, которые позволяют осуществлять предварительное ознакомление кафедр с перечнем закрепленных за ними дисциплин и распределением аудиторных часов.

Из минусов можно выделить следующее:

– устаревший, громоздкий интерфейс;

– большое количество выводимой информации;

– так как приложение имеет систему доступа, то требуется установка и постоянное подключение к конкретной СУБД, а именно *Microsoft SQL Server* версии 2000 и выше;

– невозможность работы с форматом расчетного листа кафедральной нагрузки нашего университета;

– распространение на коммерческой основе.

1.2.2 Автоматизированная система на базе платформы «1С»

Графический интерфейс приложения «АВТОМАТИЗИРОВАННАЯ СИСТЕМА РАСЧЕТА И РАСПРЕДЕЛЕНИЯ УЧЕБНОЙ НАГРУЗКИ

ПРЕПОДАВАТЕЛЕЙ КАФЕДРЫ ВУЗА НА БАЗЕ ПЛАТФОРМЫ «1С» представлен на рисунках 1.6, 1.7, 1.8.

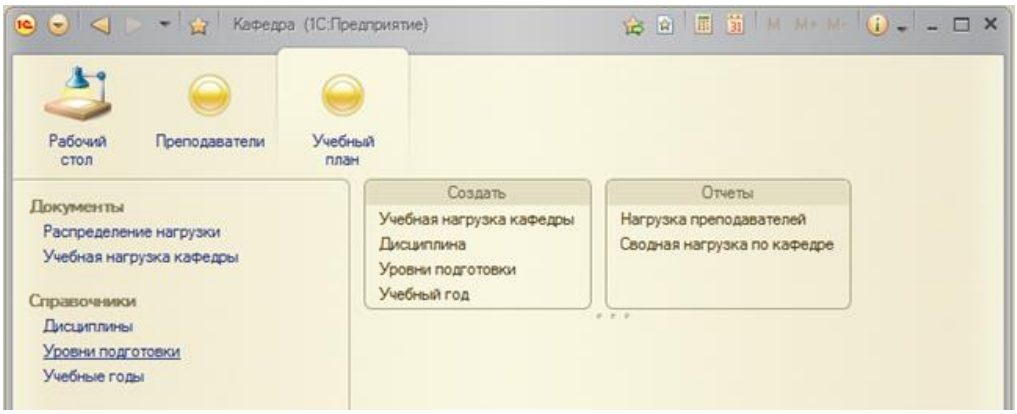


Рисунок 1.6 – Главный интерфейс приложения

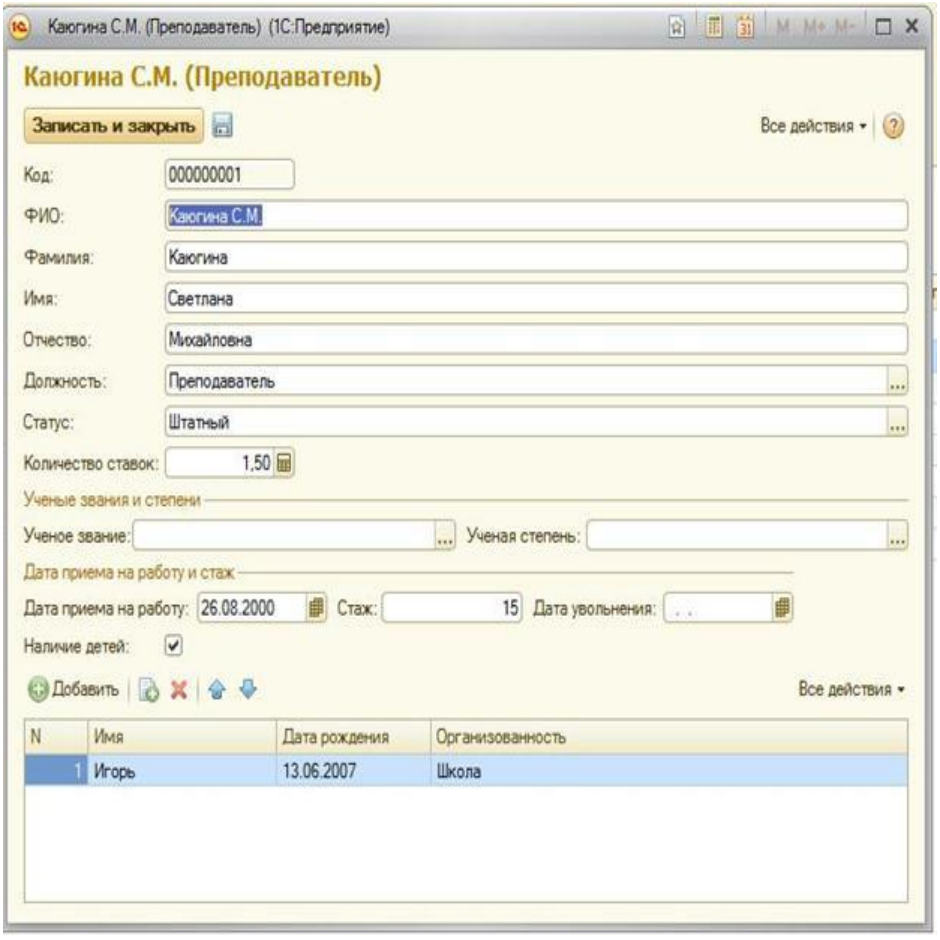


Рисунок 1.7 – Окно для взаимодействия с информацией о преподавателе

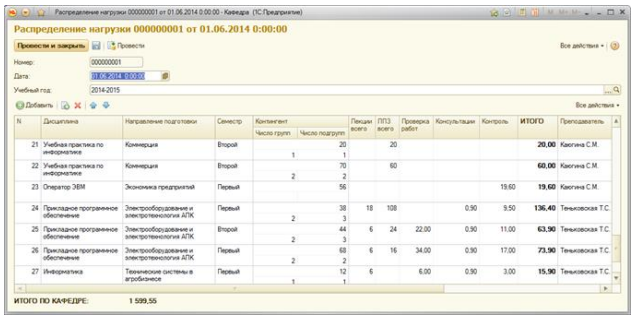


Рисунок 1.8 – Табличный вид учебной нагрузки

Данная система позволяет хранить подробную информацию о преподавателях. Также приложение производит определение часов необходимых для лекций, лабораторно-практических занятий. Из описания данного приложения [2], можно выделить возможность определение затрат времени на прием зачета или экзамена, необходимых часов для проверки письменных работ, а также определяются часы консультаций в зависимости от формы обучения.

Приложение имеет возможность формирования отчетов, представленных в виде диаграмм и графиков.

Имеется возможность экспортирования информации в электронную таблицу *Excel*, но формат таблицы не соответствует нашему университету.

1.2.3 Подсистема «Педагогическая нагрузка»

Графический интерфейс приложения «Подсистема «Педагогическая нагрузка»» представлен на рисунках 1.9 и 1.10.

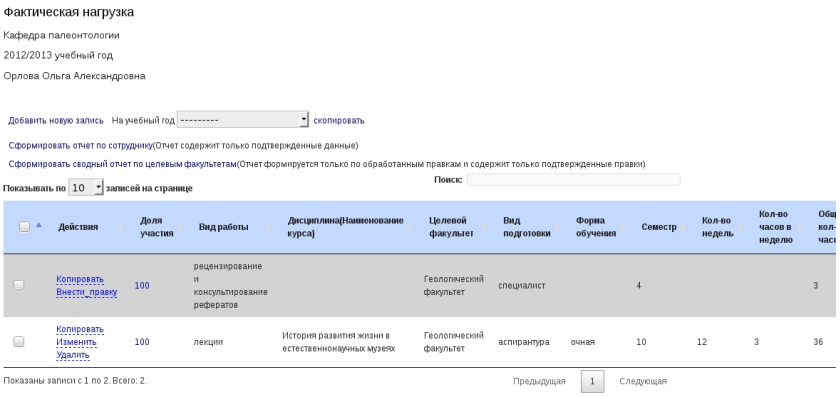


Рисунок 1.9 – Интерфейс для фактической нагрузки

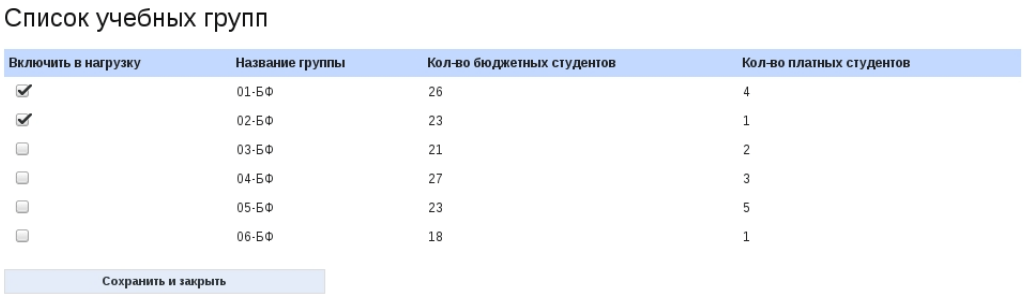


Рисунок 1.10 – Список учебных групп

У данного приложения имеются функциональные возможности, позволяющие преподавателям вносить и просматривать информацию о своей фактической преподавательской нагрузке, оформлять заявки на изменение данных. Также имеется функциональность для формирования отчетов.

В процессе работы система также может оповещать пользователя о некорректности выполняемого им действия.

Важно отметить, что данное приложение не предназначено для коммерческого использования.

1.3 Выбор языка программирования

1.3.1 Язык программирования *C++*

C++ – объектно-ориентированный, компилируемый, строго типизированный язык программирования. *C++* поддерживает различные способы программирования, такие как процедурный, объектно-ориентированный, функциональный и т. д. Это делает *C++* мощным и гибким. Но C++ не является достаточно гибким для создания графического интерфейса.

1.3.2 Язык программирования *Java*

Java – кроссплатформенный строго типизированный, объектно-ориентированный язык программирования. Кроссплатформенность достигается использованием виртуальной машиной *jvm – java virtual machine*. Также данный язык имеет библиотеки, которые подходят для работы с

файлами формата *xlsx*. К недостаткам можно отнести то, что при работе с библиотеками для создания оконных приложений, процесс создание сложных, настраиваемых интерфейсов может стать довольно долгим.

1.3.3 Язык программирования *Python*

Python – объектно-ориентированный интерпретируемый язык программирования высокого уровня с динамической типизацией. Он имеет очень простой синтаксис, который упрощает изучение этого языка, а также повышает скорость разработки.

Интерпретатор *Python* и стандартные библиотеки распространяются бесплатно, поскольку они доступны в исходном или двоичном виде, а отладка программ на *Python* очень проста благодаря их функциям.

Несмотря на явные достоинства *Python*, у него есть и недостатки, о которых следует помнить. *Python*. Это интерпретируемый скриптовый язык, поэтому он работает относительно медленнее своих скомпилированных аналогов. Также у существующих библиотек для графического интерфейса сложно модифицировать макет приложений.

1.3.4 Язык программирования *C#*

C# – это язык программирования, разработанный и запущенный *Microsoft* в 2001 году. *C#* – строго типизированный объектно-ориентированный язык программирования, который предоставляет современным разработчикам гибкость и лёгкость для создания программного обеспечения.

Данный язык программирования имеет открытый исходный код в рамках *.NET Foundation*. Благодаря открытости, есть множество сообществ, активно занимающихся улучшением языка.

Поскольку *C#* и *Windows* разработаны *Microsoft*, разработчики могут легко создавать настольные приложения на основе *Windows*.

Также можно добавить к описанным выше преимуществам наличие библиотек для автоматизации работы в *Excel*. Данные библиотеки позволяют

выполнять такие действия, как создание новой таблицы, добавление различных данных в эту таблицу. В *Excel* и других приложениях *Microsoft Office* практически все действия, которые можно выполнить вручную через пользовательский интерфейс, также можно выполнить программным путем через данные библиотеки.

1.4 Выводы

По результатам анализа существующих аналогов было принято решение разработать автоматизированную систему, которая будет решать задачи по взаимодействию с файлами формата *xlsx*, предоставлять возможность взаимодействия со всей необходимой информацией для планирования преподавательской нагрузки.

Разработка будет производиться с помощью языка *C#*, так как C# позволяет создавать удобный и понятный для пользователя графический интерфейс.

1.5 Выбор инструментов разработки

1.5.1 Выбор среды разработки

В качестве среды разработки было решено выбрать *Microsoft Visual Studio*. Немаловажным преимуществом данной среды является наличие диспетчера пакетов «*NuGet*». Пакеты в основном представляют собой скомпилированную библиотеку с некоторыми описательными метаданными. *NuGet* – это важный инструмент для любой современной платформы разработки. Это механизм, с помощью которого разработчики могут создавать, совместно использовать и использовать полезный код. *NuGet* сам обрабатывает все промежуточные детали.

1.5.2 Выбор платформы для реализации графического интерфейса

В качестве графической платформы приложения была выбрана платформа *WPF*.

Windows Presentation Foundation – это платформа для создания приложений *Windows*, которая позволяет разрабатывать многофункциональные пользовательские интерфейсы с меньшей сложностью кода.

В работе [3] указана особенность описание данной платформы ,а и именно наличие векторного движка рендеринга, который использует аппаратное ускорение современных графических видеокарт, что делает графический интерфейс более быстрым и легко масштабируемым.

WPF можно рассматривать как эволюцию *Windows Forms*. Платформа предлагает легкое разделение между пользовательским интерфейсом и функциональной логикой.

Все графические элементы определяются с помощью языка *XAML*. *XAML* — это язык декларативной разметки на основе *XML* для указания и установки характеристик классов. Синтаксический анализатор *XAML* будет автоматически анализировать, и создавать фактический объект во время визуализации приложения. *С* помощью данного языка определяется весь макет пользовательского интерфейса.

Также основными преимуществами *WPF* при создании графического интерфейса являются:

— стили *WPF*, которые позволяют устанавливать определённые свойства для графических элементов управления и повторно использовать эти конкретные настройки для нескольких элементов, для придания единообразного внешнего вида всему приложению;

— шаблоны элементов, позволяющие определять внешний вид элемента управления. Например, внешний вид кнопки по умолчанию можно изменить, сделав кнопку круглой, с наличием картинки и определенного текста, изменится лишь только внешний вид, а поведение кнопки останется прежним;

— механизм привязки данных, который позволят выполнять переноса данных из кода на уровень пользовательского интерфейса или наоборот.

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

1.5.3 Выбор вспомогательного фреймворка

В качестве вспомогательного фреймворка был выбран *Prism.*

Prism – это фреймворк, который позволяет разрабатывать слабосвязанные приложения, которые являются гибкими, удобными в сопровождении и простыми в тестировании. Приложения *Prism* состоят из модулей – слабо связанных функциональных блоков, которые инкапсулируют части общей функциональности приложения. *Prism* можно использовать для разработки как настольных, так и мобильных приложений, соответствующих шаблону проектирования *MVVM* (*Model — View — ViewModel*), поскольку он поддерживает создание проектов *WPF*.

Prism предоставляет различные возможности и шаблоны проектирования, основные концепций *Prism* представлены ниже:

Module – модули по своей сути являются библиотеками, содержащими различные классы, бизнес-функции, графические части приложения. Модули можно независимо разрабатывать и повторно использовать в других приложениях;

Shell – оболочка определяет общий макет и структуру приложения, но обычно не знает, какие именно модули она будет размещать. Оболочка также предоставляет окно верхнего уровня или визуальный элемент, в котором затем размещаются различные компоненты пользовательского интерфейса, предоставленные загруженными модулями;

Views – представления содержат в себе элементы управления пользовательского интерфейса. Представления используются в сочетании с шаблоном *MVVM*. Представления используют механизм привязки данных

WPF для взаимодействия с классами модели представления или отображение различных данных из этих классов;

ViewModels – модели представлений это классы, которые инкапсулируют логику представления и состояние приложения. Они являются частью шаблона *MVVM*. Модели представлений инкапсулируют большую часть функциональных возможностей приложений, определяют свойства, команды и события, к которым элементы управления могут привязываться;

Commands – команды используются для инкапсуляции функциональных возможностей приложения таким образом, чтобы их можно было определять и выполнять независимо от пользовательского интерфейса приложения;

Regions – области, определенные в пользовательском интерфейсе приложения (в оболочке или в представлениях), в которых отображаются различные представления, с содержащимися в них графическими элементами;

Navigation – процесс, с помощью которого приложение изменяет свой пользовательский интерфейс в результате взаимодействия пользователя с приложением или изменений внутреннего состояния приложения;

EventAggregator – класс, который реализует паттерн событий *pub – sub* (издатель – подписчик), тем самым имеется возможность различным компонентам приложения публиковать события, а другим компонентам подписываться на эти события, при этом ни один из них не требует ссылки на другой. *EventAggregator* часто используется, чтобы позволить классам, которые находятся в разных модулях, взаимодействовать друг с другом;

Container DI – контейнер внедрения зависимостей используется в *Prism* для управления зависимостями между компонентами. *Prism* предназначен для работы с контейнерами внедрения зависимостей, которые могут использоваться с платформой *.NET*.

1.5.4 Выбор библиотеки для взаимодействия с таблицами *Excel*

Для работы с таблицами используется *EPPlus*. *EPPlus* – это библиотека *.NET*, которая читает и записывает файлы *Excel* в формате *Office Open XML* (*xlsx*). Данная библиотека поддерживает широкий круг возможностей:

– работа с диапазонами ячеек;

– стилизация ячеек (граница, цвет, заливка, шрифт, выравнивание);

– расстановка формул;

– создание графиков.

С помощью данной библиотеки можно осуществлять считывание данных из ячеек таблицы и генерацию собственных файлов.

1.6 Архитектура приложения

1.6.1 Паттерн *MVVM*

При разработке приложения основным паттерном разработки является *MVVM*. Паттерн *Model View ViewModel* (*MVVM*) помогает четко отделить бизнес-логику и логику представления приложения от его пользовательского интерфейса. Поддержание четкого разделения между логикой приложения и пользовательским интерфейсом помогает решать многочисленные проблемы разработки и проектирования и может значительно упростить тестирование, обслуживание и развитие приложения. Также может значительно улучшить возможности повторного использования кода.

Паттерн можно разделить на три отдельных компонента:

– представление (*View*), которое инкапсулирует пользовательский интерфейс и логику пользовательского интерфейса;

– модель представления (*ViewModel*), которая инкапсулирует логику представления и состояние;

– модель (*Model*), которая инкапсулирует различные данные приложения.

На рисунке 1.11 показаны три компонента использующиеся в паттерне *MVVM* и отображено взаимодействие между ними.

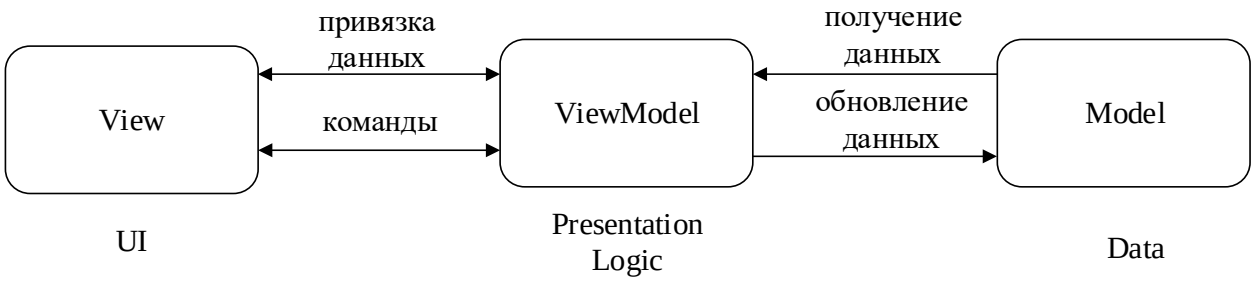


Рисунок 1.11 – Интерфейс для фактической нагрузки

1.6.2 Архитектура

Так как при разработке приложения используется фреймворк *PRISM*, то архитектура с учетом возможностей данного фреймворка из источника [4] представлена на рисунке 1.12.

Архитектура состоит из следующих слоев:

– *PRESENTATION LAYER* – слой представлений;

– *MODULES LAYER* – слой модулей;

– *CORE LAYER* – основной слой;

– *DATA LAYER* – слой данных.

Слой представлений состоит из оболочки, которая содержит в себе регионы, необходимые для отображения различных представлений (*Views*), содержащих в себе различные графические элементы управления.

В слое модулей содержатся различные модули приложения, состоящие из моделей представлений (*ViewModel*) и соответствующих им необходимых представлений (*View*), для реализации паттерна *MVVM*. Также в модулях могут содержаться различные контроллеры, для инкапсуляции взаимодействия различных сервисов.

Основной слой содержит в себе различные сервисы, которые реализуют взаимодействие с различными файлами, сохранение данных о преподавателях и учебных группах. Также через сервисы происходит взаимодействие с классами, которые находятся в слое данных.

Слой данных состоит из различных моделей, которые содержат в себе всю необходимую информацию о преподавателях, учебных дисциплинах, учебной нагрузки, учебных группах. Также в данных моделях реализован функционал по расчету времени, обрабатывание событий изменений свойств в зависимых классах. Также модели проверяют корректность данных, которые были введены пользователем.

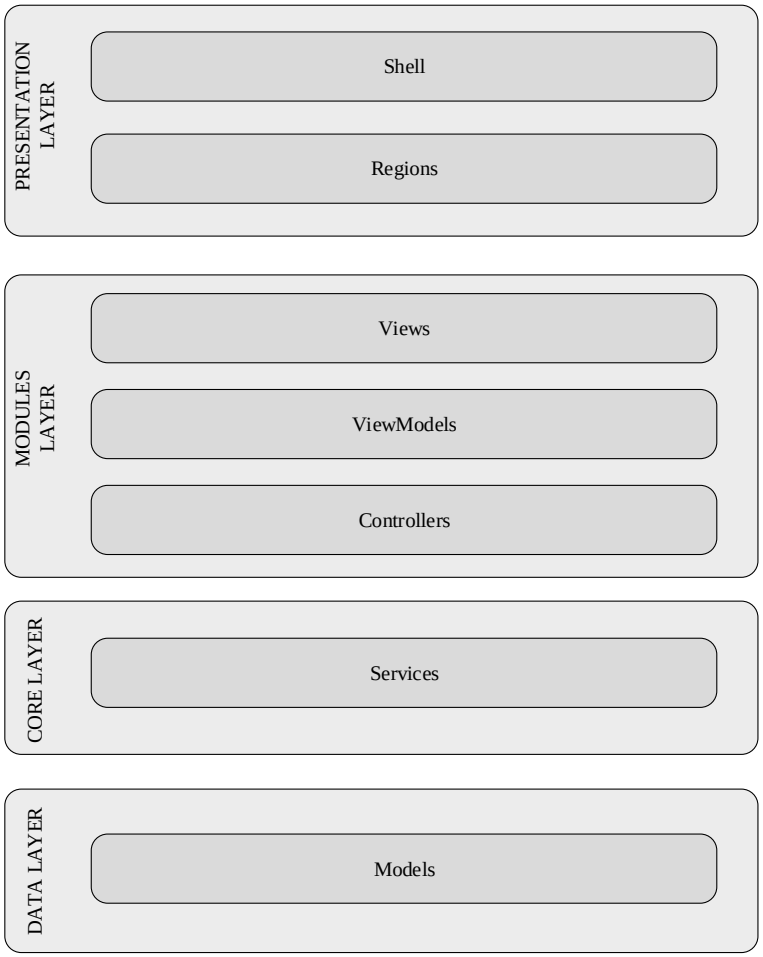


Рисунок 1.12 – Архитектура Приложения

1.7 Разработка приложения

1.7.1 Разработка графического интерфейса

Графический интерфейс должен предоставлять возможности для взаимодействия с приложением в соответствие *UML* диаграммы использования представленной на рисунке 1.13.

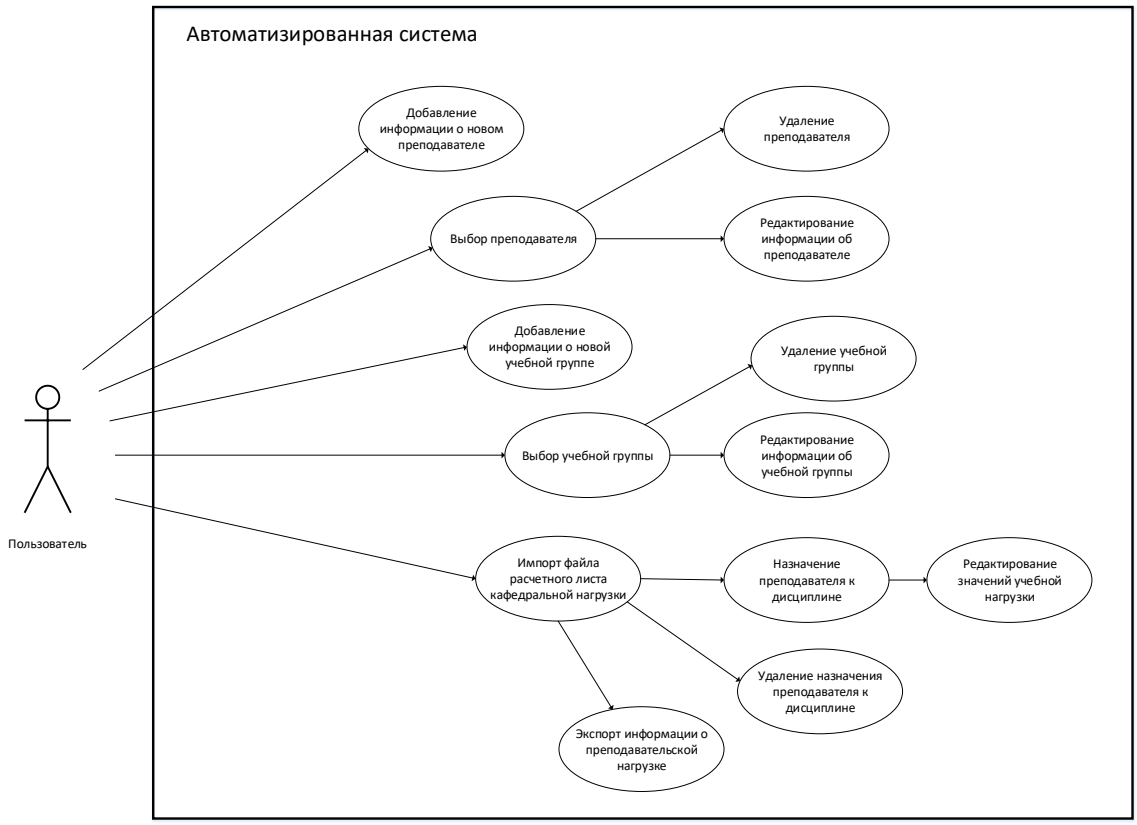


Рисунок 1.13 – UML – диаграмма вариантов использования автоматизированной системы.

Графический интерфейс состоит из разных *View*, каждое из которых предназначено для определённого взаимодействия с пользователем.

AddGroupView – представление позволяющие пользователю добавлять информацию для новой учебной группы.

AddTeacherView – представление необходимо для возможности пользователя добавлять информацию для нового преподавателя.

AddTeacherLoadDisciplineView – представление для добавления пользователем назначения преподавателя к дисциплине.

EditTeacherView – представление для редактирования пользователем информации о преподавателе.

EditGroupView – представление для редактирования информации о учебной группе.

CalculationSheetView – представление, которое используется для отображения информации из импортированного расчетного листа кафедральной нагрузки.

GroupsView – представление для отображения данных всех учебных групп.

TeachersView – представление необходимо для отображения пользователю информации о преподавателях.

TabsView – представление содержащие в себе регионы фреймворка *Prism*, которые содержат в себе представление описанные выше. Для удобной навигации между данными регионами используется графический элемент управления *TabControl*.

В работе [5] рассмотрены примеры, которые можно использовать для реализации блокировки ввода пользователем некоренных значений в процессе работы

1.7.1.1 Создание стилей *WPF*

Стили позволяют придать единообразный внешний вид набору элементов управления. *WPF* предоставляет свойство *Style* для каждого элемента управления, которое можно произвольно задать. Стили могут быть объявлены для любого элемента, производного от *FrameworkElement* или *FrameworkContentElement*.

Самый распространенный способ объявить стиль в качестве ресурса и создать отдельный словарь ресурсов языка *XAML*. В процессе разработки были созданы словари ресурсов для каждого графического элемента, которые используются в приложении.

Buttons.xaml – используется для хранения стилей элемента *Button* (кнопка).

Colors.xaml – содержит ресурсы всех цветов используемых в приложении.

ComboBoxes.xaml – предназначен для хранения стилей элемента *ComboBox*(выпадающий список).

Converters.xaml – необходим для определения созданных конверторов. Привязке *WPF*, которая помогает передавать данные между двумя объектами пользовательского интерфейса *WPF*. Объект, который передает данные, называется объект – источник, а объект, который принимает данные, называется объект – приемник. Конверторы позволяют преобразовать данные из одного формата в другой формат, когда они переходят из объекта – источника в объект – приемник или наоборот.

DataGrids.xaml – предназначен для хранения стилей элемента *DataGrids* (элемент управления, отображающий данные в таблице).

DatePicker.xaml – используется для хранения стилей элемента *DatePicker* (текстовое поле, предназначенное для ввода даты с выпадающим календарем после ввода).

Icons.xaml – предназначен для хранения всех векторных иконок используемых в приложении.

TabsControls.xaml – используется для хранения стилей элемента *TabsControl*.

Text.xaml – предназначен для настраивания таких параметров текста как размер, шрифт, цвет, форматирование, цвет заднего фона в элементах *TextBox* и *TextBlock*.

ToolTips.xaml – необходим для хранения стилей элемента *ToolTip* (всплывающая подсказка).

1.7.1.2 Диалоговые окна

Диалоговое окно позволяет пользователям выполнять различные команды, запрашивать у пользователя подтверждение выполнения операции или показывать пользователю сообщение о ходе выполнения программы.

В приложении используются три диалоговых окна, для каждого из которых имеется свой класс *View* (графическая часть), *ViewModel* (взаимодействие с графической частью).

AddDialog – диалоговое окно, предназначенное для добавления новой информации пользователем.

ConfirmDialog – диалоговое окно, предназначенное для подтверждения пользователей выполнения определённого действия.

NotificationDialog – диалоговое окно, предназначенное для уведомления пользователя о возникновении события.

Для удобного взаимодействия с данными окнами используется класс *DialogController*, имплементирующий работу с интерфейсом *IDialogService* фреймворка *Prism*, предназначенного для создания диалоговых окон.

Класс *DialogController* содержит три метода:

– *OpenAddDialog* – метод открытия диалогового окна *AddDialog*, в качестве параметров принимает *AddDialogParameters*, и делегат *Action* для функции обратного вызова;

– *OpenConfirmDialog* – метод открытия диалогового окна *ConfirmDialog*, в качестве параметров принимает *confirmDialogParameters*, и делегат *Action* для функции обратного вызова;

– *OpenNotificationDialog* – метод открытия диалогового окна *NotificationDialog*, в качестве параметров принимает строку сообщения, и строку заголовка окна.

Графический интерфейс диалоговых окон представлен на рисунках 1.14, 1.15, 1.16.

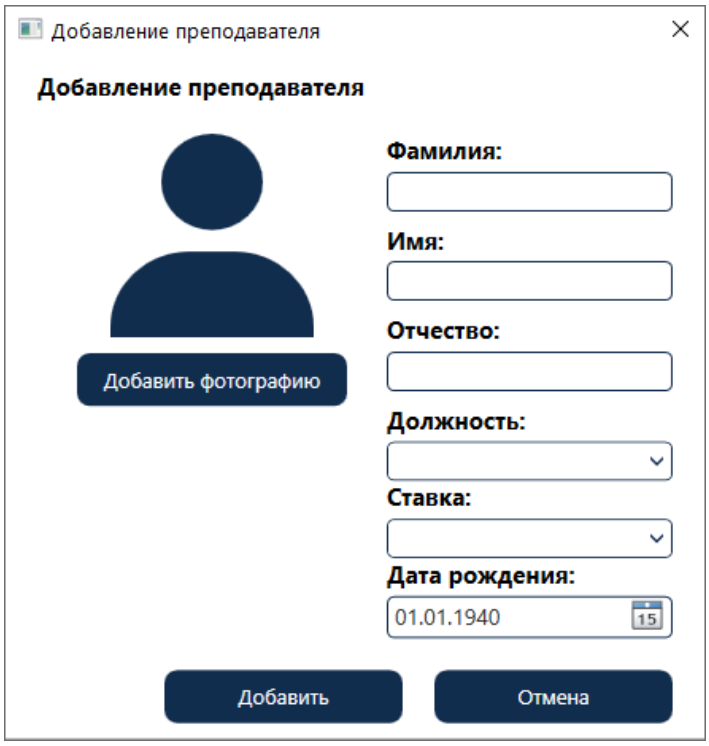


Рисунок 1.14 – Диалоговое окно AddDIalog

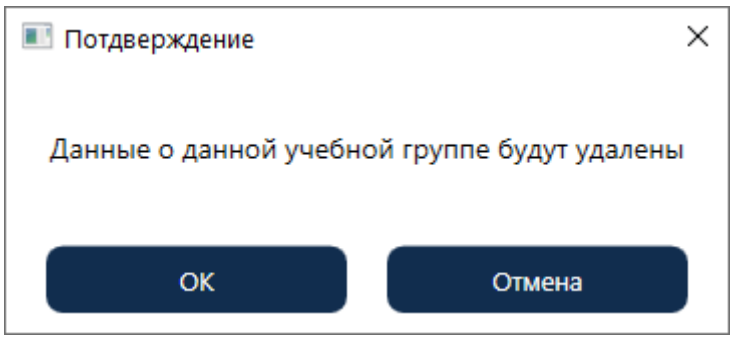


Рисунок 1.15 – Диалоговое окно ConfirmDialog

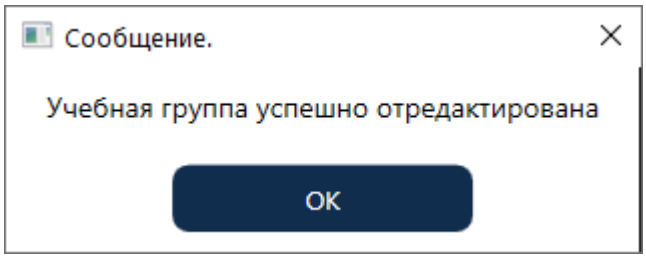


Рисунок 1.16 – Диалоговое окно NotificationDialog

1.7.2 Разработка моделей

Для хранения основной информации о расчетном листе кафедральной нагрузки, дисциплинах, преподавателях, учебных группах, назначений преподавателей к учебным дисциплинам в процессе работы приложения используются классы моделей.

1.7.2.1 Класс *Teacher*

Класс *Teacher* содержит в себе поля для хранения необходимой информации о преподавателях. Основные поля описаны ниже.

FirstName – поле, предназначенное для хранения имени преподавателя.

LastName – поле для хранения отчества преподавателя.

MiddleName – поле, предназначенное для хранения фамилии преподавателя.

AcademicTitle – поле, предназначенное для хранения имени преподавателя. Используется тип перечислений *AcaemicTitle*.

Rate – поле, предназначенное для хранения информации о ставке. Используется тип перечислений *Rate*.

Birthday – поле для хранения даты рождения. Используется структура языка *C# DateTime*.

PhotoPath – поле, предназначенное для хранения пути к фотографии.

UML диаграмма класса представлена на рисунке 1.17.

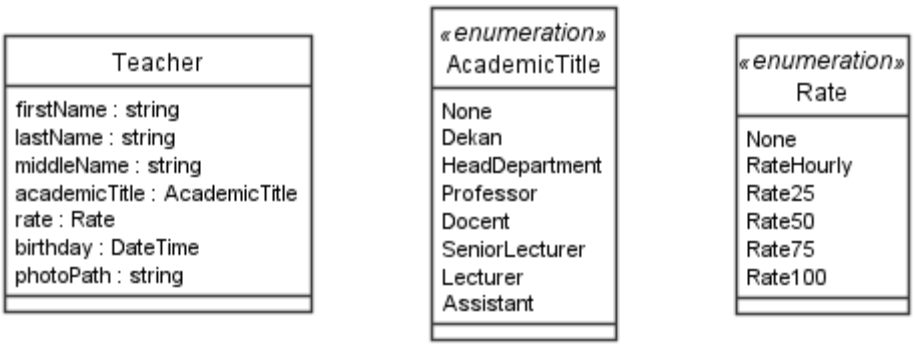


Рисунок 1.17 – *UML* – диаграмма класса *Teacher*

1.7.2.2 Класс *Group*

Класс *Group* необходим для хранения информации об учебных группах. Используемые поля описаны ниже.

Name – поле, предназначенное для хранения названия группы преподавателя.

StudentsBudget – поле для хранения количества студентов на бюджетной основе.

StudentsContract – поле для хранения количества студентов на договорной основе.

Students – get свойство для получения общего количества студентов.

UML диаграмма класса *Group* представлена на рисунке 1.18.

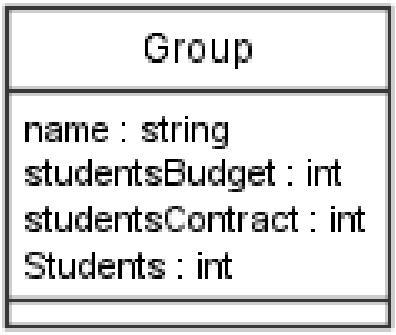


Рисунок 1.18 – *UML* – диаграмма класса *Group*

1.7.2.3 Класс *CalculationSheet*

Класс *CalculationSheet* используется в качестве модели для хранения информации из расчетного листа кафедральной нагрузки. В данной модели содержатся поля для хранения следующих данных:

– название факультета;

– кафедра;

– учебный год;

– количество студентов на бюджетной и договорной основе;

– суммарное количество часов под различные виды проведения учебных занятий.

Также в этом классе имеется список *CalculationSheetDisciplines* для хранения информации о учебных дисциплинах.

1.7.2.4 Класс *CalculationSheetDiscipline*

Класс *CalculationSheetDisciplines* предназначен для хранения данных учебных дисциплин. Данный класс хранится в списке *CalculationSheetDisciplines* класса *CalculationSheet*

Поля данного класса содержат в себе следующую информацию:

– индекс дисциплины;

– название дисциплины;

– семестр;

– учебные группы, количество студентов;

– часы, отведенные для проведения каждого вида учебных занятий:

В данном классе используется коллекция *ObservableCollection*, для хранения объектов класса *TeacherLoadDiscipline*. В работе [6] сказано, что класс *ObservableCollection* поддерживает событие по имени *CollectionChanged*. Указанное событие будет инициироваться каждый раз, когда вставляется новый элемент, удаляется (или перемещается) существующий элемент либо модифицируется вся коллекция целиком. Данная особенность используется для автоматического обновления части графического интерфейса, связанного с отображением назначений преподавателей к конкретной учебной дисциплине.

1.7.2.5 Класс *TeacherLoadDiscipline*

Класс используется в качестве модели назначения преподавателя к учебной дисциплине. Данный класс состоит из полей, которые необходимы для хранения информации о количестве часов под каждый вид занятий. Имеется ссылка на объект класса *Teacher* для хранения информации о преподавателе, для которого происходит назначение. Для хранения информации об учебных группах используется список *Groups*.

Для автоматического обновления суммарного количества часов для весеннего или осеннего семестра и годовой нагрузки используется обработчик для события изменения свойств данного класса. Автоматическое обновление происходит и в графическом интерфейсе.

1.7.2.6 Отношения между объектами классов

Отношение между объектами классов моделей показаны на рисунке 1.19. Также на этом рисунке указана кратность в соответствие с работой [7]. Кратность – это количество экземпляров одного класса которые могут быть с связаны с экземпляром другого класса. Значение кратности указывается в виде диапазона «1» (ровно один), «0..*» (ноль и более).

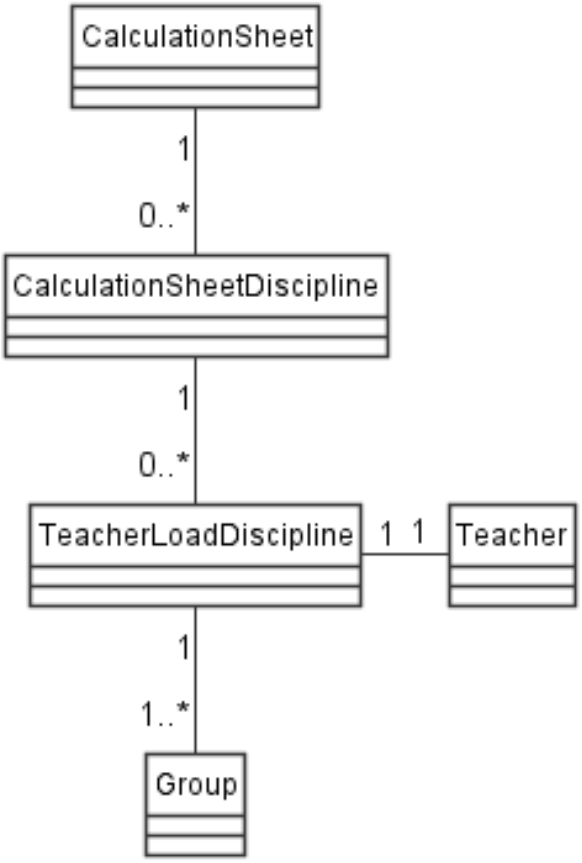


Рисунок 1.19 – *UML* – диаграмма отношений между объектами классов

Класс CalculationSheet связан с классом CalculationSheetDiscipline отношением один ко многим. В одном кафедральном листе имеется содержится большое количество учебных дисциплин.

Класс CalculationSheetDiscipline связан отношением один ко многим с классом TeacherLoadDiscipline. Учебная дисциплина может иметь назначения для различных преподавателей.

Класс TeacherLoadDiscipline связан с классом Teacher отношением один к одному. Назначение создается только для одного преподавателя.

Класс TeacherLoadDiscipline связан с классом Groups отношением один ко многим. Преподаватель может вести дисциплину у разных групп.

1.8 Описание приложения

1.8.1 Руководство пользователя

После запуска клиентского приложения откроется главное окно программы. Окно содержит в верхней части вкладки для переключения интерфейса для взаимодействия с преподавателями, учебными группами, нормами времени, расчетным листом кафедральной нагрузки. Вкладки представлены на рисунке 1.20. Интерфейс главного окна представлен на рисунке 1.21.

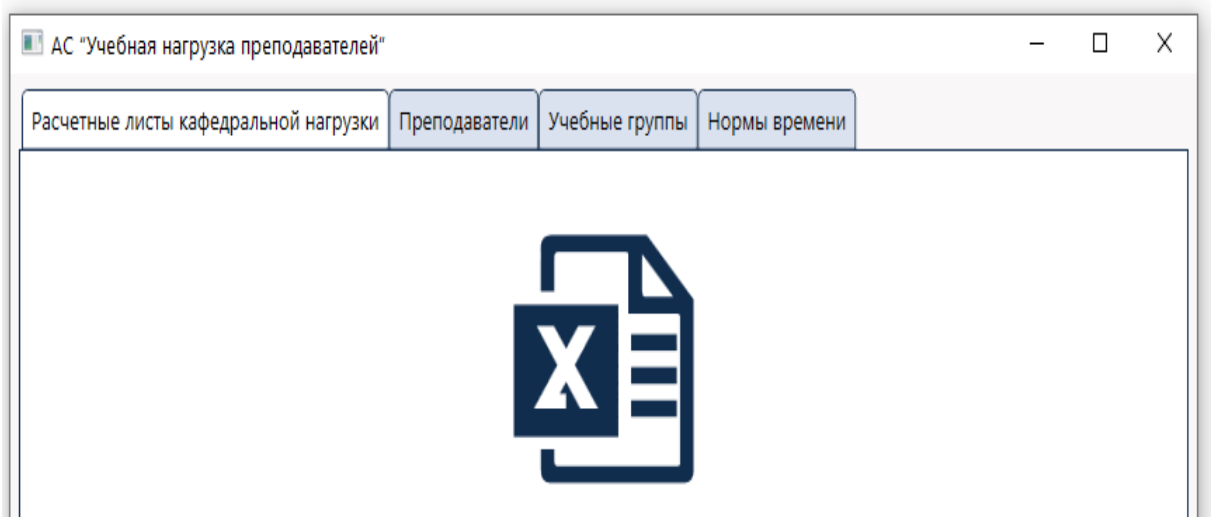


Рисунок 1.20 – Вкладки приложения

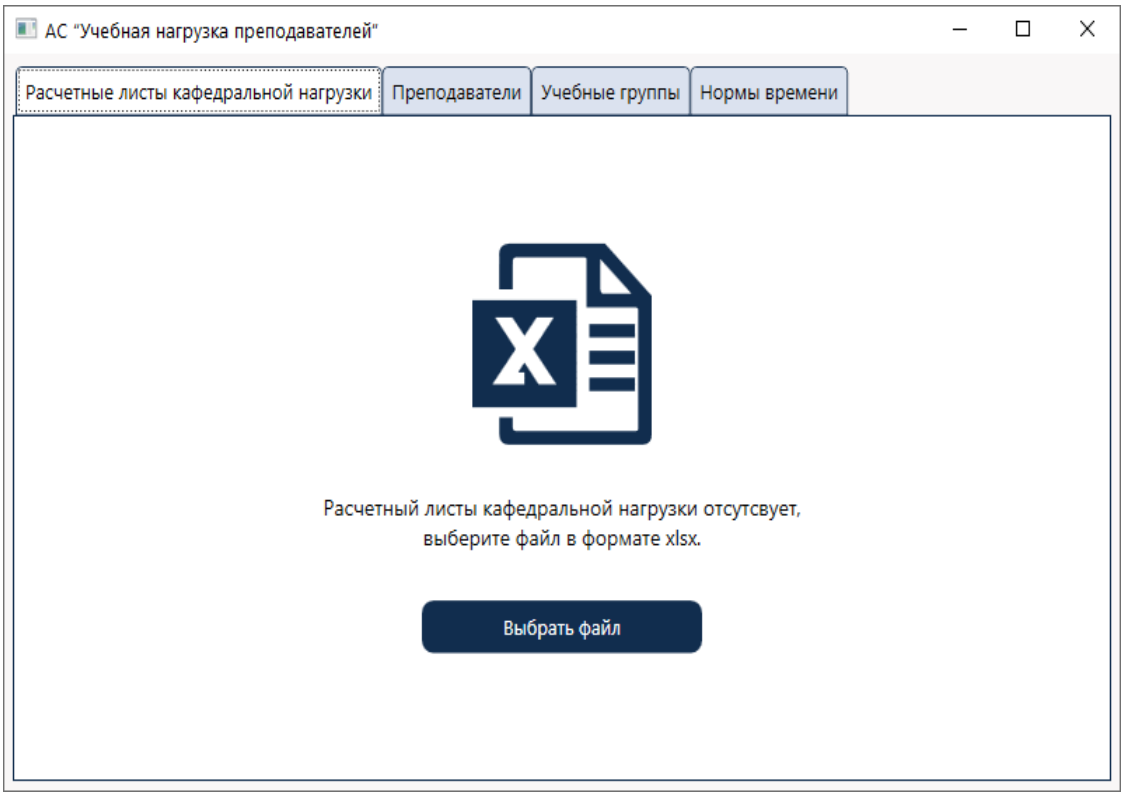


Рисунок 1.21 – Основное окно приложения

Пользователю предлагается выбрать *xlsx* файл с помощью стандартного окна выбора файла. Если информацию из выбранного пользователя файлом не удается импортировать в приложение, то появляется окно с сообщением об ошибке. Окно представлено на рисунке 1.22.

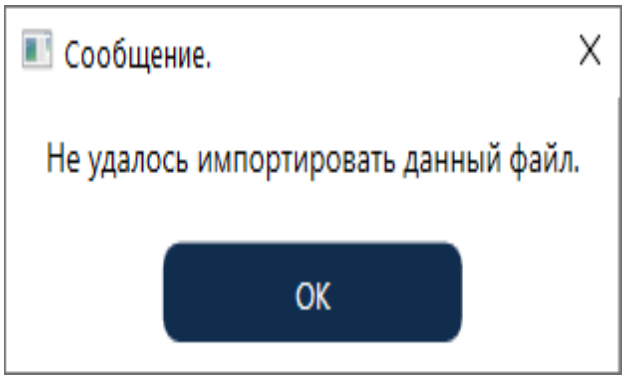


Рисунок 1.22 – Окно с сообщением о неудачном импорте файла

В случае успешной загрузки информации, она выводится в табличном виде, представленном на рисунке 1.23. Пользователь может удалить

информацию из загруженного расчетного листа кафедральной нагрузки из приложения нажатием соответствующей кнопки рядом с названием файла. Для импорта нового расчетного листа необходимо повторить действия описанные выше.

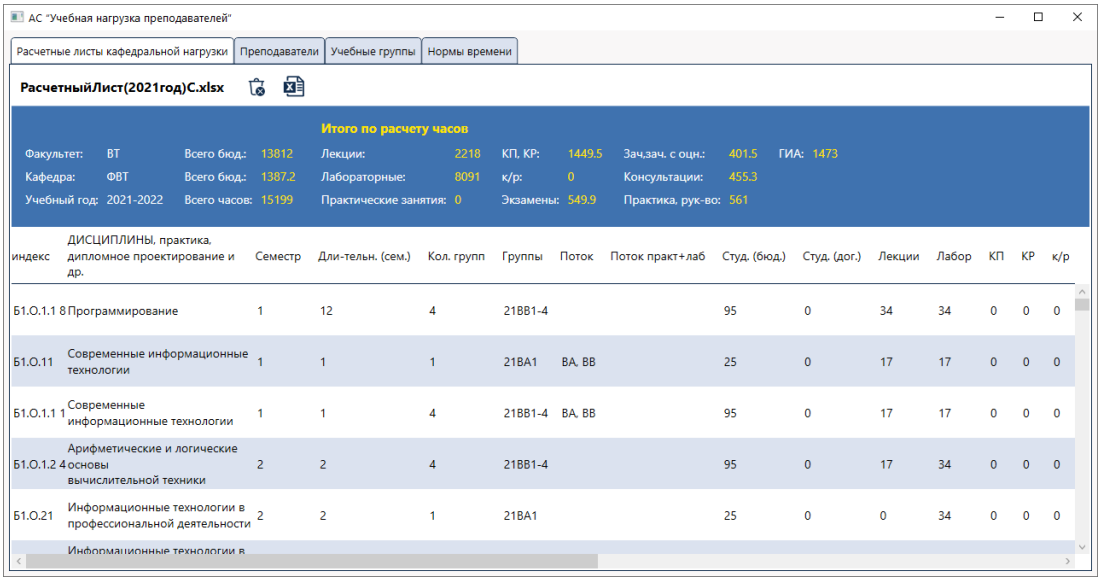


Рисунок 1.23 – Импортированная информация из расчетного листа кафедральной нагрузки

Для назначений преподавателя к дисциплине необходимо выбрать левым нажатием мыши необходимую учебную дисциплину. Под выбранной дисциплиной появится раздел сведений с кнопкой для назначения преподавателя. Раздел сведений показан на рисунке 1.24.

По нажатию на данную кнопку отрывается диалоговое окно для назначения преподавателя. Диалоговое окно представлено на рисунке 1.25. В данном окне пользователь может выбрать назначаемого преподавателя, выбрать одну или несколько учебных групп из предложенных, указать информацию о том, что преподаватель является основным или дополнительном лектором.

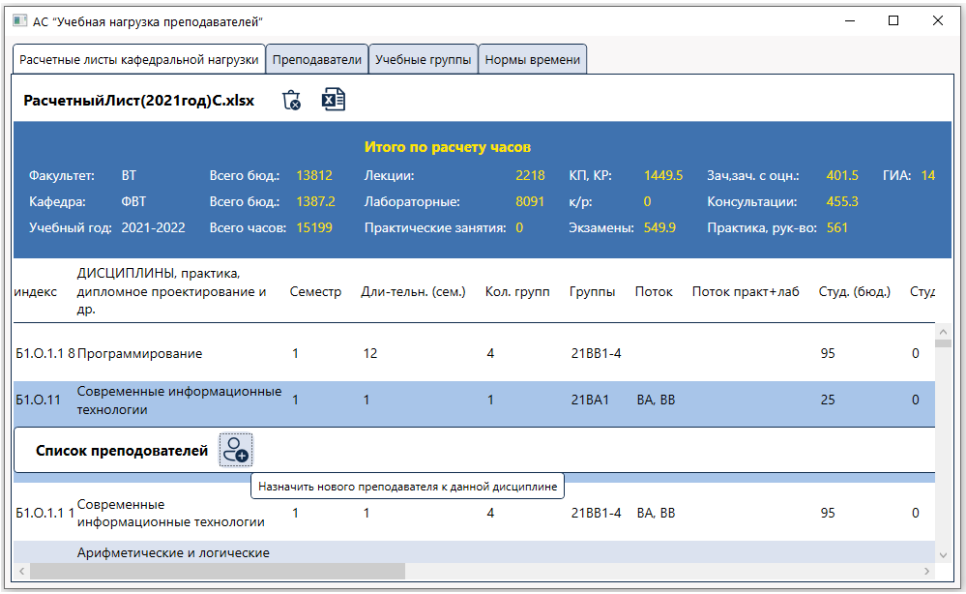


Рисунок 1.24 – Раздел сведений о назначенных преподавателях

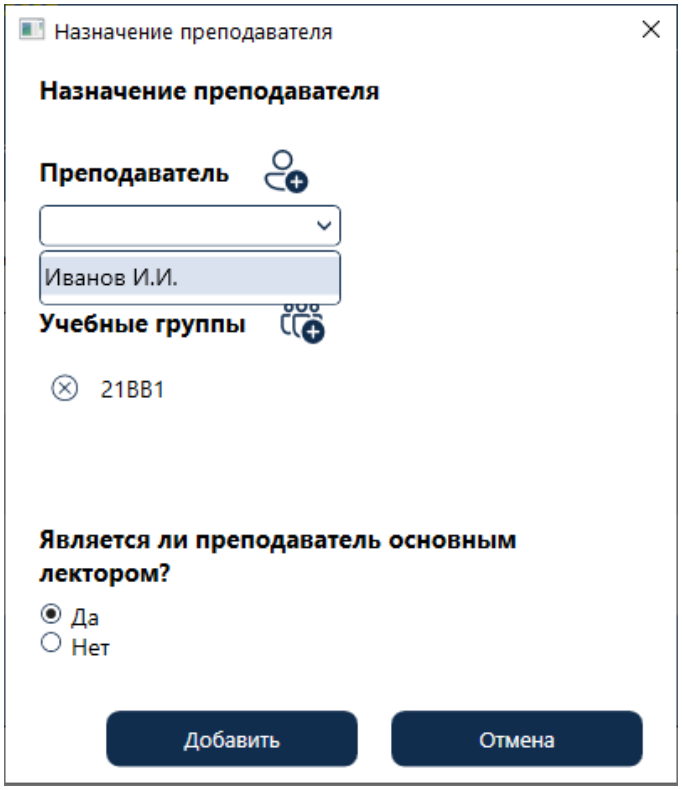


Рисунок 1.25 – Диалоговое окно для назначения преподавателя

После назначений преподавателя, раздел сведений примет вид, показанный на рисунке 1.26. Пользователь может редактировать отдельные значений, если они доступны для редактирования.

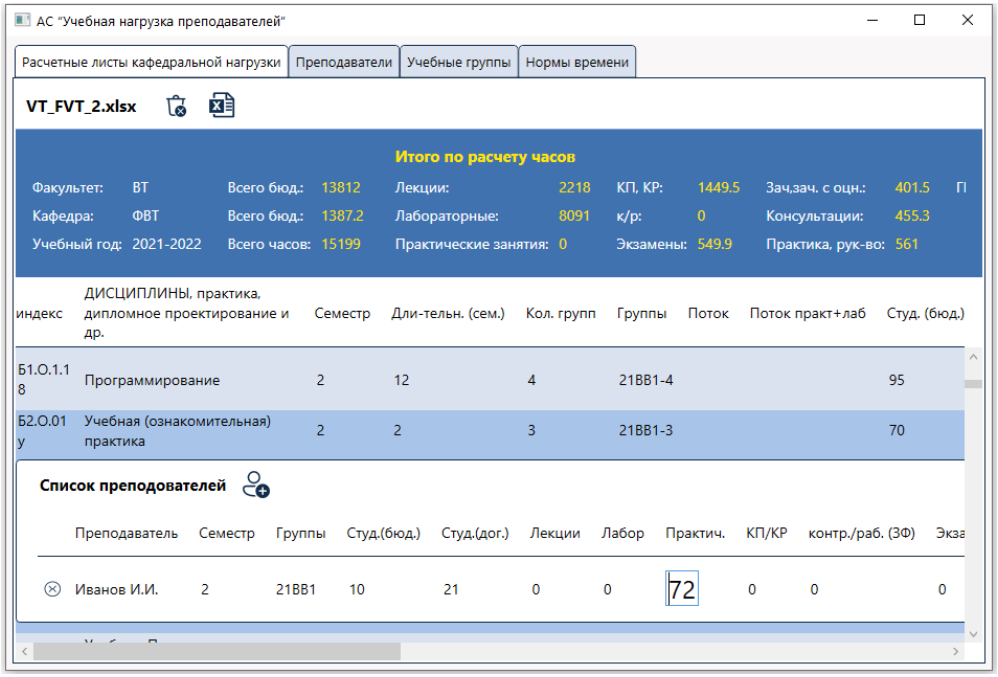


Рисунок 1.26 – Раздел сведений с назначенным преподавателем

Для экспорта информации о преподавательской нагрузке необходимо нажать на кнопку, показанную на рисунке 1.27. Программа предложит пользователю ввести желаемое имя файла, в который будет производиться экспорт информации о преподавательской нагрузке.

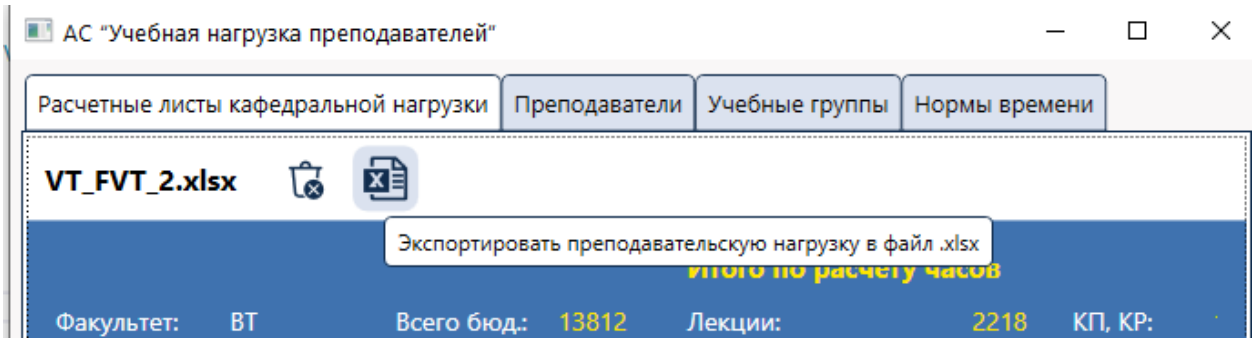


Рисунок 1.27 – Кнопка для экспортирования преподавательской нагрузки

Для взаимодействия с преподавателями, пользователю необходимо перейти по вкладке «Преподаватели». Интерфейс, отображающий информацию о преподавателях представлен на рисунке 1.28. Имеется возможность производить поиск по ФИО.

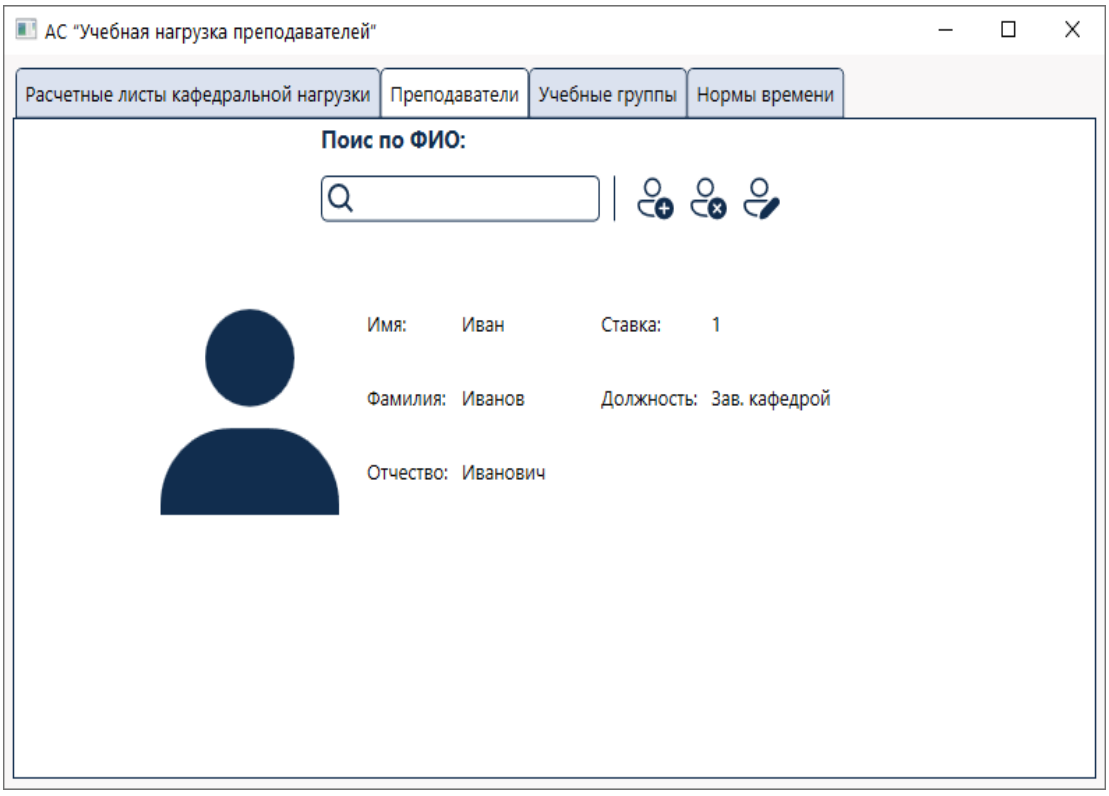


Рисунок 1.28– Интерфейс для взаимодействия с преподавателями

Для добавления нового преподавателя необходимо нажать на кнопку, показанную на рисунке 1.29.

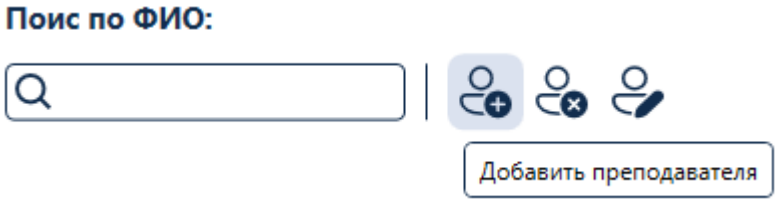


Рисунок 1.29 – Кнопка для добавления преподавателя

После нажатия кнопки для добавления преподавателя открывается диалоговое окно для заполнения информации о новом преподаватели. Вид диалогового окна показан на рисунке 1.30. В данном окне необходимо заполнить поля для фамилии имени отчества, выбрать одну из предоставляемых должностей, выбрать ставку, указать дату рождения. Выбор фотографии является необязательным.

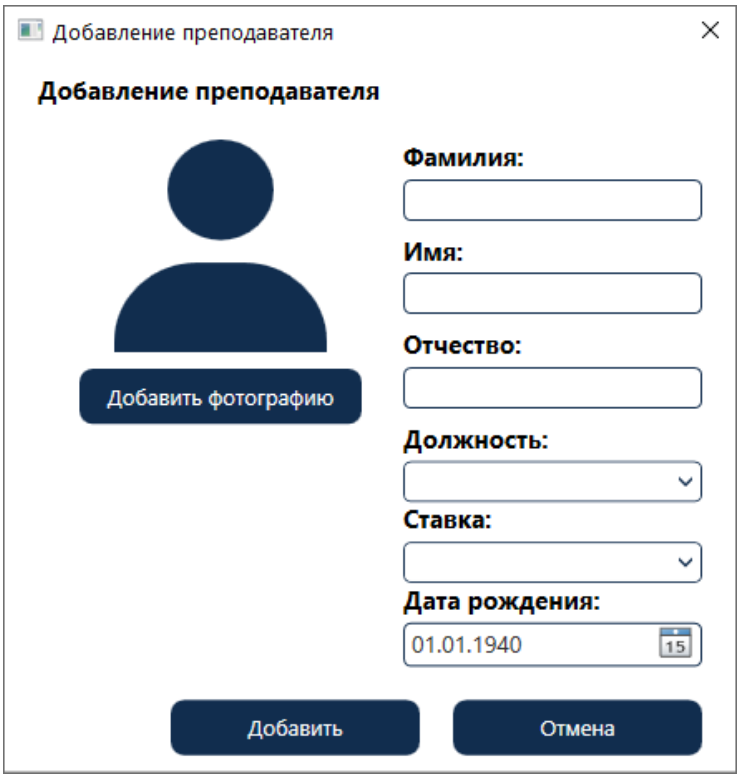


Рисунок 1.30 – Диалоговое окно для заполнения информации о новом преподавателе

После успешного добавления появляется окно с сообщением. Вид сообщения представлен на рисунке 1.31

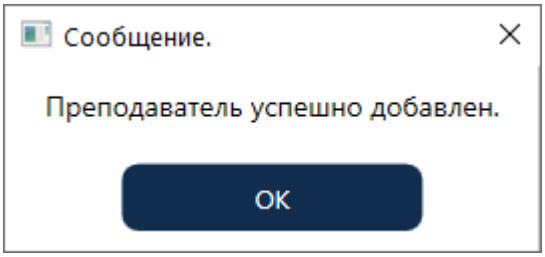


Рисунок 1.31 – Сообщение об успешном добавлении преподавателя

Для удаления информации о преподавателе необходимо выбрать нужного преподавателя из списка, и нажать кнопку для удаления. Процесс удаления показан на рисунке 1.32.

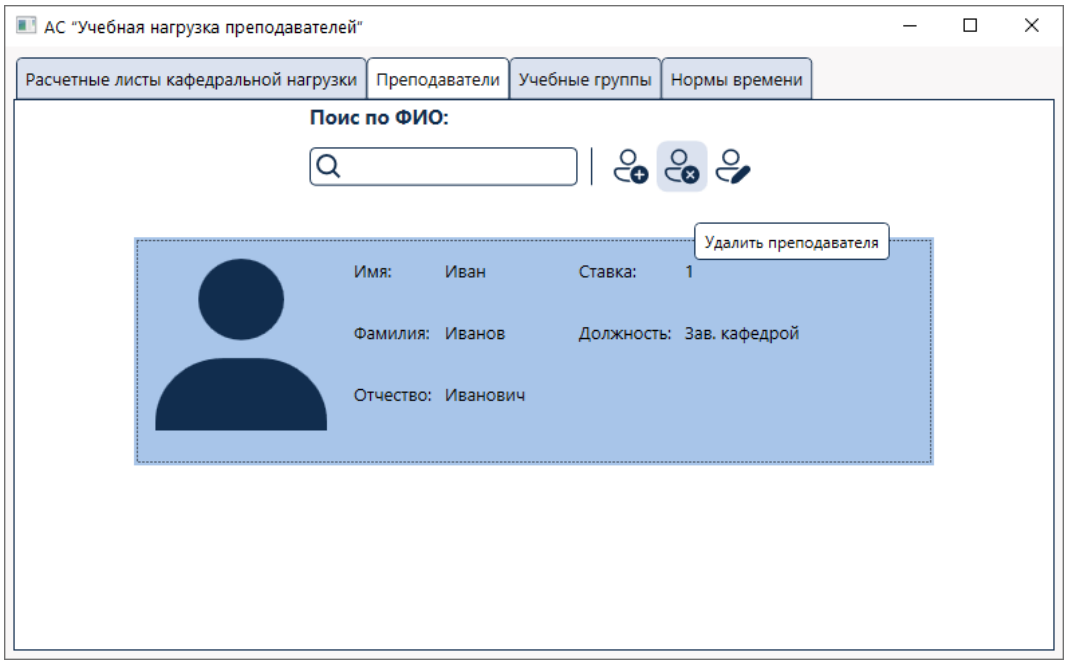


Рисунок 1.32 – Процесс удаления преподавателя

После нажатия кнопки появляется окно с подтверждением, изображенное на рисунке 1.33. Для подтверждения операции удаления необходимо нажать кнопку «ОК». Для отмены операции необходимо нажать кнопку «Отмена».

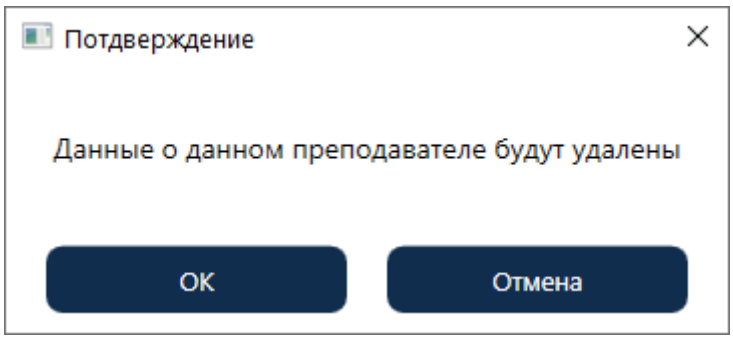


Рисунок 1.33 – Окно для подтверждения удаления

Для редактирования информации о преподаватели необходимо выбрать нужного преподавателя из списка, и нажать кнопку для редактирования. Процесс редактирования показан на рисунке 1.34.

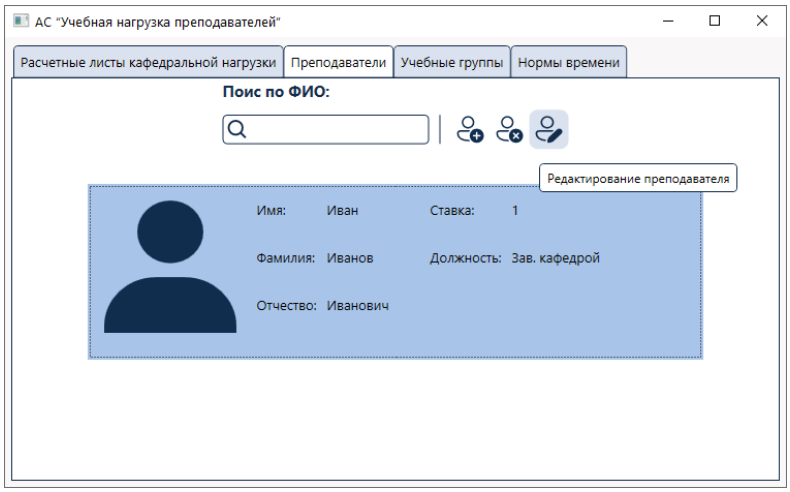


Рисунок 1.34– Процесс редактирования преподавателя

После нажатия кнопки редактирования появляется диалоговое окно для редактирования информации о выбранном преподаватели. Пользователь может отменить операцию редактирования нажатием кнопки «Отмена». Диалоговое окно редактирование представлено на рисунке 1.35.

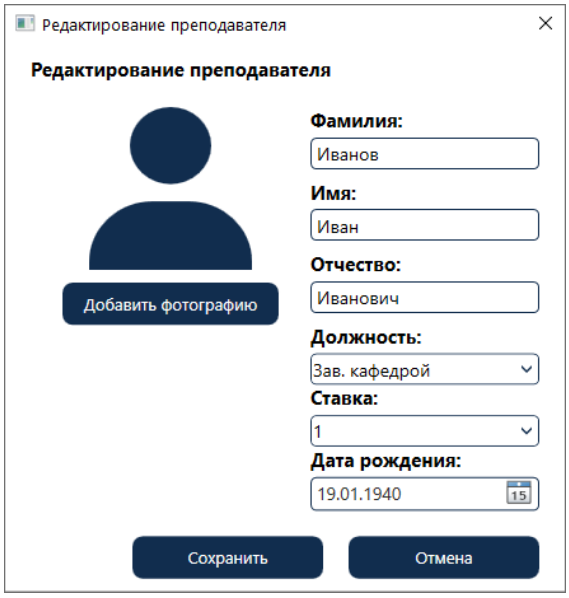


Рисунок 1.35 – Диалоговое окно для редактирования данных о преподавателе

Для взаимодействия с преподавателями, пользователю необходимо перейти по вкладке «Учебные группы». Графический интерфейс с информацией об учебных группах представлен на рисунке 1.36. Процесс

добавления новой учебной группы, редактирования и удаления выбранной учебной группы происходит аналогичным способом, как и для преподавателей. Имеется возможность производить поиск по названию группы.

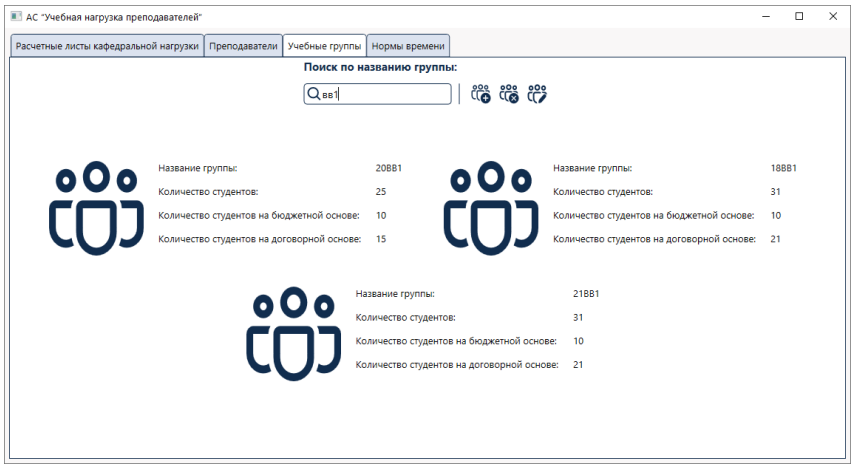


Рисунок 1.36 – Интерфейс для взаимодействия с учебными группами

2 Охрана труда

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

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

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

– правильная организация рабочего места;

– освещенность рабочего места;

– соблюдение правил электробезопасности;

– соблюдение правил пожарной безопасности:

– режим труда и отдыха.

2.1.1 Оборудование рабочего места

Ниже представлены основные требования, которым должно удовлетворять рабочее место:

– расстояние от глаз до экрана должно быть в пределах 600 - 700 мм;

– монитор должен располагаться ниже уровня глаз. Угол между направлением при взгляде прямо и направлением на центр экрана должен составлять 10-15 градусов;

– рабочий стол должен регулироваться по высоте в пределах 680 - 800 мм, при отсутствии такой возможности его высота должна составлять 725мм;

– клавиатуру следует располагать на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной регулируемой по высоте рабочей поверхности, отделенной от основной столешницы;

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

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

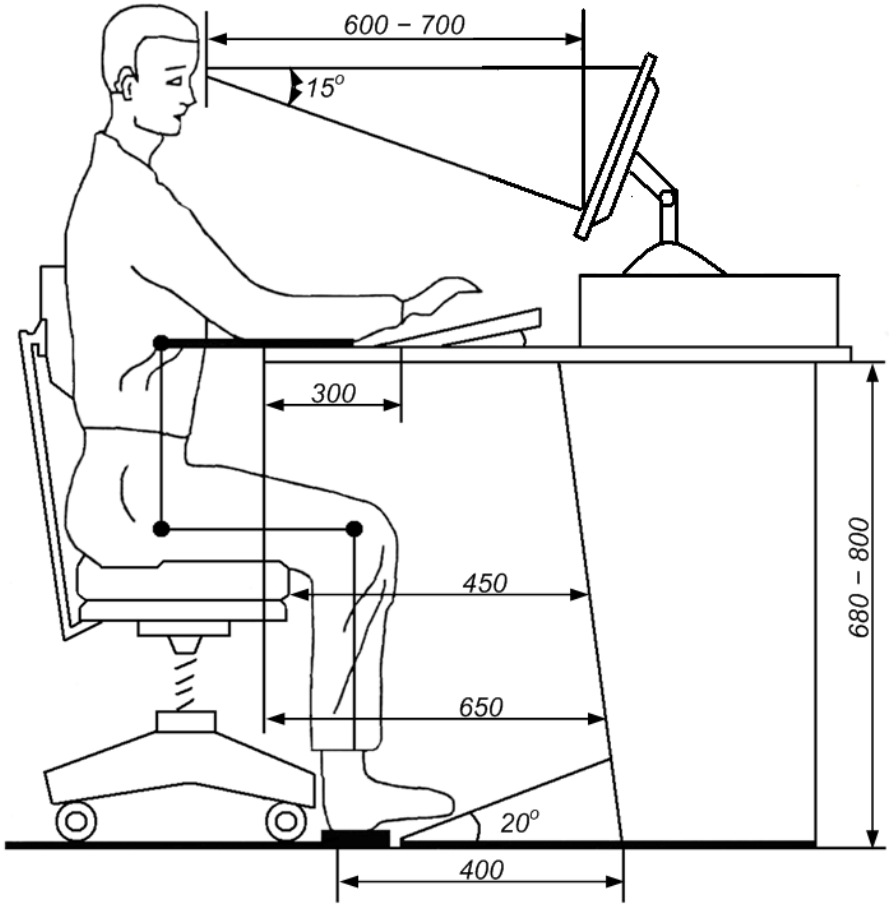


Рисунок 2.1 – Организация рабочего места

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

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

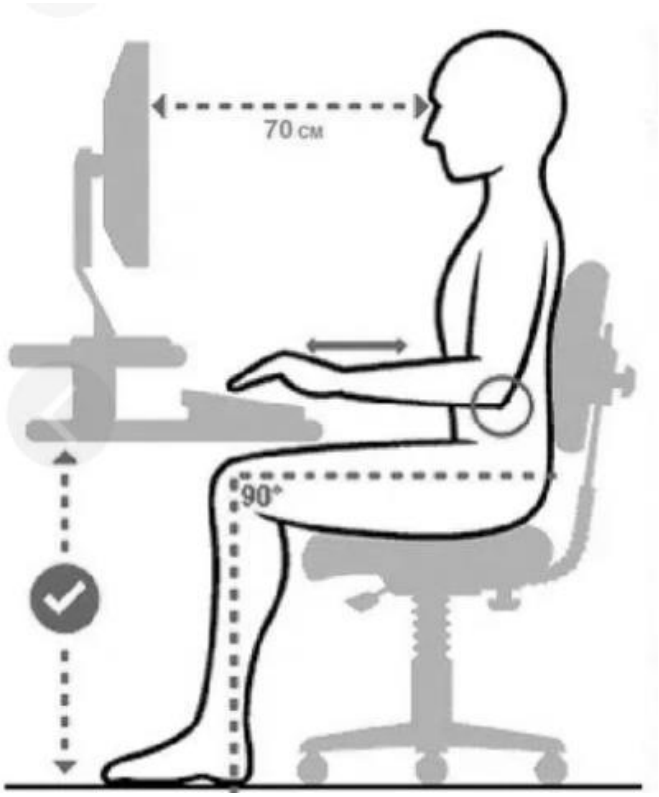


Рисунок 2.2 – Правильная посадка за ПК

Посадка должна обеспечивать выполнение следующих условий:

– запястья должны находиться в прямом положении;

– плечи находятся в расслабленном состоянии;

– локти должны полностью лежать на столе, не свисая и располагаться перпендикулярно полу (угол в локтевых суставах должен составлять 90-100 градусов).

2.1.2 Освещенность рабочего места

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

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

Для освещения используются в основном потолочные или встроенные светильники с люминесцентными лампами. Для исключения засвечивания экранов прямыми световыми потоками светильники общего освещения располагают сбоку от рабочего места, параллельно линии зрения оператора.

2.3 Электробезопасность

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

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

— все устройства должны питаться от одной фазы электрической сети;

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

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

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

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

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

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

2.1.3 Пожарная безопасность

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

– короткие замыкания в проводках и электрических устройствах;

– воспламенение горючих материалов вблизи от электрических устройств;

– большие переходные сопротивления в местах контактных соединений.

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

– проходы в помещения и запасные выходы не должны быть загромождены посторонними предметами, которые препятствуют свободному прохождению;

– к средствам пожаротушения всегда должен быть свободный доступ;

– система вентиляции должна быть оборудована устройствам, обеспечивающим автоматическое отключение ее при пожаре, а также огне задерживающим клапаном;

– установленная автоматическая пожарная сигнализация и средства пожаротушения должны содержаться в исправном состоянии и в

соответствии с типовыми правилами технического содержания установок пожарной автоматики.

2.1.4 Режим труда и отдыха

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

Существуют следующие виды трудовой деятельности связанные с персональным компьютером:

– группа А – работа по считыванию информации с экрана видео дисплейного терминала с предварительным запросом;

– группа Б – работа по вводу информации;

– группа В – творческая работа в режиме диалога с ПЭВМ.

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

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

– для группы А – по суммарному числу считываемых знаков за рабочую смену (не более 60000 знаков за смену);

– для группы Б – по суммарному числу считываемых или вводимых знаков за рабочую смену (не более 40000 знаков за смену);

– для группы В – по суммарному времени непосредственной работы с компьютером за рабочую смену (не более 6 часов за смену).

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

Таблица 2.1 – Время перерывов

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

Заключение

В ходе выполнения выпускной квалификационной работы была разработана автоматизированная система планирования учебной нагрузки преподавателей на языке *C#* . Графический интерфейс был реализован с помощью фреймворка *WPF*.

Разработанная система позволяет импортировать информацию о учебных дисциплинах из расчетного листа кафедральной нагрузки, и производить планирование учебной нагрузки для преподавателей. Результат планирование можно экспортировать из приложения в файл формата *xlsx*.

Разработанное приложение также позволяет автоматизировать расстановку формул, заполнение таблиц и редактирование внешнего вида отдельных ячеек.

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

Приложение «Автоматизированная система планирования учебной нагрузки преподавателей» было выдвинуто на регистрацию ЭВМ.

В дальнейшем можно усовершенствовать разработанную автоматизированную систему и добавить взаимодействие с базой данных, для хранения информации о преподавателях, учебных группах. Улучшить графический интерфейс, добавив отдельные вкладки для информации об отдельных преподавателях со всеми учебными дисциплинами, к которым они назначены. Добавить возможность работы с расчетным листом кафедральной нагрузки для магистрантов и для заочных форм обучения. Реализовать процесс работы с несколькими расчетными листами одновременно. Разработать функционал по сохранению информации в процессе работы программы. Также можно автоматизировать процесс формирования поручений для составления расписания.

Список использованных источников

1. mmis.ru: [Электронный ресурс]. URL: https://www.mmis.ru/programs/nagruzka (дата обращения: 26.04.2022)
2. snauka.ru.: [Электронный ресурс]. URL: https://web.snauka.ru/issues/2015/11/58943 (дата обращения: 27.04.2022)
3. Натан А. *WPF* 4. Подробное руководство. - Пер. с англ. - СПб.: Символ-Плюс, 2011. - 880 с.
4. prismlibrary.com: [Электронный ресурс]. URL: https://prismlibrary.com/#home (дата обращения:28.04.2022)
5. Зиборов, В. *Visual C#* 2010 на примерах / Виктор Зиборов. - М.: БХВ-Петербург, 2011. - 432 *с*.
6. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2018 — 1328 с.
7. Дж. Рамбо, М. Блаха UML 2.0. Объектно-ориентированное моделирование и разработка, 2–е издание. – СПБ.: Питер, 2007 — 544 с.

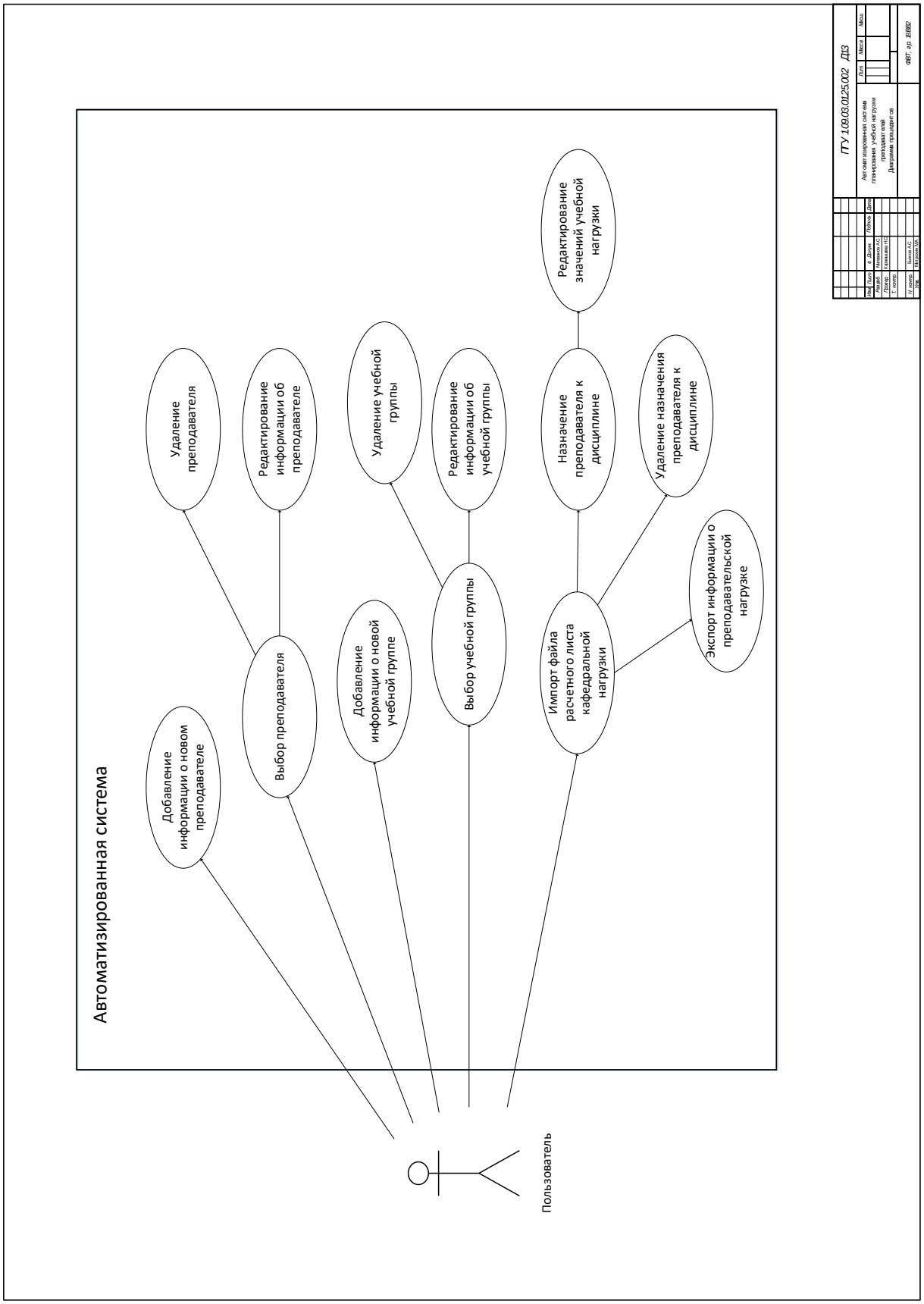
Приложение А

(Обязательное)

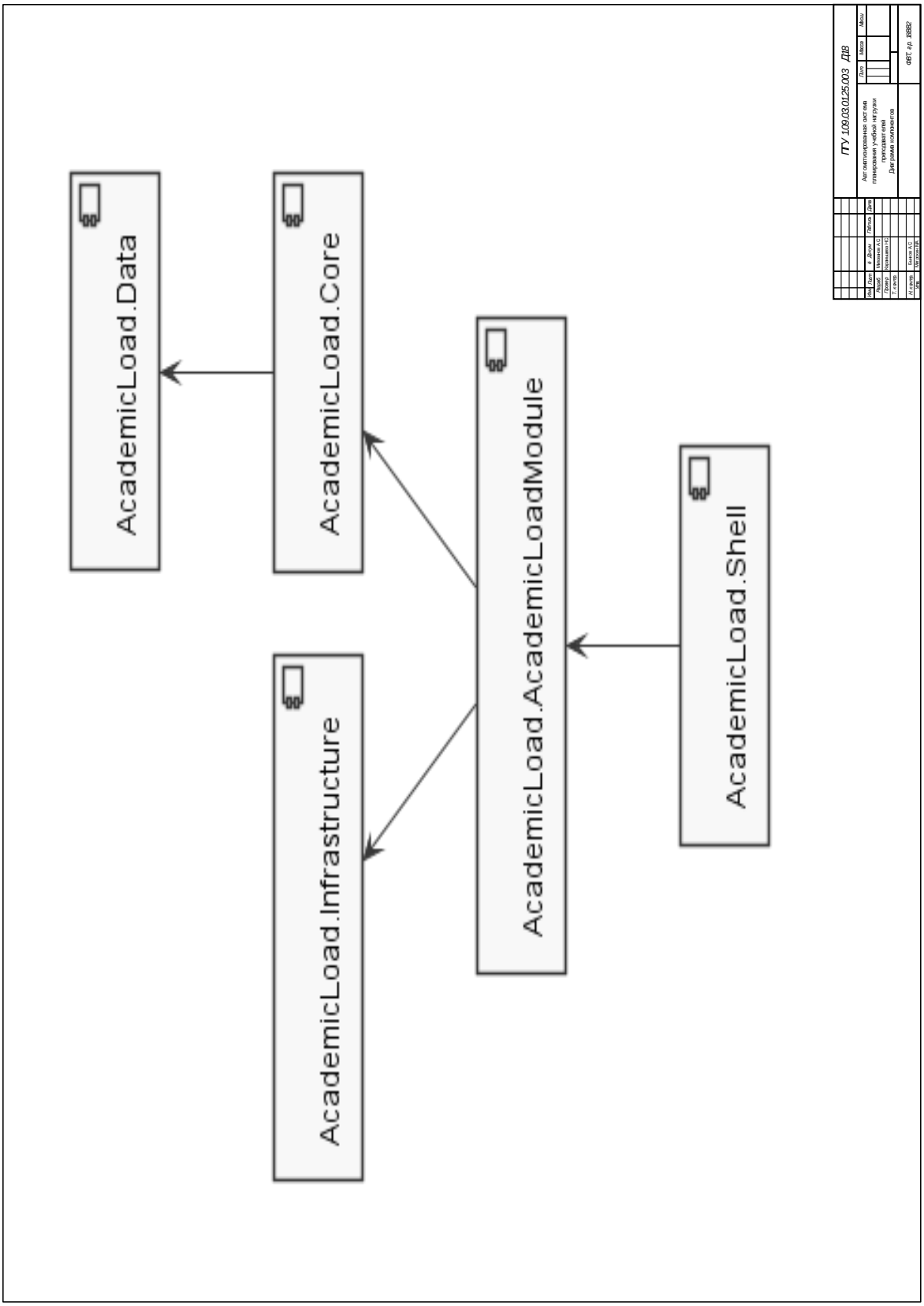
Автоматизированная система планирования учебной нагрузки преподавателей

UML диаграммы

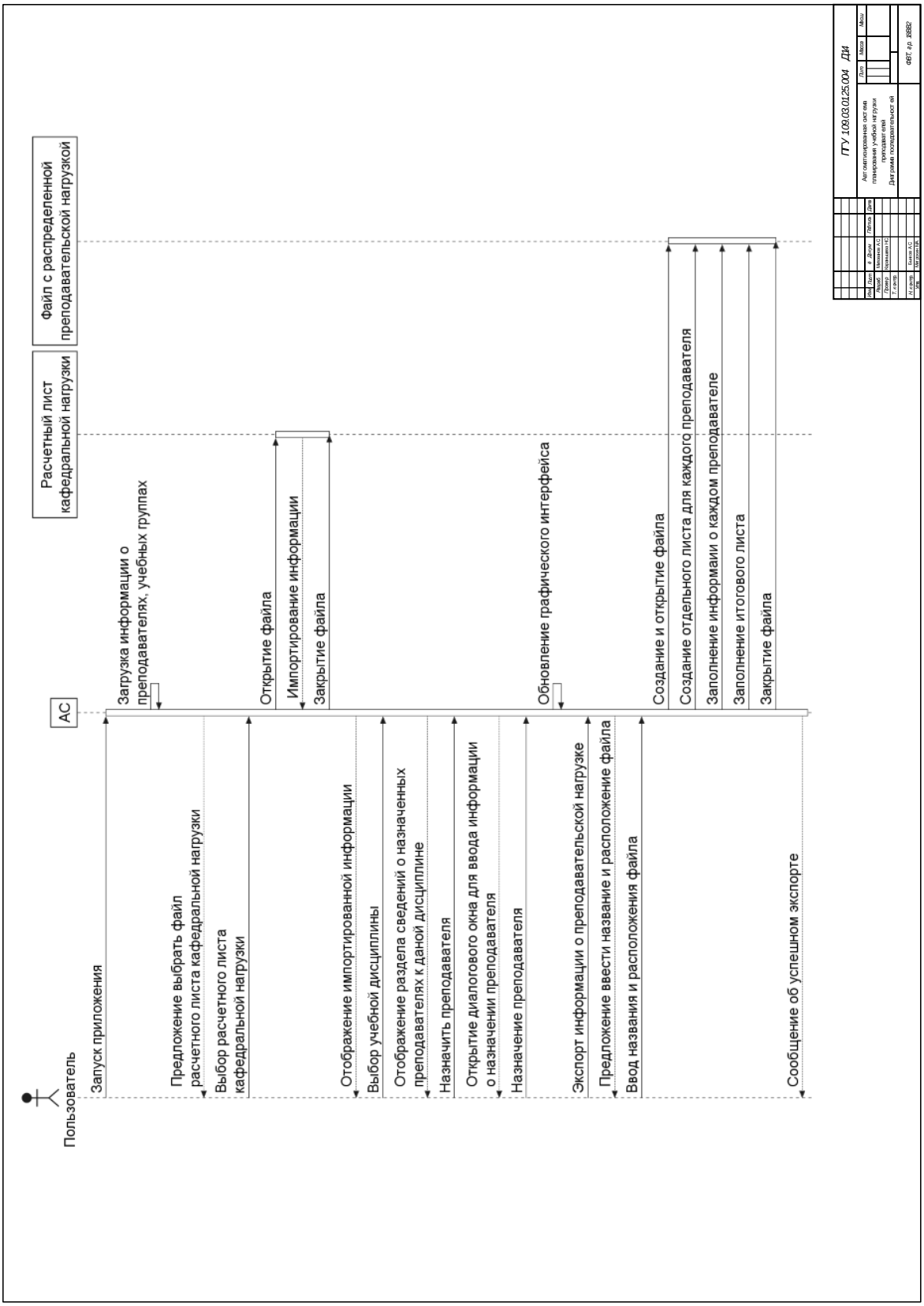
Приложение А.1 – UML – диаграмма прецедентов



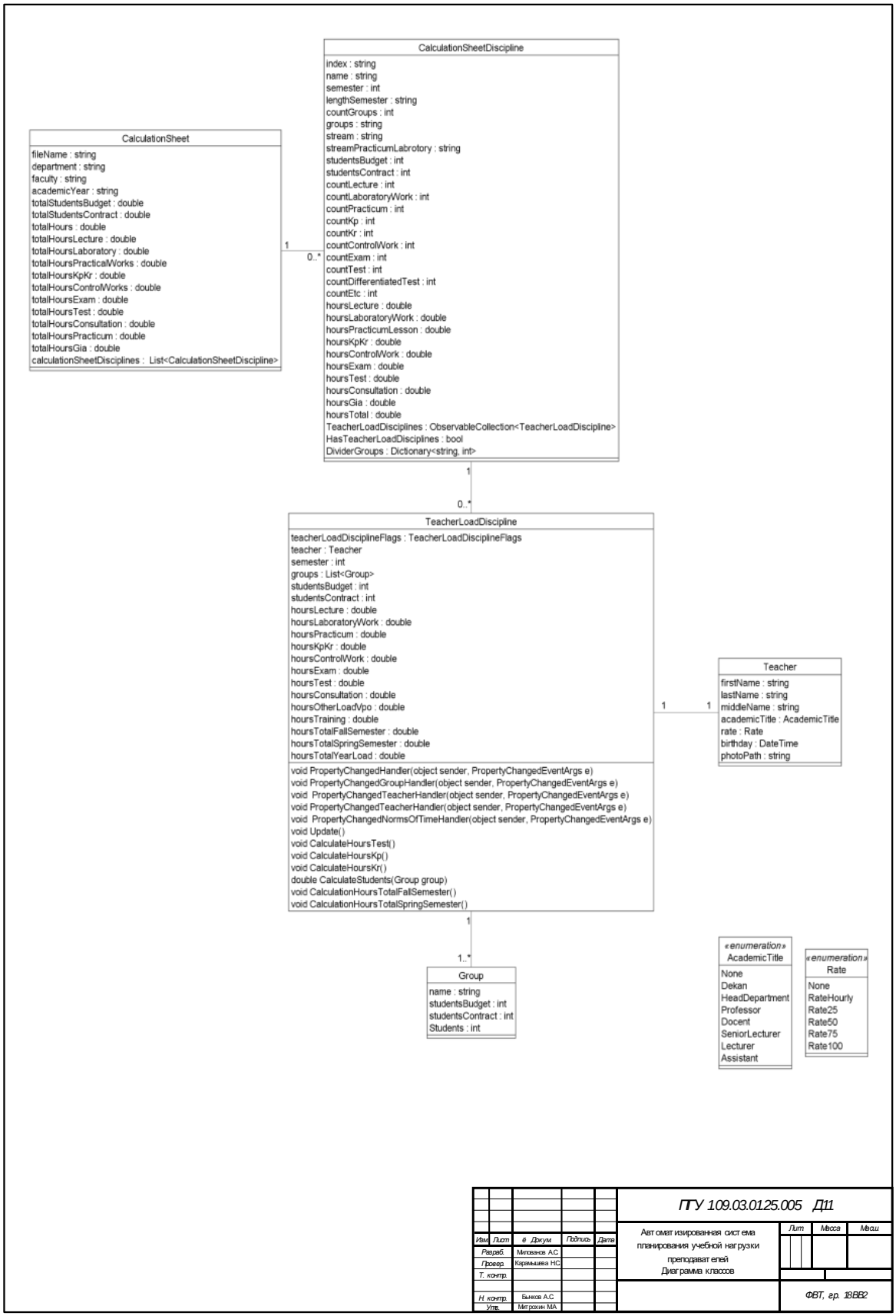
Приложение А.2 – UML – диаграмма компонентов



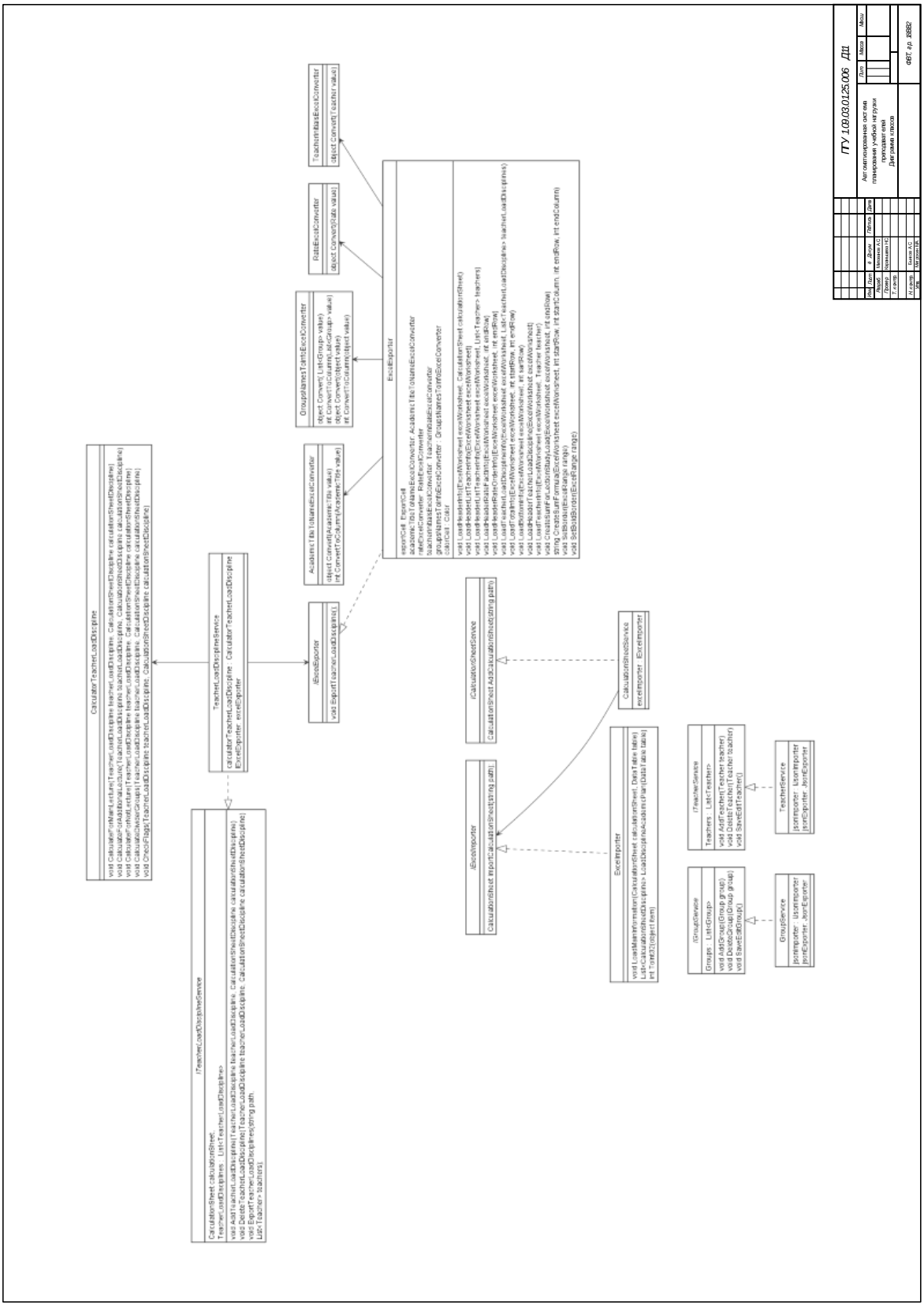
Приложение А.3 – UML – диаграмма последовательностей



Приложение А.4 – UML – диаграмма классов моделей



Приложение А.5 – UML – диаграмма классов слоя «*CORE*»



Приложение Б

(Обязательное)

Автоматизированная система планирования учебной нагрузки преподавателей

Листинг программы

Файл «*Teacher.cs*»

```
using System;
using System.IO;
using Data.Enums;
using Prism.Mvvm;

namespace Data.Models
{
    /// <summary>
    /// Модель преподователя.
    /// </summary>
    public class Teacher : BindableBase
    {
        private string firstName;
        private string lastName;
        private string middleName;
        private AcademicTitle academicTitle;
        private Rate rate;
        private DateTime birthday;
        private string photoPath;

        /// <summary>
        /// Имя.
        /// </summary>
        public string FirstName
        {
            get => firstName;
            set => SetProperty(ref firstName, value);
        }

        /// <summary>
        /// Фамилия.
        /// </summary>
        public string LastName
        {
            get => lastName;
            set => SetProperty(ref lastName, value);
        }

        /// <summary>
        /// Отчество.
        /// </summary>
        public string MiddleName
        {
            get => middleName;
            set => SetProperty(ref middleName, value);
        }

        /// <summary>
        /// Должность.
        /// </summary>
```

```
        public AcademicTitle AcademicTitle
        {
            get => academicTitle;
            set => SetProperty(ref academicTitle, value);
        }

        /// <summary>
        /// Ставка.
        /// </summary>
        public Rate Rate
        {
            get => rate;
            set => SetProperty(ref rate, value);
        }

        /// <summary>
        /// Дата рождения.
        /// </summary>
        public DateTime Birthday
        {
            get => birthday;
            set => SetProperty(ref birthday, value);
        }

        /// <summary>
        /// Путь к фотографии.
        /// </summary>
        public string PhotoPath
        {
            get => photoPath;
            set
            {
                if (File.Exists(value))
                {
                    SetProperty(ref photoPath, value);
                    return;
                }

                SetProperty(ref photoPath, null);
            }
        }
    }
}
```

Файл «*CalculationSheet.cs*»

```
using Prism.Mvvm;
using System.Collections.Generic;

namespace Data.Models
{
    /// <summary>
    /// Модель расчетного листа кафедральной нагрузки.
    /// </summary>
    public class CalculationSheet : BindableBase
    {
        /// <summary>
        /// Имя файла.
        /// </summary>
        public string FileName { get; set; }

        /// <summary>
        /// Кафедра.
        /// </summary>
        public string Department { get; set; }

        /// <summary>
        /// Факультет.
        /// </summary>
        public string Faculty { get; set; }

        /// <summary>
        /// Учебный год.
        /// </summary>
        public string AcademicYear { get; set; }

        /// <summary>
        /// Всего бюджетников.
        /// </summary>
        public double TotalStudentsBudget { get; set; }

        /// <summary>
        /// Всего договорников.
        /// </summary>
        public double TotalStudentsContract { get; set; }

        /// <summary>
        /// Всего часов.
        /// </summary>
        public double TotalHours { get; set; }

        #region TotalHours
        /// <summary>
        /// Всего часов лекций.
        /// </summary>
```

```
        public double TotalHoursLecture { get; set; }

        /// <summary>
        /// Всего часов лабораторных.
        /// </summary>
        public double TotalHoursLaboratory { get; set; }

        /// <summary>
        /// Всего часов практических занятий.
        /// </summary>
        public double TotalHoursPracticalWorks { get; set; }

        /// <summary>
        /// Всего часов КП,КР.
        /// </summary>
        public double TotalHoursKpKr { get; set; }

        /// <summary>
        /// Всего часов контрольных работ.
        /// </summary>
        public double TotalHoursControlWorks { get; set; }

        /// <summary>
        /// Всего часов экзаменов.
        /// </summary>
        public double TotalHoursExam { get; set; }

        /// <summary>
        /// Всего засов зачетов.
        /// </summary>
        public double TotalHoursTest { get; set; }

        /// <summary>
        /// Всего часов консультаций.
        /// </summary>
        public double TotalHoursConsultation { get; set; }

        /// <summary>
        /// Всего часов пратики
        /// </summary>
        public double TotalHoursPracticum { get; set; }

        /// <summary>
        /// Всего часов ГИА
        /// </summary>
        public double TotalHoursGia { get; set; }
        #endregion

        public List<CalculationSheetDiscipline>
CalculationSheetDisciplines { get; set; }
    }
```

```
}
```

Файл «*CalculationSheetDiscipline.cs*»

```
namespace Data.Models
{
    /// <summary>
    /// Модель для строк в таблице расчетного листа кафедральной
нагрузки.
    /// </summary>
    public partial class CalculationSheetDiscipline
    {
        /// <summary>
        /// Индекс.
        /// </summary>
        public string Index { get; set; }

        /// <summary>
        /// Название.
        /// </summary>
        public string Name { get; set; }

        /// <summary>
        /// Семестр.
        /// </summary>
        public int Semester { get; set; }

        /// <summary>
        /// Длительность семестр.
        /// </summary>
        public string LengthSemester { get; set; }

        /// <summary>
        /// Количество групп
        /// </summary>
        public int CountGroups { get; set; }

        /// <summary>
        /// Учебные групы.
        /// </summary>
        public string Groups { get; set; }

        /// <summary>
        /// Поток.
        /// </summary>
        public string Stream { get; set; }

        /// <summary>
        /// Поток практические + лабораторные.
        /// </summary>
        public string StreamPracticumLabrotory { get; set; }

        /// <summary>
```

```
        /// Количество студентов на бюджетной основе.
        /// </summary>
        public int StudentsBudget { get; set; }

        /// <summary>
        /// Количество студентов на договорной основе.
        /// </summary>
        public int StudentsContract { get; set; }

        #region CountLessons
        /// <summary>
        ///
        /// </summary>
        public int CountLecture { get; set; }

        /// <summary>
        ///
        /// </summary>
        public int CountLaboratoryWork { get; set; }

        /// <summary>
        /// Количество практических занятий.
        /// </summary>
        public int CountPracticum { get; set; }

        /// <summary>
        /// Количество КП
        /// </summary>
        public int CountKp { get; set; }

        /// <summary>
        /// Количество КР
        /// </summary>
        public int CountKr { get; set; }

        /// <summary>
        /// Количество контрольных работ.
        /// </summary>
        public int CountControlWork { get; set; }

        /// <summary>
        ///
        /// </summary>
        public int CountExam { get; set; }

        /// <summary>
        /// Количество зачетов.
        /// </summary>
        public int CountTest { get; set; }

        /// <summary>
```

```
        /// Количество зачетов с оценкой.
        /// </summary>
        public int CountDifferentiatedTest { get; set; }

        /// <summary>
        /// Прочее.
        /// </summary>
        public int CountEtc { get; set; }
        #endregion

        #region Hours
        /// <summary>
        ///
        /// </summary>
        public double HoursLecture { get; set; }

        /// <summary>
        ///
        /// </summary>
        public double HoursLaboratoryWork { get; set; }

        /// <summary>
        /// Часы отведенные для практических занятий.
        /// </summary>
        public double HoursPracticumLesson { get; set; }

        /// <summary>
        /// КП, КР (так называется в таблице).
        /// </summary>
        public double HoursKpKr { get; set; }

        /// <summary>
        /// Часы отведенные для контрольных работ.
        /// </summary>
        public double HoursControlWork { get; set; }

        /// <summary>
        ///
        /// </summary>
        public double HoursExam { get; set; }

        /// <summary>
        /// Часы отведенные для зачета, зачета с оценкой.
        /// </summary>
        public double HoursTest { get; set; }

        /// <summary>
        ///  Часы отведенные для консультации.
        /// </summary>
        public double HoursConsultation { get; set; }
```

```
        /// <summary>
        /// Часы отведенные для Гиа.
        /// </summary>
        public double HoursGia { get; set; }

        /// <summary>
        /// Часы отведенные для Практики (Рук-водитель).
        /// </summary>
        public double HoursPracticum { get; set; }

        /// <summary>
        /// Всего часов.
        /// </summary>
        public double HoursTotal { get; set; }
        #endregion

   private bool hasTeacherLoadDisciplines;
        private ObservableCollection<TeacherLoadDiscipline>
teacherLoadDisciplines =
            new ObservableCollection<TeacherLoadDiscipline>();

        public ObservableCollection<TeacherLoadDiscipline>
TeacherLoadDisciplines
        {
            get => teacherLoadDisciplines;
            set => teacherLoadDisciplines = value;
        }

        public bool HasTeacherLoadDisciplines
        {
            get => hasTeacherLoadDisciplines;
            set => SetProperty(ref hasTeacherLoadDisciplines,
value);

        }

        public Dictionary<string, int> DividerGroups = new
Dictionary<string, int>();
    }
}
```

Файл «ExcelExporter.*cs*»

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Core.Excel.Constants;
using Core.Excel.Converters;
```

```
using Core.Excel.Interfaces;
using Data.Models;
using OfficeOpenXml;
using OfficeOpenXml.Style;

namespace Core.Excel
{
    /// <summary>
    /// Реализация IExcelExporter.
    /// </summary>
    public partial class ExcelExporter : IExcelExporter
    {
        private readonly ExportCell exportCell;
        private readonly AcademicTitleToNameExcelConverter
academicTitleToNameExcelConverter;
        private readonly RateExcelConverter rateExcelConverter;
        private readonly TeacherInitialsExcelConverter
teacherInitialsExcelConverter;
        private readonly GroupsNamesToInfoExcelConverter
groupsNamesToInfoExcelConverter;
        private Color colorCell = Color.Blue;

        /// <summary>
        /// ctor.
        /// </summary>
        public ExcelExporter()
        {
            ///TODO перенести в файл конфига (чтобы не было
исключения про лицензию)
            ExcelPackage.LicenseContext =
LicenseContext.NonCommercial;

            exportCell = new ExportCell();
            academicTitleToNameExcelConverter = new
AcademicTitleToNameExcelConverter();
            rateExcelConverter = new RateExcelConverter();
            teacherInitialsExcelConverter = new
TeacherInitialsExcelConverter();
            groupsNamesToInfoExcelConverter = new
GroupsNamesToInfoExcelConverter();
        }

        /// <inheritdoc/>
        public void ExportTeacherLoadDiscipline(string path,
            CalculationSheet calculationSheet,
            List<Teacher> teachers,
            List<TeacherLoadDiscipline> teacherLoadDisciplines)
        {
            var newFile = new FileInfo(path);
            using (ExcelPackage excelPackage = new
ExcelPackage(newFile))
```

```
            {
                excelPackage.Workbook.Properties.Created =
DateTime.Now;

                ExcelWorksheet mainWorksheet =
excelPackage.Workbook.Worksheets.Add("Итого");
                LoadHeaderInfo(mainWorksheet, calculationSheet);
                LoadHeaderListTeacherInfo(mainWorksheet);

                foreach (Teacher teacher in teachers)
                {
                    var worksheet =
excelPackage.Workbook.Worksheets.Add(teacher.LastName);

                    LoadHeaderInfo(worksheet, calculationSheet);
                    LoadTeacherInfo(worksheet, teacher);
                    LoadHeaderTeacherLoadDiscipline(worksheet);
                    LoadTeacherLoadDisciplineInfo(worksheet,
teacherLoadDisciplines.Where(x => x.Teacher ==
teacher).ToList());
                }

                LoadHeaderListTeacherInfo(mainWorksheet,
teachers);

                excelPackage.Workbook.Styles.UpdateXml();
                excelPackage.Save();
            }
        }

        private void LoadHeaderInfo(ExcelWorksheet
excelWorksheet, CalculationSheet calculationSheet)
        {
            excelWorksheet.Cells[exportCell.NamePzuStart.row,
                exportCell.NamePzuStart.column,
                exportCell.NamePzuEnd.row,
                exportCell.NamePzuEnd.column].Merge = true;


excelWorksheet.Cells[exportCell.DistributionTeacherLoadStart.row
,
                exportCell.DistributionTeacherLoadStart.column,
                exportCell.DistributionTeacherLoadEnd.row,

exportCell.DistributionTeacherLoadEnd.column].Merge = true;

            excelWorksheet.Cells[exportCell.NamePzuStart.row,
exportCell.NamePzuStart.column].Value = "Пензенский
Государственный Университет";
```

```
            excelWorksheet.Cells[exportCell.NamePzuStart.row,
exportCell.NamePzuStart.column].Style.HorizontalAlignment =
ExcelHorizontalAlignment.Center;


excelWorksheet.Cells[exportCell.DistributionTeacherLoadStart.row
, exportCell.DistributionTeacherLoadStart.column].Value =
"Распределение учебной нагрузки";

excelWorksheet.Cells[exportCell.DistributionTeacherLoadStart.row
,
exportCell.DistributionTeacherLoadStart.column].Style.Horizontal
Alignment = ExcelHorizontalAlignment.Center;

            excelWorksheet.Cells[exportCell.Department.row,
exportCell.Department.column].Value = "Кафедра";
            excelWorksheet.Cells[exportCell.Department.row,
exportCell.Department.column + 1].Value =
calculationSheet.Department;

            excelWorksheet.Cells[exportCell.AcademicYear.row,
exportCell.AcademicYear.column].Value = "Учебный год";
            excelWorksheet.Cells[exportCell.AcademicYear.row,
exportCell.AcademicYear.column + 1].Value =
calculationSheet.AcademicYear;

            excelWorksheet.Cells[exportCell.CfYearLoad.row,
exportCell.CfYearLoad.column].Value = "ср.год.нагрузка";
            //TODO вынести Value в класс
            excelWorksheet.Cells[exportCell.CfYearLoad.row,
exportCell.CfYearLoad.column + 1].Value = 850;
            excelWorksheet.Cells[exportCell.CfYearLoad.row,
exportCell.CfYearLoad.column +
1].Style.Font.Color.SetColor(colorCell);
        }

        private void LoadHeaderListTeacherInfo(ExcelWorksheet
excelWorksheet)
        {
            int row =
exportCell.MainSheetStartHeaderListTeacher.row;
            int column =
exportCell.MainSheetStartHeaderListTeacher.column;

            excelWorksheet.Cells[row, column].Value = "Ф.И.О.";
            excelWorksheet.Cells[row, column + 1].Value =
"декан";
            excelWorksheet.Cells[row, column + 2].Value =
"зав.каф.";
            excelWorksheet.Cells[row, column + 3].Value =
"проф.";
```

```
            excelWorksheet.Cells[row, column + 4].Value =
"доц.";
            excelWorksheet.Cells[row, column + 5].Value = "ст.
преп.";
            excelWorksheet.Cells[row, column + 6].Value =
"преп.";
            excelWorksheet.Cells[row, column + 7].Value =
"ассистент";
            excelWorksheet.Cells[row, column + 8].Value =
"ставки";
            excelWorksheet.Cells[row, column + 9].Value =
"учебная нагрузка";
            excelWorksheet.Cells[row, column + 10].Value =
"лекционная нагрузка";
            excelWorksheet.Cells[row, column + 11].Value =
"почас. фонд";

            ExcelRange range = excelWorksheet.Cells[row, column,
row, column + 11];
            SetBorder(range);
        }

        private void LoadHeaderListTeacherInfo(ExcelWorksheet
excelWorksheet, List<Teacher> teachers)
        {
            int row = exportCell.MainSheetStartListTeacher.row;
            int column =
exportCell.MainSheetStartListTeacher.column;
            int counter = 1;

            foreach (Teacher teacher in teachers)
            {
                excelWorksheet.Cells[row, column].Value =
counter;
                excelWorksheet.Cells[row, column + 1].Value =
teacherInitialsExcelConverter.Convert(teacher);
                excelWorksheet.Cells[row,
academicTitleToNameExcelConverter.ConvertToColumn(teacher.Academ
icTitle)].Value = rateExcelConverter.Convert(teacher.Rate);
                excelWorksheet.Cells[row, column + 9].Value =
rateExcelConverter.Convert(teacher.Rate);
                excelWorksheet.Cells[row, column + 10].Formula =
$"{teacher.LastName}!R12";
                excelWorksheet.Cells[row, column + 11].Formula =
$"{teacher.LastName}!F12";

                counter++;
                row++;
            }
```

```
            ExcelRange range =
excelWorksheet.Cells[exportCell.MainSheetStartListTeacher.row,
column, row - 1, column + 12];
            SetBorder(range);

            ExcelRange rangeTeacherInitials =
excelWorksheet.Cells[exportCell.MainSheetStartListTeacher.row,
column + 1, row - 1, column + 1];
            SetBoldBorder(rangeTeacherInitials);

            LoadHeaderRateFactInfo(excelWorksheet, row -1);
            LoadHeaderRateOrderInfo(excelWorksheet, row - 1);
            CreateSumForLectionStudyLoad(excelWorksheet, row -
1);
        }

        private void LoadHeaderRateFactInfo(ExcelWorksheet
excelWorksheet, int endRow)
        {
            int row = exportCell.MainSheetStartRateOrder.row;
            int column =
exportCell.MainSheetStartRateOrder.column;

            excelWorksheet.Cells[row, column].Value = "Ставки по
приказу";

            ExcelRange range = excelWorksheet.Cells[row, column,
row, column + 8];
            SetBorder(range);

            int rowSum =
exportCell.MainSheetStartHeaderListTeacher.row + 1;
            int columSum =
exportCell.MainSheetStartHeaderListTeacher.column + 1;
            column += 1;

            for (int i = 0; i < 8; i++)
            {
                excelWorksheet.Cells[row, column + i].Formula =
CreateSumFormula(excelWorksheet, rowSum, columSum + i, endRow,
columSum + i);
            }
        }

        private void LoadHeaderRateOrderInfo(ExcelWorksheet
excelWorksheet, int endRow)
        {
            int row = exportCell.MainSheetStartRateFact.row;
            int column =
exportCell.MainSheetStartRateFact.column;
```

```
            excelWorksheet.Cells[row, column].Value = "Ставки по
факту";

            ExcelRange range = excelWorksheet.Cells[row, column,
row, column + 8];
            SetBorder(range);

            int rowSum =
exportCell.MainSheetStartHeaderListTeacher.row + 1;
            int columSum =
exportCell.MainSheetStartHeaderListTeacher.column + 1;
            column += 1;

            for (int i = 0; i < 8; i++)
            {
                excelWorksheet.Cells[row, column + i].Formula =
excelWorksheet.Cells[row, column].Formula =
CreateSumFormula(excelWorksheet, rowSum, columSum + i, endRow,
columSum + i);
            }
        }

        private void CreateSumForLectionStudyLoad(ExcelWorksheet
excelWorksheet, int endRow)
        {

            ExcelRange range = excelWorksheet.Cells[11, 11, 11,
12];
            SetBorder(range);

            int row = 13;
            int column = 11;
            excelWorksheet.Cells[11, 11].Formula =
CreateSumFormula(excelWorksheet, row, column, endRow, column);

            row = 13;
            column = 12;
            excelWorksheet.Cells[11, 12].Formula =
CreateSumFormula(excelWorksheet, row, column, endRow, column);
        }

        private string CreateSumFormula(ExcelWorksheet
excelWorksheet, int startRow, int startColumn, int endRow, int
endColumn)
        {
            return $"SUM({excelWorksheet.Cells[startRow,
startColumn].Address}:{excelWorksheet.Cells[endRow,
endColumn].Address})";
        }

        private void SetBorder(ExcelRange range)
```

```
        {
            range.Style.Border.Top.Style =
ExcelBorderStyle.Thin;
            range.Style.Border.Bottom.Style =
ExcelBorderStyle.Thin;
            range.Style.Border.Left.Style =
ExcelBorderStyle.Thin;
            range.Style.Border.Right.Style =
ExcelBorderStyle.Thin;
        }

        private void SetBoldBorder(ExcelRange range)
        {
            range.Style.Border.Top.Style =
ExcelBorderStyle.Thick;
            range.Style.Border.Bottom.Style =
ExcelBorderStyle.Thick;
            range.Style.Border.Left.Style =
ExcelBorderStyle.Thick;
            range.Style.Border.Right.Style =
ExcelBorderStyle.Thick;
        }
private void LoadTeacherInfo(ExcelWorksheet excelWorksheet,
Teacher teacher)
        {
            excelWorksheet.Cells[exportCell.Initials.row,
                exportCell.Initials.column + 1,
                exportCell.Initials.row,
                exportCell.Initials.column + 2].Merge = true;
            excelWorksheet.Cells[exportCell.AcademicTitle.row,
                exportCell.AcademicTitle.column,
                exportCell.AcademicTitle.row,
                exportCell.AcademicTitle.column + 1].Merge =
true;

            excelWorksheet.Cells[exportCell.Approve.row,
exportCell.Approve.column].Value = "УТВЕРЖДАЮ";
            excelWorksheet.Cells[exportCell.Prorector.row,
exportCell.Prorector.column].Value = "Проректор по учебной
работе";

            excelWorksheet.Cells[exportCell.Initials.row,
exportCell.Initials.column].Value = "Ф.И.О";
            excelWorksheet.Cells[exportCell.Initials.row,
exportCell.Initials.column].Style.Font.Color.SetColor(colorCell)
;
            excelWorksheet.Cells[exportCell.Initials.row,
exportCell.Initials.column + 1].Value =
teacherInitialsExcelConverter.Convert(teacher);
```

```
            excelWorksheet.Cells[exportCell.Initials.row,
exportCell.Initials.column].Style.HorizontalAlignment =
ExcelHorizontalAlignment.Center;

            excelWorksheet.Cells[exportCell.AcademicTitle.row,
exportCell.AcademicTitle.column].Value = "должность";
            excelWorksheet.Cells[exportCell.AcademicTitle.row,
exportCell.AcademicTitle.column].Style.Font.Color.SetColor(color
Cell);
            excelWorksheet.Cells[exportCell.AcademicTitle.row,
exportCell.AcademicTitle.column + 2].Value =
academicTitleToNameExcelConverter.Convert(teacher.AcademicTitle)
;

            excelWorksheet.Cells[exportCell.Rate.row,
exportCell.Rate.column].Value = "ставка";
            excelWorksheet.Cells[exportCell.Rate.row,
exportCell.Rate.column].Style.Font.Color.SetColor(colorCell);
            excelWorksheet.Cells[exportCell.Rate.row,
exportCell.Rate.column + 1].Value =
rateExcelConverter.Convert(teacher.Rate);
        }

        private void
LoadHeaderTeacherLoadDiscipline(ExcelWorksheet excelWorksheet)
        {
            int row =
exportCell.HeaderTeacherLoadDiscipline.row;
            int column =
exportCell.HeaderTeacherLoadDiscipline.column;

            excelWorksheet.Cells[row, column].Value =
"Дисциплины";
            excelWorksheet.Cells[row, column + 1].Value =
"Семестр";
            excelWorksheet.Cells[row, column + 2].Value =
"Группы";
            excelWorksheet.Cells[row, column + 3].Value =
"Студенты (бюд)";
            excelWorksheet.Cells[row, column + 4].Value =
"Студенты (дог)";
            excelWorksheet.Cells[row, column + 5].Value =
"Лекции";
            excelWorksheet.Cells[row, column + 6].Value =
"Лабораторные";
            excelWorksheet.Cells[row, column + 7].Value =
"Практические";
            excelWorksheet.Cells[row, column + 8].Value = "КП /
КР";
            excelWorksheet.Cells[row, column + 9].Value =
"контр./раб. (ЗФ)";
```

```
            excelWorksheet.Cells[row, column + 10].Value =
"Экзамены";
            excelWorksheet.Cells[row, column + 11].Value =
"Зачеты";
            excelWorksheet.Cells[row, column + 12].Value =
"консультации";
            excelWorksheet.Cells[row, column + 13].Value =
"Прочая нагрузка по ВПО";
            excelWorksheet.Cells[row, column + 14].Value =
"Подготовка аспирантов, интернов, ординаторов";
            excelWorksheet.Cells[row, column + 15].Value =
"Всего  осенний семестр";
            excelWorksheet.Cells[row, column + 16].Value =
"Всего весенний семестр";
            excelWorksheet.Cells[row, column + 17].Value =
"Всего за год нагрузка";

            using (ExcelRange range = excelWorksheet.Cells[row,
column + 3, row, column + 17])
            {
                range.Style.TextRotation = 90;
                range.Style.VerticalAlignment =
ExcelVerticalAlignment.Center;
            }
        }

        private void
LoadTeacherLoadDisciplineInfo(ExcelWorksheet excelWorksheet,
List<TeacherLoadDiscipline> teacherLoadDisciplines)
        {
            int row = exportCell.TeacherLoadDiscipline.row;
            int column =
exportCell.TeacherLoadDiscipline.column;

            foreach (TeacherLoadDiscipline variable in
teacherLoadDisciplines)
            {
                excelWorksheet.Cells[row, column].Value =
variable.NameDiscipine;
                excelWorksheet.Cells[row, column + 1].Value =
variable.Semester;
                excelWorksheet.Cells[row, column + 2].Value =
groupsNamesToInfoExcelConverter.Convert(variable.Groups);
                excelWorksheet.Cells[row, column + 3].Value =
variable.StudentsBudget;
                excelWorksheet.Cells[row, column + 4].Value =
variable.StudentsContract;
                excelWorksheet.Cells[row, column + 5].Value =
variable.HoursLecture;
                excelWorksheet.Cells[row, column + 6].Value =
variable.HoursLaboratoryWork;
```

```
                excelWorksheet.Cells[row, column + 7].Value =
variable.HoursPracticum;
                excelWorksheet.Cells[row, column + 8].Value =
variable.HoursKpKr;
                excelWorksheet.Cells[row, column + 9].Value =
variable.HoursControlWork;
                excelWorksheet.Cells[row, column + 10].Value =
variable.HoursExam;
                excelWorksheet.Cells[row, column + 11].Value =
variable.HoursTest;
                excelWorksheet.Cells[row, column + 12].Value =
variable.HoursConsultation;
                excelWorksheet.Cells[row, column + 13].Value =
variable.HoursOtherLoadVpo;
                excelWorksheet.Cells[row, column + 14].Value =
variable.HoursTraining;

                excelWorksheet.Cells[row, column + 15].Formula =
$"IF(ISODD({excelWorksheet.Cells[row, column +
1].Address}),SUM({excelWorksheet.Cells[row, column +
5].Address}:{excelWorksheet.Cells[row, column +
14].Address}),0)";
                excelWorksheet.Cells[row, column + 16].Formula =
$"IF(ISEVEN({excelWorksheet.Cells[row, column +
1].Address}),SUM({excelWorksheet.Cells[row, column +
5].Address}:{excelWorksheet.Cells[row, column +
14].Address}),0)";
                excelWorksheet.Cells[row, column + 17].Formula =
$"SUM({excelWorksheet.Cells[row, column +
15].Address}:{excelWorksheet.Cells[row, column + 16].Address})";

                row++;
            }

            ExcelRange range =
excelWorksheet.Cells[exportCell.HeaderTeacherLoadDiscipline.row,
exportCell.HeaderTeacherLoadDiscipline.column, row - 1, column +
17];
            SetBorder(range);
            LoadTotalInfo(excelWorksheet,
exportCell.TeacherLoadDiscipline.row, row - 1);
            LoadBottomInfo(excelWorksheet, row);
        }

        private void LoadTotalInfo(ExcelWorksheet
excelWorksheet, int startRow, int endRow)
        {
            excelWorksheet.Cells[exportCell.Total.row,
                 exportCell.Total.column,
                 exportCell.Total.row,
                 exportCell.Total.column + 1].Merge = true;
```

```
            excelWorksheet.Cells[exportCell.Total.row,
exportCell.Total.column].Value = "ИТОГО:";

            int row = exportCell.TotalStartFormula.row;
            int column = exportCell.TotalStartFormula.column;

            excelWorksheet.Cells[row, column].Formula =
CreateSumFormula(excelWorksheet, startRow, column, endRow,
column);
            excelWorksheet.Cells[row, column + 1].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 1, endRow,
column + 1);
            excelWorksheet.Cells[row, column + 2].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 2, endRow,
column + 2);
            excelWorksheet.Cells[row, column + 3].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 3, endRow,
column + 3);
            excelWorksheet.Cells[row, column + 4].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 4, endRow,
column + 4);
            excelWorksheet.Cells[row, column + 5].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 5, endRow,
column + 5);
            excelWorksheet.Cells[row, column + 6].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 6, endRow,
column + 6);
            excelWorksheet.Cells[row, column + 7].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 7, endRow,
column + 7);
            excelWorksheet.Cells[row, column + 8].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 8, endRow,
column + 8);
            excelWorksheet.Cells[row, column + 9].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 9, endRow,
column + 9);
            excelWorksheet.Cells[row, column + 10].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 10, endRow,
column + 10);
            excelWorksheet.Cells[row, column + 11].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 11, endRow,
column + 11);
            excelWorksheet.Cells[row, column + 12].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 12, endRow,
column + 12);
            excelWorksheet.Cells[row, column + 13].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 13, endRow,
column + 13);
            excelWorksheet.Cells[row, column + 14].Formula =
CreateSumFormula(excelWorksheet, startRow, column + 14, endRow,
column + 14);
```

```
            ExcelRange range =
excelWorksheet.Cells[exportCell.Total.row,
exportCell.Total.column, exportCell.Total.row,
exportCell.Total.column + 16];
            SetBorder(range);
        }


        private void LoadBottomInfo(ExcelWorksheet
excelWorksheet, int sartRow)
        {
            excelWorksheet.Cells[sartRow, 1].Value = "зав.
кафедрой";
            excelWorksheet.Cells[sartRow, 5].Value = "декан
факультета";
            excelWorksheet.Cells[sartRow, 13].Value = "начальник
УМУ";

            excelWorksheet.Cells[sartRow + 2, 1].Value =
"почас:";
        }
    }
}
```

Файл «ExceImporter.cs»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.IO;
using System.Linq;
using Core.Excel.Interfaces;
using Data.Models;
using ExcelDataReader;

namespace Core.Excel
{
    /// <summary>
    /// Реализация IExcelImporter.
    /// </summary>
    public class ExcelImporter : IExcelImporter
    {
        public CalculationSheet ImportCalculationSheet(string
path)
        {
            CalculationSheet calculationSheet = new
CalculationSheet();

            calculationSheet.FileName =
path.Substring(path.LastIndexOf("\\")+1);
```

```
            var stream = File.Open(path, FileMode.Open,
FileAccess.Read);
            var reader =
ExcelReaderFactory.CreateReader(stream);
            var result = reader.AsDataSet();
            var table =
result.Tables.Cast<DataTable>().ToList().First();

            LoadMainInformation(calculationSheet, table);
            calculationSheet.CalculationSheetDisciplines =
LoadDisciplineAcademicPlan(table);
            return calculationSheet;
        }

        private void LoadMainInformation(CalculationSheet
calculationSheet, DataTable table)
        {
            calculationSheet.Department =
table.Rows[ImportRowColumnConstants.DepartmentRow][ImportRowColu
mnConstants.DepartmentColumn].ToString();
            calculationSheet.Faculty =
table.Rows[ImportRowColumnConstants.FacultyRow][ImportRowColumnC
onstants.FacultyColumn].ToString();
            calculationSheet.AcademicYear =
table.Rows[ImportRowColumnConstants.AcademicYearRow][ImportRowCo
lumnConstants.AcademicYearColumn].ToString();

            calculationSheet.TotalStudentsBudget =
Convert.ToDouble(table.Rows[ImportRowColumnConstants.TotalStuden
tsBudgetRow][ImportRowColumnConstants.TotalStudentsBudgetColumn]
);
            calculationSheet.TotalStudentsContract =
Convert.ToDouble(table.Rows[ImportRowColumnConstants.TotalStuden
tsContractRow][ImportRowColumnConstants.TotalStudentsContractCol
umn]);
            calculationSheet.TotalHours =
Convert.ToDouble(table.Rows[ImportRowColumnConstants.TotalHoursR
ow][ImportRowColumnConstants.TotalHoursColumn]);

            int row =
ImportRowColumnConstants.TotalHoursStartRow;
            int column =
ImportRowColumnConstants.TotalHoursStartColumn;
            calculationSheet.TotalHoursLecture =
Convert.ToDouble(table.Rows[row][column]);
            calculationSheet.TotalHoursLaboratory =
Convert.ToDouble(table.Rows[row][column + 1]);
            calculationSheet.TotalHoursPracticalWorks =
Convert.ToDouble(table.Rows[row][column + 2]);
            calculationSheet.TotalHoursKpKr =
Convert.ToDouble(table.Rows[row][column + 3]);
```

```
            calculationSheet.TotalHoursControlWorks =
Convert.ToDouble(table.Rows[row][column + 4]);
            calculationSheet.TotalHoursExam =
Convert.ToDouble(table.Rows[row][column + 5]);
            calculationSheet.TotalHoursTest =
Convert.ToDouble(table.Rows[row][column + 6]);
            calculationSheet.TotalHoursConsultation =
Convert.ToDouble(table.Rows[row][column + 7]);
            calculationSheet.TotalHoursPracticum =
Convert.ToDouble(table.Rows[row][column + 8]);
            calculationSheet.TotalHoursGia =
Convert.ToDouble(table.Rows[row][column + 9]);
        }

        private List<CalculationSheetDiscipline>
LoadDisciplineAcademicPlan(DataTable table)
        {
            List<CalculationSheetDiscipline>
calculationSheetDisciplines = new
List<CalculationSheetDiscipline>();

            for (int i =
ImportRowColumnConstants.CalculationSheetDisciplineStartRow; i <
table.Rows.Count; i++)
            {
                CalculationSheetDiscipline
calculationSheetDiscipline = new CalculationSheetDiscipline();
                int column =
ImportRowColumnConstants.CalculationSheetDisciplineStartColumn;

                calculationSheetDiscipline.Index =
table.Rows[i][column].ToString();
                calculationSheetDiscipline.Name =
table.Rows[i][column + 1].ToString();

                calculationSheetDiscipline.Semester =
Convert.ToInt32(table.Rows[i][column + 2]);
                calculationSheetDiscipline.LengthSemester =
table.Rows[i][column + 3].ToString();

                calculationSheetDiscipline.CountGroups =
Convert.ToInt32(table.Rows[i][column + 4]);
                calculationSheetDiscipline.Groups =
table.Rows[i][column + 5].ToString();

                calculationSheetDiscipline.Stream =
table.Rows[i][column + 6].ToString();

calculationSheetDiscipline.StreamPracticumLabrotory =
table.Rows[i][column + 7].ToString();
```

```
                calculationSheetDiscipline.StudentsBudget =
Convert.ToInt32(table.Rows[i][column + 8]);
                calculationSheetDiscipline.StudentsContract =
Convert.ToInt32(table.Rows[i][column + 9]);

                calculationSheetDiscipline.CountLecture =
ToInt32(table.Rows[i][column + 10]);
                calculationSheetDiscipline.CountLaboratoryWork =
ToInt32(table.Rows[i][column + 11]);
                calculationSheetDiscipline.CountPracticum =
ToInt32(table.Rows[i][column + 12]);
                calculationSheetDiscipline.CountKp =
ToInt32(table.Rows[i][column + 13]);
                calculationSheetDiscipline.CountKr =
ToInt32(table.Rows[i][column + 14]);
                calculationSheetDiscipline.CountControlWork =
ToInt32(table.Rows[i][column + 15]);
                calculationSheetDiscipline.CountExam =
ToInt32(table.Rows[i][column + 16]);
                calculationSheetDiscipline.CountTest =
ToInt32(table.Rows[i][column + 17]);

calculationSheetDiscipline.CountDifferentiatedTest =
ToInt32(table.Rows[i][column + 18]);
                calculationSheetDiscipline.CountEtc =
ToInt32(table.Rows[i][column + 19]);

                calculationSheetDiscipline.HoursLecture =
Convert.ToDouble(table.Rows[i][column + 20]);
                calculationSheetDiscipline.HoursLaboratoryWork =
Convert.ToDouble(table.Rows[i][column + 21]);
                calculationSheetDiscipline.HoursPracticumLesson
= Convert.ToDouble(table.Rows[i][column + 22]);
                calculationSheetDiscipline.HoursKpKr =
Convert.ToDouble(table.Rows[i][column + 23]);
                calculationSheetDiscipline.HoursControlWork =
Convert.ToDouble(table.Rows[i][column + 24]);
                calculationSheetDiscipline.HoursExam =
Convert.ToDouble(table.Rows[i][column + 25]);
                calculationSheetDiscipline.HoursTest =
Convert.ToDouble(table.Rows[i][column + 26]);
                calculationSheetDiscipline.HoursConsultation =
Convert.ToDouble(table.Rows[i][column + 27]);
                calculationSheetDiscipline.HoursPracticum =
Convert.ToDouble(table.Rows[i][column + 28]);
                calculationSheetDiscipline.HoursGia =
Convert.ToDouble(table.Rows[i][column + 29]);
                calculationSheetDiscipline.HoursTotal =
Convert.ToDouble(table.Rows[i][column + 30]);
```

```
calculationSheetDisciplines.Add(calculationSheetDiscipline);
            }

            return calculationSheetDisciplines;
        }

        private int ToInt32(object item) => item is DBNull ? 0 :
Convert.ToInt32(item);
    }
}
```

Файл «*CalculationSheetView.xaml*»

```
<UserControl
x:Class="AcademicLoadModule.Views.CalculationSheetView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="1080" d:DesignWidth="1080"
             xmlns:prism="http://prismlibrary.com/"
             xmlns:prop="clr-
namespace:AcademicLoadModule.Properties"
             xmlns:models="clr-
namespace:Data.Models;assembly=Data"
             xmlns:viewModels="clr-
namespace:AcademicLoadModule.ViewModels"
             prism:ViewModelLocator.AutoWireViewModel="True">


    <UserControl.Resources>
        <ResourceDictionary
Source="/Infrastructure;component/Themes/Generic.xaml"/>
    </UserControl.Resources>


    <Grid DataContext="{Binding CalculationSheet}">
        <Grid.Resources>
            <Style TargetType="TextBlock"
BasedOn="{StaticResource {x:Type TextBlock}}">
                <Setter Property="Margin"
Value="0,0,0,8"></Setter>
                <Setter Property="Foreground"
Value="{StaticResource White}"></Setter>
            </Style>
        </Grid.Resources>
```

```
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <StackPanel Grid.Row="0"
                    Margin="10,5"
                    Orientation="Horizontal">
            <TextBlock Style="{StaticResource HeaderH1}"
                       VerticalAlignment="Center"
                       Text="{Binding FileName}" />

            <Button Style="{StaticResource ButtonWithIcon}"
                    Margin="20,0,10,0"
                    Command="{Binding
DataContext.DeleteCalculationSheetCommand,
RelativeSource={RelativeSource AncestorType={x:Type
UserControl}}}"
                    ToolTip="{x:Static
prop:Resources.DeleteCalculationSheet}"
                    VerticalAlignment="Center">
                <Path Style="{StaticResource
DeleteCaclulationSheetsIcon}" />
            </Button>

            <Button Style="{StaticResource ButtonWithIcon}"
                    Command="{Binding
DataContext.ExportTeacherLoadCommand,
RelativeSource={RelativeSource AncestorType={x:Type
UserControl}}}"
                    ToolTip="{x:Static
prop:Resources.ExportTeacherLoad}"
                    VerticalAlignment="Center">
                <Path Style="{StaticResource
ExportTeacherLoadIcon}" />
            </Button>
        </StackPanel>

        <Border Grid.Row="1" Background="{StaticResource
Tint95}" Padding="16">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
```

```
                <StackPanel Orientation="Horizontal"
                            Margin="0,0,20,0"
                            Grid.Row="1" Grid.Column="0">
                    <StackPanel Orientation="Vertical">
                        <TextBlock Text="Факультет:" />
                        <TextBlock Text="Кафедра:" />
                        <TextBlock Text="Учебный год:" />
                    </StackPanel>
                    <StackPanel Orientation="Vertical"
                            Margin="10,0">
                        <TextBlock Text="{ Binding
Path=Department}" />
                        <TextBlock Text="{ Binding
Path=Faculty}" />
                        <TextBlock Text="{ Binding
Path=AcademicYear}" />
                    </StackPanel>

                    <StackPanel Orientation="Vertical"
                                Margin="15,0,0,0">
                        <TextBlock Text="Всего бюд.:" />
                        <TextBlock Text="Всего бюд.:" />
                        <TextBlock Text="Всего часов:" />
                    </StackPanel>
                    <StackPanel Orientation="Vertical"
                            Margin="10,0">
                        <TextBlock  Text="{ Binding
Path=TotalStudentsBudget}"
                                    Foreground="{StaticResource
YellowText}" />
                        <TextBlock Text="{ Binding
Path=TotalStudentsContract}"
                                   Foreground="{StaticResource
YellowText}"/>
                        <TextBlock Text="{ Binding
Path=TotalHours}"
                                   Foreground="{StaticResource
YellowText}"/>
                    </StackPanel>
                </StackPanel >

                <TextBlock Grid.Row="0" Grid.Column="1"
                           Text="Итого по расчету часов"
                           Margin="0,0,0,10"
                           Foreground="{StaticResource
YellowText}"
                           Style="{StaticResource HeaderH2}" />

                <StackPanel Orientation="Horizontal"
                            Grid.Row="1" Grid.Column="1">
                    <StackPanel Orientation="Vertical">
```

```
                        <TextBlock Text="Лекции:" />
                        <TextBlock Text="Лабораторные:" />
                        <TextBlock Text="Практические занятия:"
/>
                    </StackPanel>
                    <StackPanel Orientation="Vertical"
                            Margin="10,0">
                        <TextBlock Text="{ Binding
Path=TotalHoursLecture}"
                                   Foreground="{StaticResource
YellowText}" />
                        <TextBlock Text="{ Binding
Path=TotalHoursLaboratory}"
                                   Foreground="{StaticResource
YellowText}" />
                        <TextBlock Text="{ Binding
Path=TotalHoursPracticalWorks}"
                                   Foreground="{StaticResource
YellowText}"/>
                    </StackPanel>

                    <StackPanel Orientation="Vertical"
                                Margin="15,0,0,0">
                        <TextBlock Text="КП, КР:" />
                        <TextBlock Text="к/р:" />
                        <TextBlock Text="Экзамены:" />
                    </StackPanel>
                    <StackPanel Orientation="Vertical"
                                Margin="10,0">
                        <TextBlock Text="{ Binding
Path=TotalHoursKpKr}"
                                   Foreground="{StaticResource
YellowText}"/>
                        <TextBlock Text="{ Binding
Path=TotalHoursControlWorks}"
                                   Foreground="{StaticResource
YellowText}"/>
                        <TextBlock Text="{ Binding
Path=TotalHoursExam}"
                                   Foreground="{StaticResource
YellowText}"/>
                    </StackPanel>

                    <StackPanel Orientation="Vertical"
                                Margin="15,0,0,0">
                        <TextBlock Text="Зач,зач. с оцн.:" />
                        <TextBlock Text="Консультации:" />
                        <TextBlock Text="Практика, рук-во:" />
                    </StackPanel>
                    <StackPanel Orientation="Vertical"
                            Margin="10,0">
```

```
                        <TextBlock Text="{ Binding
Path=TotalHoursTest}"
                                   Foreground="{StaticResource
YellowText}" />
                        <TextBlock Text="{ Binding
Path=TotalHoursConsultation}"
                                   Foreground="{StaticResource
YellowText}" />
                        <TextBlock Text="{ Binding
Path=TotalHoursPracticum}"
                                   Foreground="{StaticResource
YellowText}" />
                    </StackPanel>

                    <StackPanel Orientation="Vertical"
                                Margin="15,0,0,0">
                        <TextBlock Text="ГИА:" />
                    </StackPanel>
                    <StackPanel Orientation="Vertical"
                            Margin="10,0">
                        <TextBlock Text="{ Binding
Path=TotalHoursGia}"
                                       Foreground="{StaticResource
YellowText}" />
                    </StackPanel>

                </StackPanel>
            </Grid>
        </Border>

        <DataGrid Grid.Row="2"
                  IsReadOnly="True"
                  ScrollViewer.CanContentScroll="True"
                  SelectedItem="{Binding
Path=DataContext.SelectedCalculationSheetDiscipline,
RelativeSource={RelativeSource AncestorType={x:Type
UserControl}}}"
                  ItemsSource="{Binding
CalculationSheetDisciplines}"
                  Background="{StaticResource White}"
                  Style="{StaticResource DataGridBaseStyle}"
                  ColumnHeaderStyle="{StaticResource
DataGridColumnHeaderBaseStyle}"
                  RowStyle="{StaticResource
DataGridRowBaseStyle}"
                  CellStyle="{StaticResource
DataGridCellBaseStyle}">
            <DataGrid.RowDetailsTemplate>
                <DataTemplate>
                    <Border Padding="20,2,2,2"
                            Margin="0,10"
```

```
                                Background="White"
                                BorderBrush="{StaticResource
BrandPrimary}"
                                BorderThickness="1"
                                CornerRadius="4">
                            <Grid>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="Auto" />
                                    <RowDefinition MaxHeight="300"
/>
                                </Grid.RowDefinitions>
                                <StackPanel Grid.Row="0"
                                            Orientation="Horizontal"
                                            Margin="2,2"

HorizontalAlignment="Left">
                                    <TextBlock Text="Список
преподователей"

Style="{StaticResource HeaderH2}"
                                               Margin="0,0,10,0"

VerticalAlignment="Center"/>
                                    <Button Style="{StaticResource
ButtonWithIcon}"
                                            Command="{Binding
Path=DataContext.AddTeacherLoadDisciplineCommand,
RelativeSource={RelativeSource AncestorType={x:Type
UserControl}}}"

CommandParameter="{Binding DataContext,
RelativeSource={RelativeSource AncestorType={x:Type
DataGridRow}}}"
                                            ToolTip="Назначить
нового преподавателя к данной дисциплине"

VerticalAlignment="Center">
                                        <Path Style="{StaticResource
AddTeacherIcon}"></Path>
                                    </Button>
                                </StackPanel>
                                <DataGrid Grid.Row="1"
                                          Visibility="{Binding
HasTeacherLoadDisciplines,Converter={StaticResource
BooleanToVisibilityConverter}}"

PreparingCellForEdit="MainDataGrid_PreparingCellForEdit"

ScrollViewer.CanContentScroll="True"

CanUserResizeColumns="True"
```

```
                                        AlternationCount="1"
RowDetailsVisibilityMode="Collapsed"
                                        ItemsSource="{Binding
TeacherLoadDisciplines}"
Background="{StaticResource White}"
                                        Style="{StaticResource
DataGridBaseStyle}"
ColumnHeaderStyle="{StaticResource
DataGridColumnHeaderBaseStyle}"
                                        RowStyle="{StaticResource
DataGridRowStyleWithoutSelected}"
                                        CellStyle="{StaticResource
DataGridCellBaseStyle}">
                                    <DataGrid.Resources>
                                        <ControlTemplate
x:Key="validationFailed">
                                            <StackPanel
Orientation="Horizontal">
                                                <Border
BorderBrush="Red" BorderThickness="2">
<AdornedElementPlaceholder />
                                                </Border>
                                                <TextBlock
Foreground="Red" FontSize="18" FontWeight="Bold">!</TextBlock>
                                            </StackPanel>
                                        </ControlTemplate>

                                        <Style x:Key="EditCellStyle"
TargetType="{x:Type TextBox}">
                                            <Setter
Property="FontSize" Value="24" />
                                            <Setter
Property="BorderThickness" Value="1" />
                                            <Setter
Property="BorderBrush" Value="{StaticResource BlueHeader}" />
                                        </Style>


                                        <Style
x:Key="EditCellStyleError" TargetType="{x:Type TextBox}">
                                            <Setter
Property="Validation.ErrorTemplate" Value="{StaticResource
validationFailed}"></Setter>
                                            <Style.Triggers>
                                                <Trigger
Property="Validation.HasError" Value="true">
```

```
                                                    <Setter
Property="ToolTip" Value="{Binding
RelativeSource={RelativeSource Self},
Path=(Validation.Errors)[0].ErrorContent}"/>
                                                    <Setter
Property="Background" Value="Red"/>
                                                </Trigger>
                                            </Style.Triggers>
                                        </Style>

                                        <Style
x:Key="CellStyleError" TargetType="{x:Type TextBlock}">
                                            <Setter
Property="TextBlock.TextWrapping" Value="Wrap" />
                                            <Setter
Property="Validation.ErrorTemplate" Value="{StaticResource
validationFailed}"></Setter>
                                            <Style.Triggers>
                                                <Trigger
Property="Validation.HasError" Value="true">
                                                    <Setter
Property="ToolTip" Value="{Binding
RelativeSource={RelativeSource Self},
Path=(Validation.Errors)[0].ErrorContent}"/>
                                                </Trigger>
                                            </Style.Triggers>
                                        </Style>


                                    </DataGrid.Resources>
                                    <DataGrid.Columns>
                                        <DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>
                                                <DataTemplate>
                                                    <Button
Style="{StaticResource ButtonWithIcon}"

ToolTip="Удалить назначение для данного преподавателя"

Command="{Binding
Path=DataContext.DeleteTeacherLoadDisciplineCommand,
RelativeSource={RelativeSource AncestorType={x:Type
UserControl}}}"

CommandParameter="{Binding .}"

VerticalAlignment="Center"

Margin="0,0,8,0">
```

```
                                                    <Path
Style="{StaticResource DeleteIcon}" />
                                                </Button>
                                            </DataTemplate>

</DataGridTemplateColumn.CellTemplate>
                                    </DataGridTemplateColumn>

                                    <DataGridTextColumn
Header="Преподаватель"

MaxWidth="150"

IsReadOnly="True"

Binding="{Binding Teacher,Converter={StaticResource
TeacherInitialsConverter}}">

<DataGridTextColumn.ElementStyle>
                                            <Style>
                                                <Setter
Property="TextBlock.TextWrapping" Value="Wrap" />
                                            </Style>

</DataGridTextColumn.ElementStyle>
                                    </DataGridTextColumn>

                                    <DataGridTextColumn
Header="Семестр"

IsReadOnly="True"

MaxWidth="150"

Binding="{Binding Semester}"/>

                                    <DataGridTextColumn
Header="Группы"

IsReadOnly="True"

MaxWidth="70"

ElementStyle="{StaticResource CellStyleError}"

Binding="{Binding Groups,Converter={StaticResource
GroupsNamesToInfoConverter}}"/>
                                    <DataGridTextColumn
Header="Студ.(бюд.)"

IsReadOnly="True"
```

```
Binding="{Binding StudentsBudget}"/>
                                        <DataGridTextColumn
Header="Студ.(дог.)"

IsReadOnly="True"

Binding="{Binding StudentsContract}"/>

                                        <DataGridTextColumn
Header="Лекции"

Binding="{Binding HoursLecture,
ValidatesOnDataErrors=True,UpdateSourceTrigger=LostFocus}"

Validation.ErrorTemplate="{StaticResource validationFailed}"

EditingElementStyle="{StaticResource EditCellStyle}" />

                                        <DataGridTextColumn
Header="Лабор"

Binding="{Binding HoursLaboratoryWork,
UpdateSourceTrigger=LostFocus}"

EditingElementStyle="{StaticResource EditCellStyle}"
                                                            />
                                        <DataGridTextColumn
Header="Практич."

Binding="{Binding HoursPracticum ,
UpdateSourceTrigger=LostFocus}"

EditingElementStyle="{StaticResource EditCellStyle}"/>
                                        <DataGridTextColumn
Header="КП/КР"

Binding="{Binding HoursKpKr, UpdateSourceTrigger=LostFocus}"

EditingElementStyle="{StaticResource EditCellStyle}"/>
                                        <DataGridTextColumn
Header="контр./раб. (ЗФ)"

Binding="{Binding HoursControlWork,
UpdateSourceTrigger=LostFocus}"

EditingElementStyle="{StaticResource EditCellStyle}"/>
                                        <DataGridTextColumn
Header="Экзамены"

Binding="{Binding HoursExam, UpdateSourceTrigger=LostFocus}"
```

```
EditingElementStyle="{StaticResource EditCellStyle}"/>
                                    <DataGridTextColumn
Header="Зачеты"

Binding="{Binding HoursTest, UpdateSourceTrigger=LostFocus}"

EditingElementStyle="{StaticResource EditCellStyle}"/>
                                    <DataGridTextColumn
Header="Консультации"

Binding="{Binding HoursConsultation,
UpdateSourceTrigger=LostFocus}"

EditingElementStyle="{StaticResource EditCellStyle}"/>
                                    <DataGridTextColumn
Header="Прочая нагрузка по ВПО"

Binding="{Binding HoursOtherLoadVpo,
UpdateSourceTrigger=LostFocus}"

EditingElementStyle="{StaticResource EditCellStyle}"/>
                                    <DataGridTextColumn
Header="Подготовка аспирантов, интернов, ординаторов"

MaxWidth="200"

Binding="{Binding HoursTraining, UpdateSourceTrigger=LostFocus}"

EditingElementStyle="{StaticResource EditCellStyle}">

<DataGridTextColumn.ElementStyle>
                                        <Style>
                                            <Setter
Property="TextBlock.TextWrapping" Value="Wrap" />
                                        </Style>

</DataGridTextColumn.ElementStyle>
                                    </DataGridTextColumn>

                                    <DataGridTextColumn
Header="Всего осенний семестр"

IsReadOnly="True"

Binding="{Binding HoursTotalFallSemester}"/>
                                    <DataGridTextColumn
Header="Всего весенний семестр"

IsReadOnly="True"
```

```
Binding="{Binding HoursTotalSpringSemester}"/>
                                    <DataGridTextColumn
Header="Всего за год нагрузка"

IsReadOnly="True"

Binding="{Binding HoursTotalYearLoad}"/>
                                </DataGrid.Columns>

                            </DataGrid>
                        </Grid>

                    </Border>
                </DataTemplate>
            </DataGrid.RowDetailsTemplate>

            <DataGrid.Columns>
                <DataGridTextColumn Header="индекс"
                                    Binding="{Binding Index}"/>
                <DataGridTextColumn Header="ДИСЦИПЛИНЫ,
практика, дипломное проектирование и др."
                                    MaxWidth="220"
                                    Binding="{Binding Name}">
                    <DataGridTextColumn.ElementStyle>
                        <Style>
                            <Setter
Property="TextBlock.TextWrapping" Value="Wrap" />
                        </Style>
                    </DataGridTextColumn.ElementStyle>
                </DataGridTextColumn>
                <DataGridTextColumn Header="Семестр"
                                    MaxWidth="150"
                                    Binding="{Binding
Semester}"/>
                <DataGridTextColumn Header="Дли-тельн. (сем.)"
                                    MaxWidth="150"
                                    Binding="{Binding
LengthSemester}"/>

                <DataGridTextColumn Header="Кол. групп"
                                    MaxWidth="150"
                                    Binding="{Binding
CountGroups}"/>

                <DataGridTextColumn Header="Группы"
                                    MaxWidth="150"
                                    Binding="{Binding Groups}"/>

                <DataGridTextColumn Header="Поток"
                                    MaxWidth="70"
```

```
                                    Binding="{Binding Stream}"/>
                <DataGridTextColumn Header="Поток практ+лаб"
                                    MaxWidth="150"
                                    Binding="{Binding
StreamPracticumLabrotory}"/>

                <DataGridTextColumn Header="Студ. (бюд.)"
                                    Binding="{Binding
StudentsBudget}"/>
                <DataGridTextColumn Header="Студ. (дог.)"
                                    Binding="{Binding
StudentsContract}"/>

                <DataGridTextColumn Header="Лекции"
                                    Binding="{Binding
CountLecture}"/>
                <DataGridTextColumn Header="Лабор"
                                    Binding="{Binding
CountLaboratoryWork}"/>
                <DataGridTextColumn Header="КП"
                                    Binding="{Binding
CountKp}"/>
                <DataGridTextColumn Header="КР"
                                    Binding="{Binding
CountKr}"/>
                <DataGridTextColumn Header="к/р"
                                    Binding="{Binding
CountControlWork}"/>
                <DataGridTextColumn Header="Экз"
                                    Binding="{Binding
CountExam}"/>
                <DataGridTextColumn Header="Зач"
                                    Binding="{Binding
CountTest}"/>
                <DataGridTextColumn Header="Зач с оц"
                                    Binding="{Binding
CountDifferentiatedTest}"/>

                <DataGridTextColumn Header="Про-чее (нед)"

                                    Binding="{Binding
CountEtc}"/>
                <DataGridTextColumn Header="Лекции"
                                    Binding="{Binding
HoursLecture}"/>
                <DataGridTextColumn Header="Лабор"
                                    Binding="{Binding
HoursLaboratoryWork}"/>
                <DataGridTextColumn Header="Практ занятия"
                                    Binding="{Binding
HoursPracticumLesson}"/>
```

```
                <DataGridTextColumn Header="КП, КР"
                                    Binding="{Binding
HoursKpKr}"/>
                <DataGridTextColumn Header="Экз"
                                    Binding="{Binding
HoursExam}"/>
                <DataGridTextColumn Header="Зач, зач с оц"
                                    Binding="{Binding
HoursTest}"/>
                <DataGridTextColumn Header="Конс"
                                    Binding="{Binding
HoursConsultation}"/>
                <DataGridTextColumn Header="Практика, рук-во"
                                    Binding="{Binding
HoursPracticum}"/>
                <DataGridTextColumn Header="ГИА"
                                    Binding="{Binding
HoursGia}"/>
                <DataGridTextColumn Header="Всего"
                                    Binding="{Binding
HoursTotal}"/>
            </DataGrid.Columns>
        </DataGrid>

    </Grid>
</UserControl>
```

Приложение В

(Обязательное)

Автоматизированная система планирования учебной нагрузки преподавателей

Презентация

Тема работы:
«Автоматизированная система планирования учебной нагрузки преподавателей»

Автор работы: студент группы 18ВВ2 Милованов А.С.

Научный руководитель: к.т.н., доцент каф. ВТ ПГУ Карамышева Н.С.

Пенза 2022

Цель и объект работы

Цель работы

разработка системы, позволяющей автоматизировать процесс планирования учебной нагрузки преподавателей.

Объект работы

Приложение, выполняющее следующие функции:

- импорт информации из расчетного листа кафедральной нагрузки;
- распределение нагрузки преподавателей с учетом установленных норм времени;
- экспорт преподавательской нагрузки в файл формата xlsx;
- наличие удобного графического интерфейса.

Постановка задачи

В процессе разработки должны быть решены следующие задачи:

- импорт данных в приложение из расчетных листов кафедральной нагрузки, которые хранятся в файлах в формате xlsx;
- отображение импортированной информации из расчетного листа кафедральной нагрузки;
- экспорт учебной нагрузки в отдельный файл в формате xlsx;
- автоматическая расстановка основных расчетных формул в советующие ячейки таблицы;
- добавление, изменение, хранение основных данных о преподавателях;
- добавление, изменение, хранение основных данных об учебных группах;
- возможность ручной корректировки значений расчетов пользователем.

Расчетный лист кафедральной нагрузки

Кафедра	Факультет	Уч. год	Всего бюд.	Всего дог.	Всего часов
ВТ	ФВТ	2021-2022	13812,0	1387,2	15199

к 1-му приказу

индекс	ДИСЦИПЛИНЫ, практика, дипломное проектирование и др.	Семестр	Длительн. (сем.)	Кол. групп	Группы	Поток	Поток практ+лаб	Студ. (бюд.)	Студ. (дог.)	по дисциплинам: Лекции	Лабор	Практ	КП	КР	к/р	Экз	Зач	Зач с оц	Прочее (нед)	итого по расчету (час.): Лекции	Лабор	Практ занятия	КП, КР	к/р	Экз	Зач, зач с оц	Конс	Практика, рук-во	ГИА	Всего
																				2218,0	8091,0	0,0	1449,5	0,0	549,9	401,5	455,3	561,0	1473,0	15199,2
Б1.О.1.18	Программирование	1	12	4	21ВВ1-4			95	0	34	34					1				34,0	238,0	0,0	0,0	0,00	28,5	0,00	17,6	0,0	0,0	318,1
Б1.О.11	Современные информационные технологии	1	1	1	21ВА1	ВА, ВВ		25	0	17	17						1			17,0	34,0	0,0	0,0	0,00	0,0	6,25	1,7	0,0	0,0	59,0
Б1.О.1.11	Современные информационные технологии	1	1	4	21ВВ1-4	ВА, ВВ		95	0	17	17						1			0,0	119,0	0,0	0,0	0,00	0,0	23,75	6,8	0,0	0,0	149,6
Б1.О.1.24	Арифметические и логические основы вычислительной техники	2	2	4	21ВВ1-4			95	0	17	34					1				17,0	238,0	0,0	0,0	0,00	28,5	0,00	10,8	0,0	0,0	294,3
Б1.О.21	Информационные технологии в профессиональной деятельности	2	2	1	21ВА1			25	0		34						1			0,0	68,0	0,0	0,0	0,00	0,0	6,25	0,0	0,0	0,0	74,3
Б1.О.1.15	Информационные технологии в профессиональной деятельности	2	2	4	21ВВ1-4			95	0		34						1			0,0	238,0	0,0	0,0	0,00	0,0	23,75	0,0	0,0	0,0	261,8
Б1.О.1.18	Программирование	2	12	4	21ВВ1-4			95	0	34	34			1		1				34,0	238,0	0,0	142,5	0,00	28,5	0,00	17,6	0,0	0,0	460,6
Б2.О.01 у	Учебная (ознакомительная) практика	2	2	3	21ВВ1-3			70	0										2	0,0	0,0	0,0	0,0	0,00	0,0	0,00	0,0	72,0	0,0	72,0
	Учебная: Практика по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков	2	2	1	21ВО2			20	0										4	0,0	0,0	0,0	0,0	0,00	0,0	0,00	0,0	48,0	0,0	48,0

Итоговой лист

	A	B	C	D	E	F	G	H	I	J	K
4											
5											
6	Кафедра	ВТ									
7											
8	Учебный год	2021/22		850							
9											
10		ставки по приказу	1	3,75	8,25	2,00	0,00	2,00	15,25	12707,50	
11		ставки по факту	1,00	3,75	8,25	1,00		2,50	16,00	12651,40	2057,75
12		Ф.И.О.	зав.каф	проф.	доц.	ст. преп.	преп.	ассистент	ставки	учебная нагрузка	лекционная нагрузка
13	1	Бикташев Р.А.			0,75				0,75	640,90	142,00
14	2	Бычков А.С.			1,00				1,00	860,30	158,00
15	3	Гурин Е.И.		1,00					1,00	850,05	191,00
16	4	Дорошенко И.Н.			0,25				0,25	216,05	68,00
17	5	Дубинин В.Н.		1,00					1,00	770,60	193,00
18	6	Дубравин А.В.			0,25				0,25	221,30	68,00
19	7	Егоров В.Ю.			0,25				0,25	223,85	68,00
20	8	Зинкин С.А.		1,00					1,00	847,40	198,00
21	9	Калиниченко Е.И.				0,50			0,50	406,05	89,00
22	10	Карамышева Н.С.			1,00				1,00	826,35	70,00

Отдельный лист для преподавателя

Пензенский Государственный Университет

УТВЕРЖДАЮ

Распределение учебной нагрузки

Проректор по учебной работе

Кафедра **ВТ**

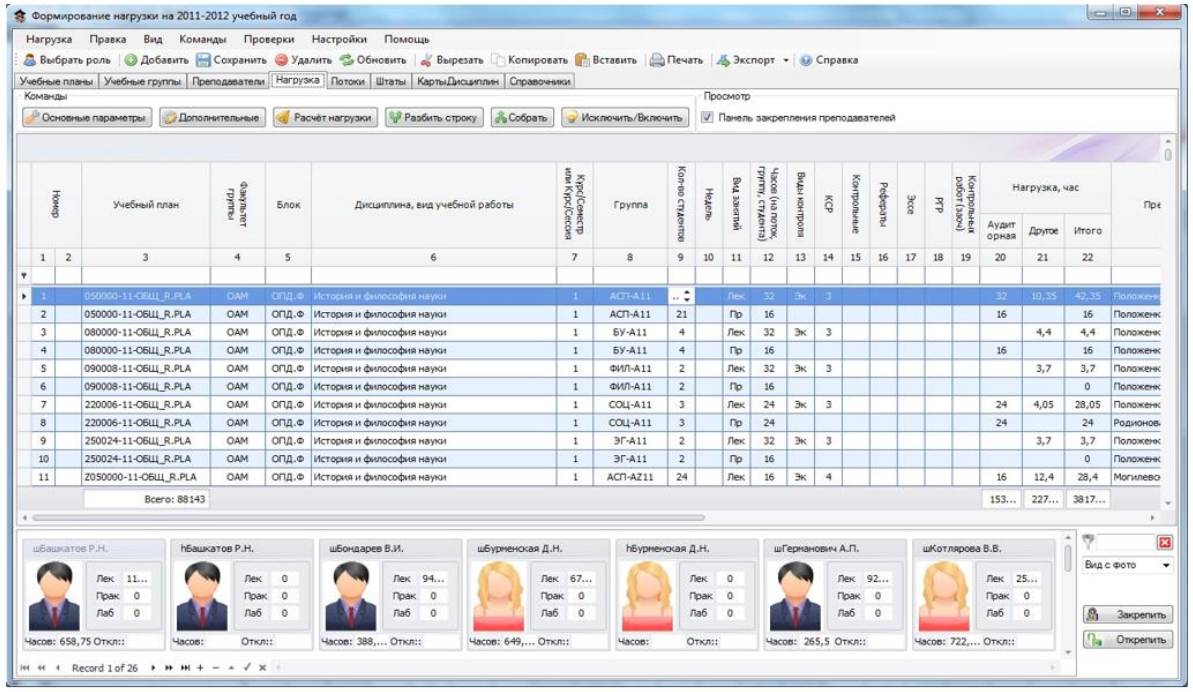
Учебный год **2021/22** ср. год. нагрузка **850**

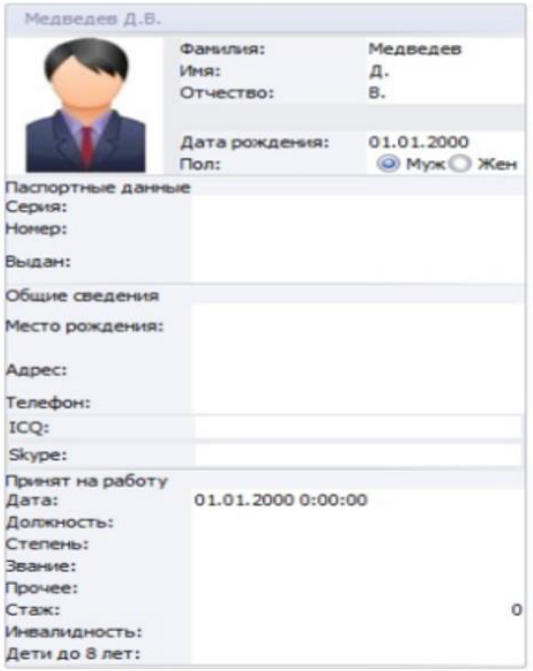
Ф.И.О. **Карамышева Н.С.** должность **доцент** ставка **1,00**

	Студенты (бюд)	Студенты (дог)	Лекции	Лабораторные	Практические	КП / КР	контр./раб. (ЗФ)	Экзамены	Зачеты	консультации	Прочая нагрузка по ВПО	Подготовка аспирантов, интернов, ординаторов	Всего **осенний** семестр	Всего **весенний** семестр	Всего за год нагрузка
итого :	369,0	51,0	70,0	393,0	0,0	39,0	0,0	5,4	35,25	19,7	264,0	0,0	212,3	614,1	826,35

Дисциплины	Семестр	Группы	Студенты (бюд)	Студенты (дог)	Лекции	Лабораторные	Практические	КП / КР	контр./раб. (ЗФ)	Экзамены	Зачеты	консультации	Прочая нагрузка по ВПО	Подготовка аспирантов, интернов, ординаторов	Всего **осенний** семестр	Всего **весенний** семестр	Всего за год нагрузка
Программно-аппаратные средства хранения и обработки данных	6	19ВВ2	25	2		34,0					3,75	2,2			0	40	40,0
Защита информации в сети Internet	5	19ВВ1	22	5		34,0									34	0	34,0
Программирование для сети	5	19ВО2	17	1	36	36				5,4		4,6			82	0	82,0

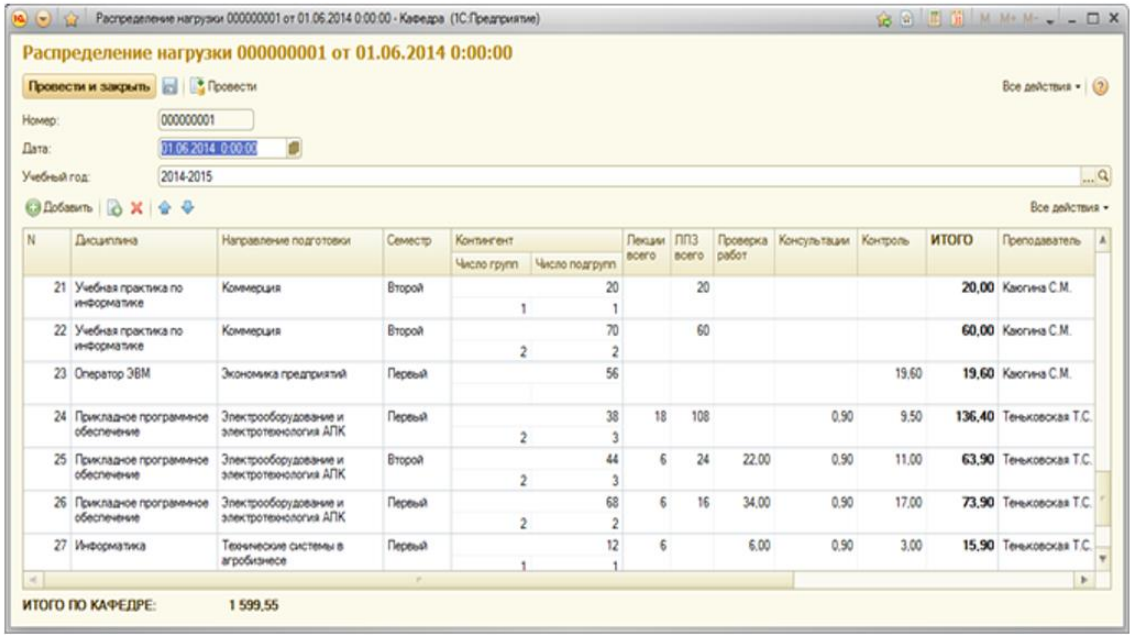
Аналоги

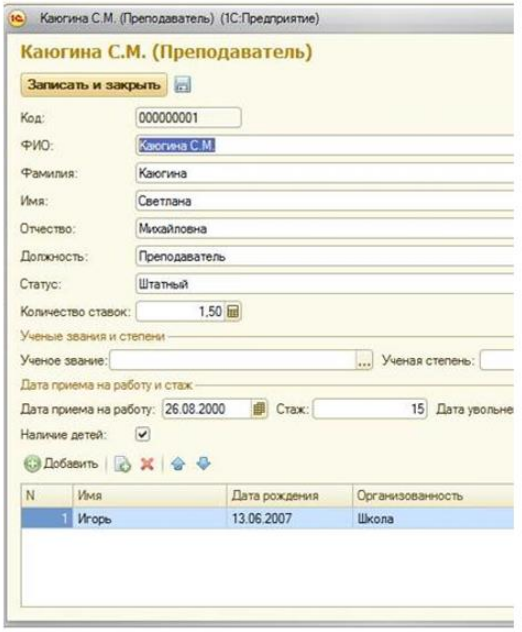




Автоматизированная система «НАГРУЗКА ВУЗА»

Аналоги





АВТОМАТИЗИРОВАННАЯ СИСТЕМА РАСЧЕТА И РАСПРЕДЕЛЕНИЯ УЧЕБНОЙ НАГРУЗКИ ПРЕПОДАВАТЕЛЕЙ КАФЕДРЫ ВУЗА НА БАЗЕ ПЛАТФОРМЫ «1С»

Аналоги

Фактическая нагрузка

Кафедра палеонтологии

2012/2013 учебный год

Орлова Ольга Александровна

Добавить новую запись На учебный год --------- скопировать

Сформировать отчет по сотруднику(Отчет содержит только подтвержденные данные)

Сформировать сводный отчет по целевым факультетам(Отчет формируется только по обработанным правкам и содержит только подтвержденные правки)

Показывать по 10 записей на странице

Поиск:

Действия	Доля участия	Вид работы	Дисциплина(Наименование курса)	Целевой факультет	Вид подготовки	Форма обучения	Семестр	Кол-во недель	Кол-во часов в неделю	Общее кол-во часов	Кол-во бюджетных студентов	Кол-во платных студентов	Кол-во листов	Кол-во работ	Категория
Копировать Внести правку	100	рецензирование и консультирование рефератов		Геологический факультет	специалист		4			3	3	0			Запись из ПН
Копировать Изменить Удалить	100	лекции	История развития жизни в естественнонаучных музеях	Геологический факультет	аспирантура	очная	10	12	3	36					Нет в ПН

Показаны записи с 1 по 2. Всего: 2.

Предыдущая 1 Следующая

Подсистема «Педагогическая нагрузка»

Стек технологий

WPF

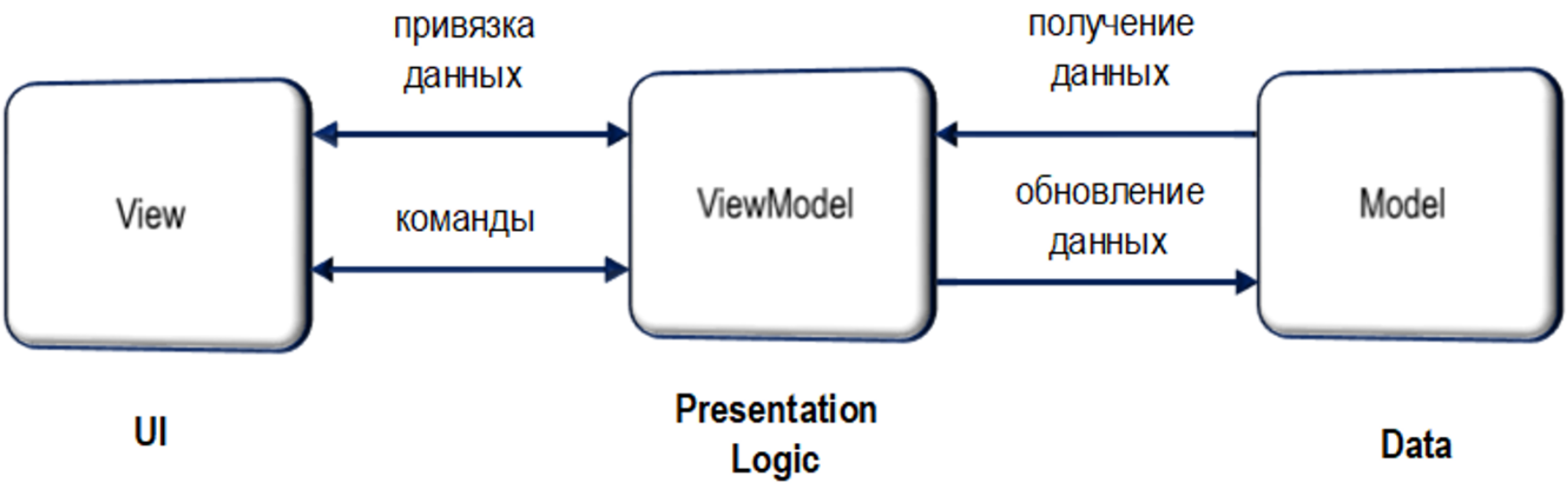
Microsoft® .NET

Windows Presentation Foundation

PRISM LIBRARY

EPPlus

Паттерн Model—View—ModelView



Архитектура приложения

При разработке приложения использовался фреймворк PRISM, архитектура, с учетом возможностей данного фреймворка, состоит из следующих частей:

- PRESENTATION LAYER – слой представлений
- MODULES LAYER – слой модулей
- CORE LAYER – основной слой
- DATA LAYER – слой данных

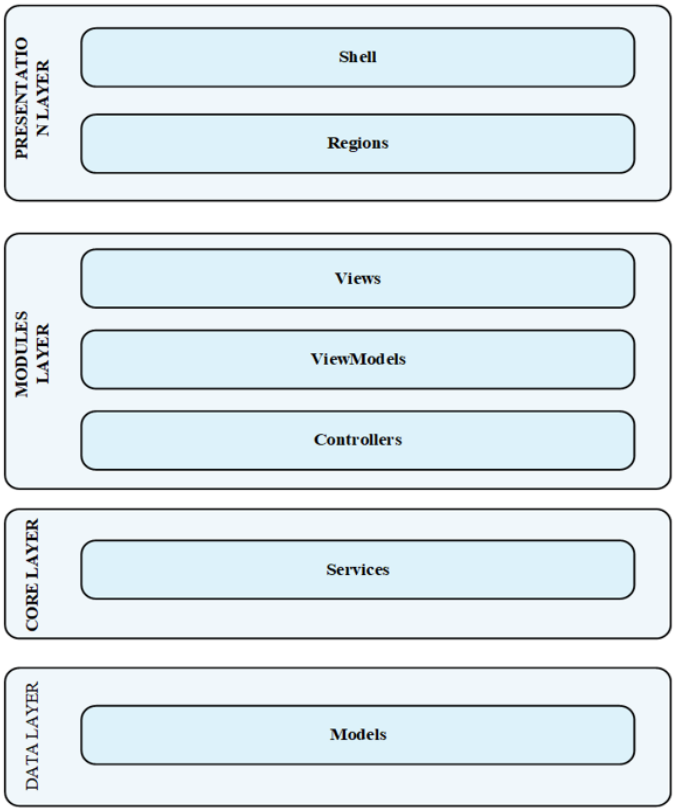


Диаграмма вариантов использования

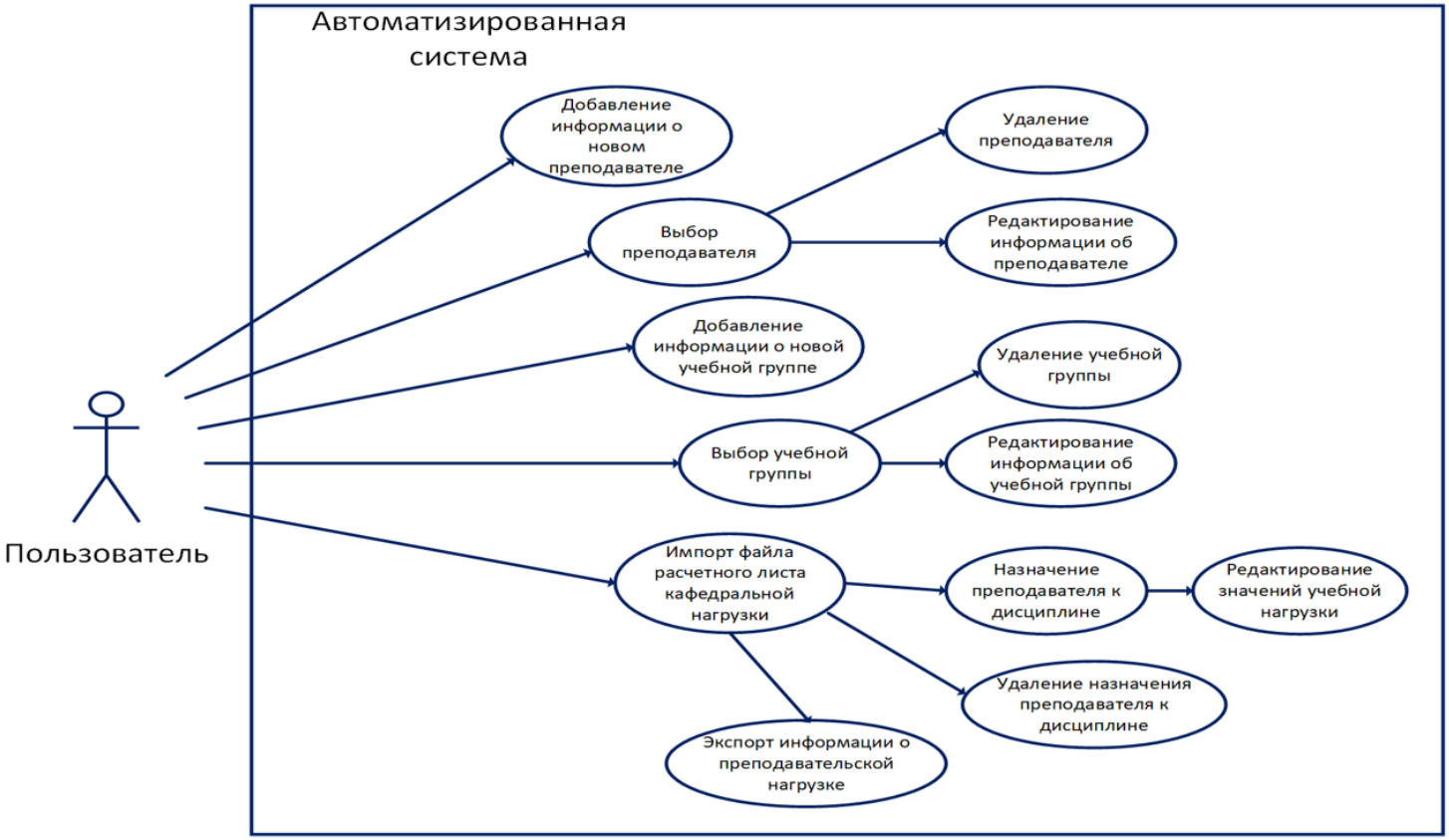
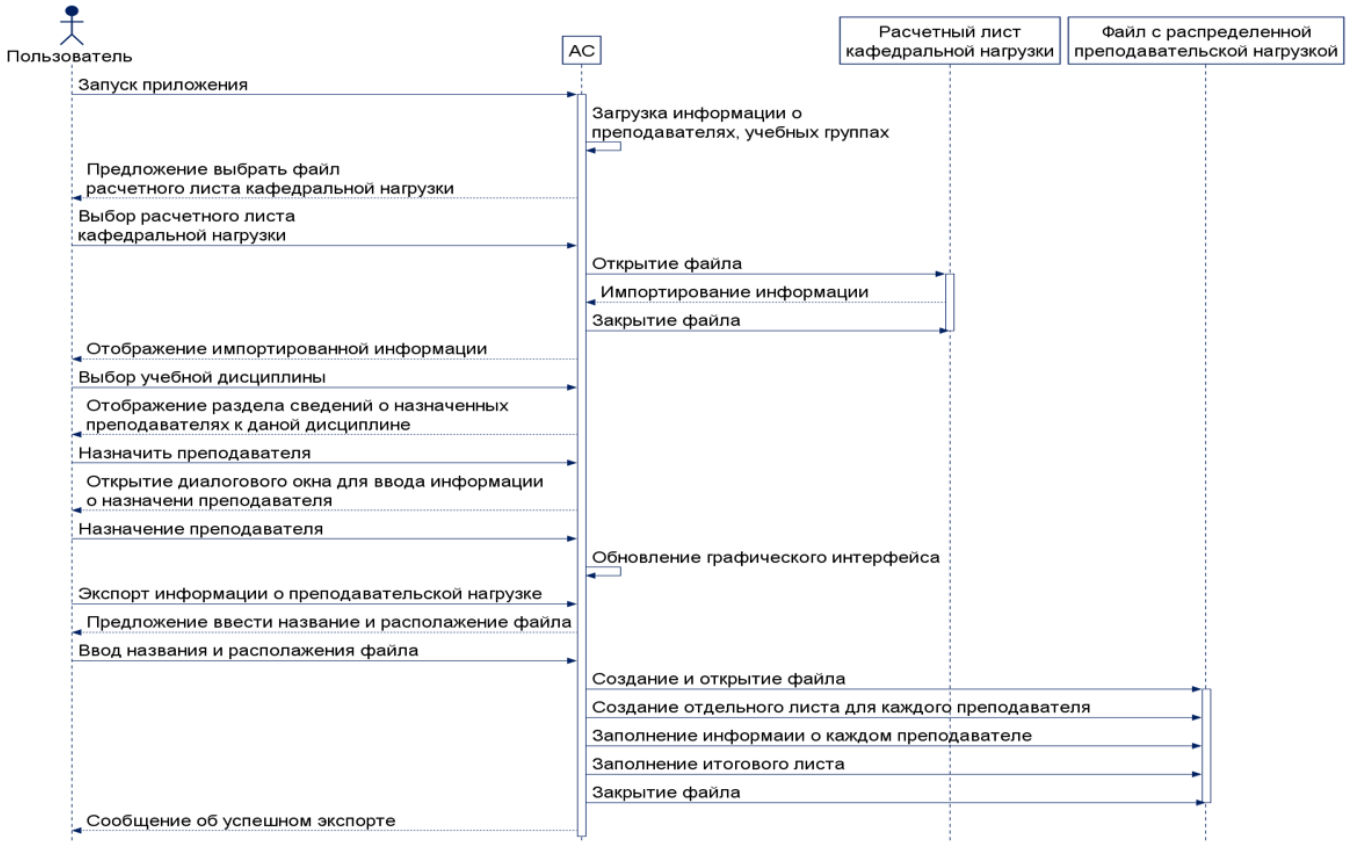


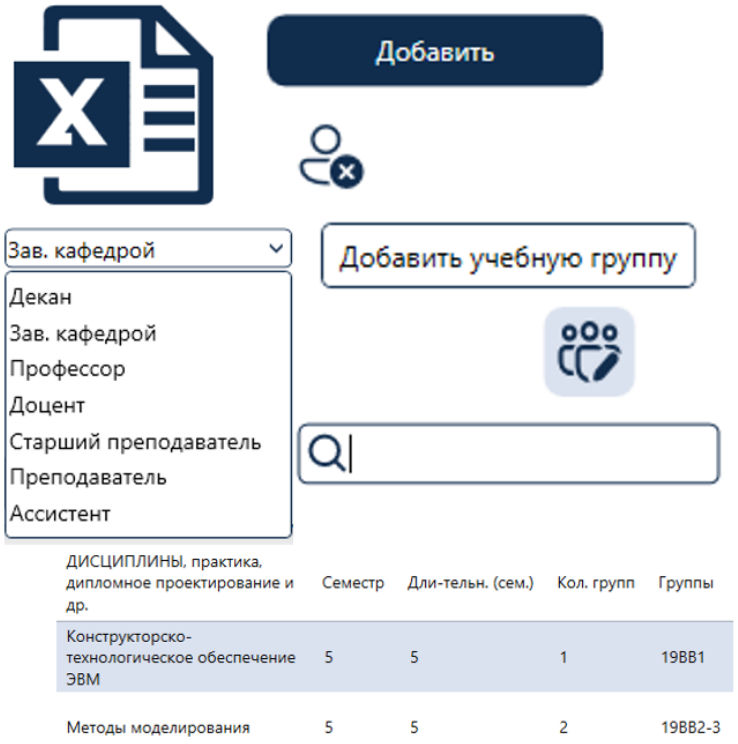
Диаграмма последовательностей



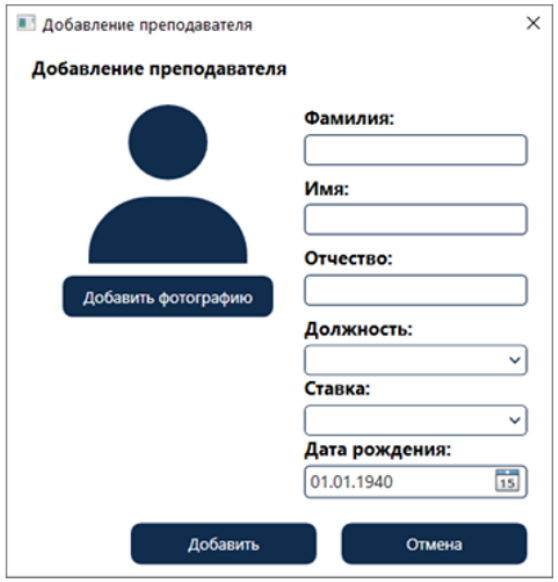
Создание стилей WPF

В процессе разработки были созданы словари ресурсов, содержащие стили для каждого графического элемента используемого в приложении:

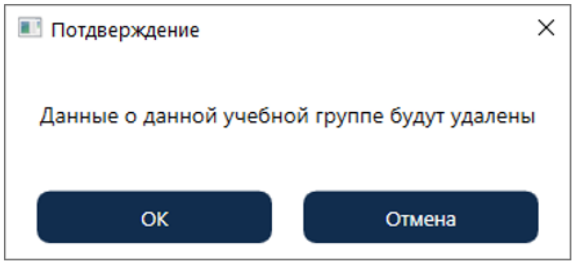
- Buttons.xaml
- Colors.xaml
- ComboBoxes.xaml
- Converters.xaml
- DataGrids.xaml
- DatePicker.xaml
- Icons.xaml
- TabsControls.xaml
- Text.xaml
- ToolTips.xaml



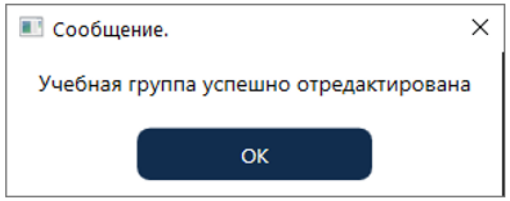
Диалоговые окна



Диалоговое окно AddDIalog



Диалоговое окно ConfirmDialog



Диалоговое окно NotificationDialog

Разработка классов моделей

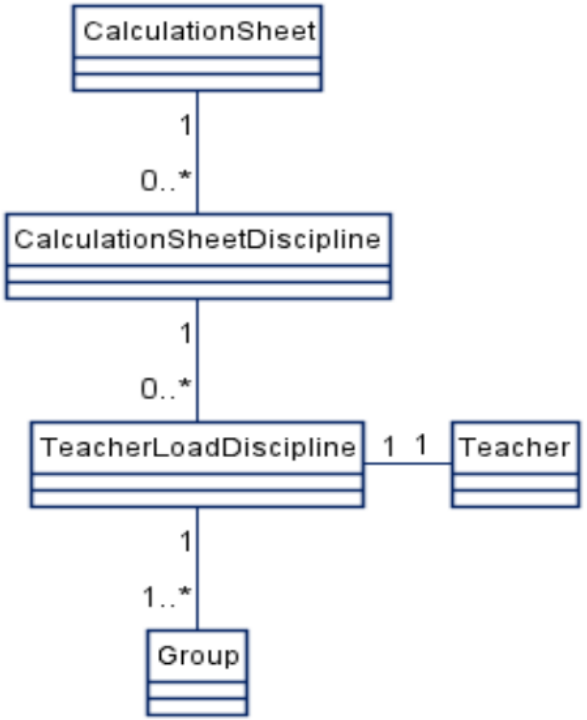
CalculationSheet
Класс CalculationSheet используется в качестве модели для хранения информации из расчетного листа кафедральной нагрузки.

CalculationSheetDisciplines
Класс CalculationSheetDisciplines предназначен для хранения данных учебных дисциплин.

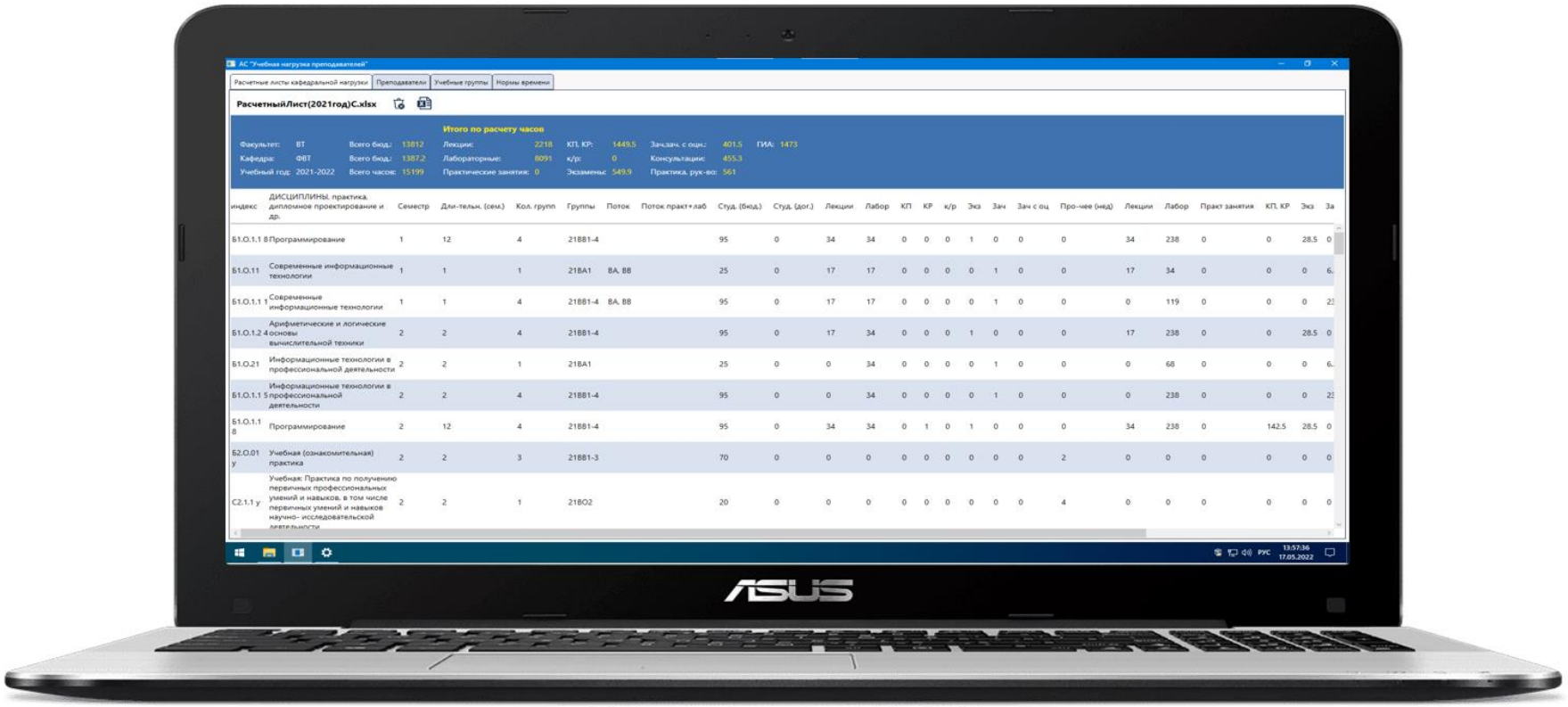
TeacherLoadDiscipline
Класс используется в качестве модели назначения преподавателя к учебной дисциплине.

Group
Класс Group необходим для хранения информации об учебных группах.

Teacher
Класс Teacher является моделью для преподавателей.



Результат


АС "Учебная нагрузка преподавателей"
Расчетные листы кафедральной нагрузки
Преподаватели
Учебные группы
Нормы времени
РасчетныйЛист(2021год)С.xlsx
Итого по расчету часов
Факультет: ВТ
Кафедра: ФВТ
Учебный год: 2021-2022
Всего бюд.: 13812
Всего бюд.: 1387.2
Всего часов: 15199
Лекции: 2218
Лабораторные: 8091
Практические занятия: 0
КП, КР: 1449.5
к/р: 0
Экзамены: 549.9
Зачзач. с оцн.: 401.5
Консультации: 455.3
Практика, рук-во: 561
ГИА: 1473
ASUS

Заключение

В ходе выполнения выпускной квалификационной работы была разработана автоматизированная система планирования учебной нагрузки преподавателей на языке C# . Графический интерфейс был реализован с помощью фреймворка WPF. Приложение было выдвинуто на регистрацию ЭВМ.

Разработанное приложение позволяет:

- импортировать информацию о учебных дисциплинах из расчетного листа кафедральной нагрузки;
- добавлять данные преподавателей (ФИО, должность, ставка, дата рождения, фото);
- редактировать контингент студенческих групп;
- производить планирование учебной нагрузки для преподавателей;
- экспортировать учебную нагрузку в файл формата xlsx, по всем требованиям учебного управления;
- автоматизировать расстановку формул;
- автоматизировать заполнение таблиц;
- редактировать нормы времени, в случае необходимости.

Возможно усовершенствовать разработанную автоматизированную систему, добавив:

- взаимодействие с базой данных
- возможность работы несколькими расчетными листами одновременно
- функционал по сохранению информации в процессе работы программы
- формирование поручений для составления расписания.

Спасибо за внимание!