

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

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

БАКАЛАВРСКАЯ РАБОТА

на тему

СИСТЕМА АВТОМАТИЗАЦИИ СОСТАВЛЕНИЯ
УЧЕБНОГО РАСПИСАНИЯ С ИСПОЛЬЗОВАНИЕМ
МЕТОДОВ ОПТИМИЗАЦИИ И УЧЕТОМ
РЕСУРСНЫХ ОГРАНИЧЕНИЙ

Студент

Юртаев 26.05.2026
(подпись, дата)

Юртаев Дмитрий Юрьевич
(ФИО полностью)

Руководитель

Е 26.05.2026
(подпись, дата)

Гурин Е.И.
(фамилия, инициалы)

Нормоконтролёр

А 26.05.2026
(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой

Митрохин
(подпись)

Митрохин М.А.
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

02 июня 2026г. № 8)

Секретарь ГЭК

Е
(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2026

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

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

«07» 04 2026 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Юртаев Дмитрий Юрьевич гр. 22ВВП1 факультета ВТ

2. Тема работы Система автоматизации составления учебного расписания с использованием методов оптимизации и учетом ресурсных ограничений

Тема утверждена приказом ПГУ № 347/о от " 7 " апреля 2026 г.

3. Руководитель бакалаврской работы Гурин Е.И.

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

Разработать систему автоматизации составления учебного расписания с использованием методов оптимизации и учетом ресурсных ограничений. Входными данными является информация об учебных группах, дисциплинах, преподавателях, аудиториях и их характеристиках, временных интервалах, учебных сменах и различного вида ограничениях, которые влияют на составление расписания.

5. Перечень подлежащих разработке вопросов

Необходимо провести анализ предметной области, изучить существующие аналоги, определить требования к системе, произвести выбор и настройку инструментов разработки, спроектировать архитектуру и различные модели в системе, разработать систему, провести тестирование готового приложения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	16	07.04.2026 – 08.04.2026	
2	Изучение аналогов	24	09.04.2026 – 11.04.2026	
3	Определение требований и постановка задачи	16	12.04.2026 – 13.04.2026	
4	Выбор и настройка инструментов разработки	32	14.04.2026 – 17.04.2026	
5	Проектирование архитектуры и различных моделей	56	18.04.2026 – 24.04.2026	
6	Разработка системы	120	25.04.2026 – 09.05.2026	
7	Тестирование и отладка	48	09.05.2026 – 14.05.2026	
8	Оформление пояснительной записки	56	15.05.2026 – 21.05.2026	

Дата выдачи задания " 7 " апреля 2026 г.

Руководитель бакалаврской работы  07.04.2026 Гурин Е.И.
(подпись, дата)

Задание к исполнению принял студент  07.04.2026 Юртаев Д.Ю.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
(подпись, дата)
28.05.2026

Реферат

Бакалаврская работа содержит: 134 листов, 15 рисунков, 16 таблиц, 20 источников и 3 приложения.

СИСТЕМА АВТОМАТИЗАЦИИ, УЧЕБНОЕ РАСПИСАНИЕ, МЕТОДЫ ОПТИМИЗАЦИИ, РЕСУРСНЫЕ ОГРАНИЧЕНИЯ, PYTHON, STREAMLIT, OR-TOOLS CP-SAT, PANDAS, OPENPYXL, SQLITE

Цель работы – разработка системы автоматизации составления учебного расписания с использованием методов оптимизации и учетом ресурсных ограничений.

Для достижения поставленной цели был проведен анализ предметной области, сравнение аналогов, определены требования к системе, спроектирована модель входных данных, модель ограничений и математическая модель, а также спроектирован алгоритм составления расписания.

В результате проделанной работы была разработана система автоматизации составления учебного расписания с использованием Streamlit, и языка программирования Python. Для распределения занятий применялась библиотека OR-Tools CP-SAT, которая осуществляет поиск решений с учетом ограничений. Для работы с таблицами были использованы библиотеки pandas и openpyxl. Для хранения истории последних составленных расписаний применялась база данных SQLite. Данная система автоматизации имеет возможность одновременного учета различных факторов: учебные группы, преподаватели, дисциплины, типы занятий, аудитории и их характеристики, временные интервалы, дни военной подготовки, учебные смены и индивидуальные предпочтения и ограничения преподавателей.

ПГУ 1.090301.27.001 ПЗ

Изм	Лист	№ докум.	Подпись	Дата	Система автоматизации составления учебного расписания с использованием методов оптимизации и учетом ресурсных ограничений. Пояснительная записка	Литера	Лист	Листов
Разраб.		Юртаев Д.Ю.		26.05.26				
Пров.		Гурин Е.И.		26.05.26			2	67
Н. Контр.		Бычков А.С.		26.05.26				
Утв.		Митрохин М.А.		26.05.26				

ФВТ, каф. ВТ
гр. 22ВВП1

Содержание

Введение.....	8
1 Система автоматизации составления учебного расписания с использованием методов оптимизации и учётом ресурсных ограничений.....	9
1.1 Анализ предметной области	9
1.1.1 Актуальность разработки.....	11
1.1.2 Анализ существующих аналогов	11
1.1.3 Обоснование актуальности разработки программы.....	16
1.1.4 Постановка задачи	18
1.2 Выбор и настройка инструментов разработки.....	19
1.2.1 Выбор языка программирования.....	19
1.2.2 Выбор платформы и среды разработки	21
1.2.3 Выбор фреймворка для разработки пользовательского интерфейса	23
1.2.4 Выбор средства хранения данных.....	25
1.2.5 Выбор библиотек для обработки табличных данных и работы с файлами Excel.....	27
1.2.6 Настройка инструментов разработки	29
1.3 Проектирование системы автоматизации	32
1.3.1 Архитектура приложения.....	32
1.3.2 Модель входных данных	34
1.3.3 Модель ограничений расписания.....	37
1.3.4 Математическая модель	40
1.3.5 Проектирование хранения истории расписаний.....	44
1.4 Разработка системы автоматизации	47
1.4.1 Реализация загрузки и проверки входных данных.....	47
1.4.2 Создание шаблона входного Excel-файла	50
1.4.3 Разработка настройки дополнительных параметров	51
1.4.4 Реализация алгоритма составления расписания	53
1.4.5 Разработка способа отображения и редактирования результата	56

1.4.6 Реализация экспорта расписания	57
1.4.7 Реализация хранения истории расписаний	57
1.5 Тестирование и отладка.....	58
2 Безопасность жизнедеятельности.....	61
Заключение	64
Список использованных источников	66
Приложение А – UML-диаграммы	68
Приложение Б – Листинг программы	73
Приложение В – Презентация.....	128

Перечень сокращений и обозначений

В данной работе используются следующие обозначения и сокращения

БД — база данных

СА — система автоматизации

СУБД — система управления базами данных

BLOB — тип данных для хранения двоичной информации, используемый для хранения расписания в БД

ER-модель — модель «сущность — связь», используемая для описания структуры данных

openruhl — библиотека языка Python, используемая для создания и обработки Excel-файлов

OR-Tools CP-SAT — библиотека и решатель, используемые для составления расписания с учетом ограничений

pandas — библиотека языка Python, используемая для работы с табличными данными

SQLite — встраиваемая СУБД, используемая для локального хранения истории расписаний

Streamlit — фреймворк языка Python, используемый для создания веб-интерфейса приложения

UML — унифицированный язык моделирования

Введение

В различных учебных заведениях формирование расписания является важной организационной задачей. Оно упорядочивает образовательный процесс и демонстрирует распределение учебных групп, дисциплин и преподавателей по временным интервалам.

Задача составления учебного расписания вручную очень трудоемкая, так как требует внимательности и учета множества факторов и ограничений. При этом, увеличивая количество входных данных, число различных вариантов распределения сильно возрастает и затрудняет работу. Из всего вышеперечисленного следует вывод, что данная задача относится к комбинаторным задачам оптимизации [1].

Актуальность темы данной выпускной квалификационной работы заключается в том, что необходимо автоматизировать формирование учебного расписания, чтобы снизить трудозатраты и минимизировать ошибки.

Целью данной работы является реализация системы автоматизации составления учебного расписания с использованием методов оптимизации и учетом ресурсных ограничений.

Практическая значимость заключается в том, что данная система может использоваться в учебных заведениях для облегчения труда сотрудников, которые занимаются планированием учебного процесса.

1 Система автоматизации составления учебного расписания с использованием методов оптимизации и учётом ресурсных ограничений

1.1 Анализ предметной области

Расписание как инструмент для организации процессов применяется в различных областях, таких как производственная деятельность, транспортные сети, сфера обслуживания, образовательная деятельность и тд. В данной работе рассматривается задача составления расписания в системе образования, а конкретно формирование учебного расписания.

Предметной областью данной системы можно считать сам процесс составления и редактирования расписания занятий в образовательных учреждениях среднего профессионального и высшего образования.

Расписание представляет из себя набор структурированных данных, подразумевающий распределение учебных дисциплин по определенным временным промежуткам, учебным группам, аудиториям и преподавателям. Процесс формирования учебного расписания является трудоемкой организационной задачей, которая требует учета многих факторов и ресурсных ограничений.

Основными элементами данной предметной области являются:

- учебные дисциплины;
- учебные группы;
- преподаватели;
- аудитории;
- временные промежутки (пары, дни, недели);
- дополнительные ограничивающие факторы и требования.

Составление расписания подразумевает собой распределение учебных дисциплин таким образом, чтобы были выполнены следующие требования:

- соответствие организационным и санитарным нормам;

- отсутствие пересечений учебных дисциплин в одном временном промежутке;
- соответствие аудитории преподаваемой дисциплине;
- учет индивидуальных возможностей преподавателя и тд.

Одной из главных сложностей данной предметной области является одновременный учет множества ограничений, которые условно можно поделить следующим образом.

Конфликтные ограничения:

- один преподаватель не может в один момент времени вести более одной учебной дисциплины;
- студенты одной учебной группы не могут в один момент времени изучать более одной учебной дисциплины и тд.

Ограничения рациональности расписания (иногда можно пренебречь, при условии, что нет других вариантов):

- минимизация «окон» в расписании;
- равномерное распределение занятий;
- учет дополнительных предпочтений преподавателя и тд.

Составление расписания относится к комбинаторным задачам оптимизации [1], так как требуется выбирать самое подходящее решение из большого множества возможных вариантов. Причем количество этих вариантов стремительно растет с увеличением количества учебных групп, дисциплин, аудиторий и преподавателей, что делает эту задачу более сложной.

Во многих образовательных учреждениях составление расписания выполняется вручную, без использования систем автоматизации, что влечет за собой некоторые проблемы:

- очень высокая трудоемкость;
- большая вероятность появления ошибки;
- сложность редактирования готового расписания.

Таким образом, возникает потребность в разработке автоматизированной системы, которая будет способна эффективно составлять расписание с учетом оптимизации и ресурсных ограничений.

1.1.1 Актуальность разработки

Разработка данной системы автоматизации является актуальной, ввиду усложнения современного образовательного процесса и роста количества данных, подлежащих структурированию.

Многие современные учебные учреждения обладают следующими признаками:

- большое количество студентов и учебных групп;
- широкий спектр образовательных программ и дисциплин;
- потребность редактировать учебный процесс и управлять им;
- стремление использовать имеющиеся ресурсы максимально эффективно.

В связи с этим, в современных реалиях возникает потребность во внедрении различных информационных технологий и автоматизаций в образовательные процессы [1], в том числе для составления учебного расписания.

При использовании современных способов разработки и оптимизации возможно выполнить следующие задачи:

- минимизировать трудозатраты на составление расписания;
- избежать пересечений и конфликтов;
- учитывать множество ограничений;
- эффективно управлять и вносить изменения в образовательный процесс.

Ввиду этого можно сделать вывод, что разработка системы автоматизации составления расписания действительно является актуальной задачей, способной повысить качество и эффективность образовательного процесса.

1.1.2 Анализ существующих аналогов

В настоящее время разработано несколько видов специализированного программного обеспечения, которые предназначены для автоматизации составления расписания. Подобные системы направлены на использование в образовательных учреждениях и способны учитывать большое количество

требований и ограничений, которые возникают при распределении учебного процесса.

Для того, чтобы создать своё качественное решение, изучим наиболее популярные аналоги.

Первый рассматриваемый аналог – это aSc TimeTables.

aSc TimeTables на данный момент одна из самых популярных систем [2] для автоматизированного составления расписания, которая используется в школах и учреждениях среднего профессионального образования.

Она предоставляет возможность автоматически расставлять занятия по определенным временным промежуткам, позволяет вести учет нагрузки преподавателей, дает возможность задать различные требования, ограничения и правила и тд. Примерный вариант расписания, составленного с помощью данной системы представлены на рисунке 1. На рисунке 1 также отмечены следующие области системы :1 — меню; 2 — инструментальная панель; 3 — расписание; 4 — названия строк; 5 — панель текущих операций; 6 — краткая информация о выбранном уроке.

Сам алгоритм распределения основан на переборе возможных вариантов, с применением эвристических методов поиска [2], с учетом заданных требований и правил.

Преимущества данной системы:

- готовый, понятный интерфейс;
- большой спектр возможных ограничений;
- широкое практическое применение.

Недостатки этого решения:

- зависимость от коммерческой лицензии;
- невозможность полноценного учета всех специфических потребностей конкретного пользователя;
- недоступность в ряде стран, в том числе в России.

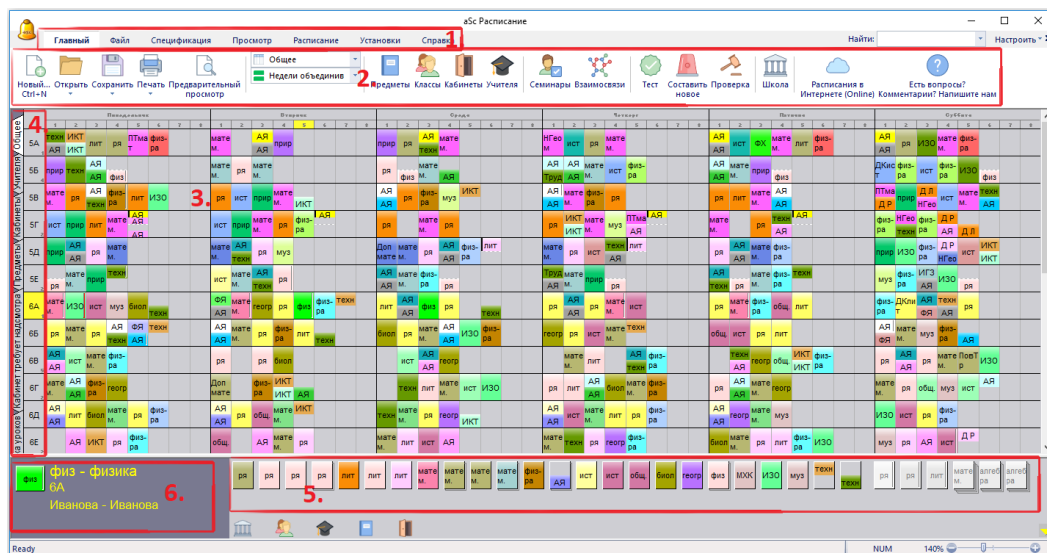


Рисунок 1 – основное окно системы

Второй аналог – это UniTime.

UniTime это программное обеспечение [3], которое специализируется в первую очередь для использования в высших учебных заведениях.

Данная система предоставляет подобные возможности что и предыдущая, при этом использует методы математической оптимизации [3, 4], и так же способна учитывать множество ресурсов и параметров: ресурсы учебного заведения, различные направления подготовки и дисциплины, индивидуальные возможности и предпочтения преподавателей, ограничения по времени, дате, аудитории и тд. Примерный вариант расписания, составленного с помощью данной системы представлены на рисунке 2.

У данного решения есть ряд преимуществ:

- эффективные методы оптимизации;
- хорошая масштабируемость;
- доступная архитектура.

Также есть существенные недостатки:

- повышенная сложность настройки;
- трудность внедрения;
- значительные временные затраты на адаптацию.

Timetable

Filter

Export PDF Refresh

Timetable Legend

	7:30a	8:00a	8:30a	9:00a	9:30a	10:00a	10:30a	11:00a	11:30a	12:00p	12:30p	1:00p	1:30p	2:00p	2:30p	3:00p	3:30p	4:00p	4:30p	5:00p
PHYS 112 (268)																				
Mon			ENGR 126R Lec 1 4 54 0	OLS 274 Lec 1 0 0 0	MA 154 Lec 2 0 10 0	OLS 274 Lec 3 24 1 0	PHYS 219 Lec 1 0 0 0	OLS 274 Lec 2 0 0 0	CGT 163 Lec 1 4 3 0	ENGR 126A Lec 1 0 0 0	ENGR 126H Lec 1 4 69 0	EPCS 101 Lec 1 0 0 0								
Tue	OLS 252 Lec 1 4 5 3	PHYS 272 Lec 1 0 0 0	PHYS 221 Lec 1 0 0 0	PHYS 241 Lec 1 0 0 0	PHYS 241 Lec 2 0 0 0	PHYS 241 Lec 3 0 0 0	PSY 335 Lec 1 0 0 0	SOC 100 Lec 10 32 4 4	HIST 152 Lec 1 0 14 0											
Wed		ENGR 126R Lec 1 4 54 0	OLS 274 Lec 1 0 0 0	MA 154 Lec 2 0 10 0	OLS 274 Lec 3 24 1 0	PHYS 219 Lec 1 0 0 0	OLS 274 Lec 2 0 0 0	CGT 163 LabP 1 4 5 0	ENGR 126A Lec 1 0 0 0	ENGR 126H Lec 1 4 69 0										
Thu	OLS 252 Lec 1 4 5 3	PHYS 272 Lec 1 0 0 0	PHYS 221 Lec 1 0 0 0	PHYS 241 Lec 1 0 0 0	PHYS 241 Lec 2 0 0 0	PHYS 241 Lec 3 0 0 0	PSY 335 Lec 1 0 0 0	SOC 100 Lec 10 32 4 4	HIST 152 Lec 1 0 14 0											
Fri		PHYS 221 Lec 1 0 0 0				MA 154 Lec 2 0 10 0				PHYS 219 Lec 1 0 0 0	PHYS 219 Lec 2 0 0 0	PHYS 218 Lec 1 0 0 0	PHYS 218 Lec 2 0 0 0							
PHYS 114 (273)																				
Mon	CGT 163 Lec 2 4 5 4	PHYS 214 Lec 1 0 0 0	ANTH 205 Lec 1 16 61 0	PHYS 172H Lec 1 40 8 4	MA 165 Lec 5 0 15 0	PHYS 218 Lec 1 0 0 0	PHYS 218 Lec 2 0 0 0	AGEC 217 Lec 2 0 0 0	AGEC 217 Lec 3 0 0 0	PSY 200 Lec 1 24 38 0										
Tue		PHYS 220 Lec 1 0 0 0	PHYS 220 Lec 2 0 11 0	PHYS 172 Lec 1 0 0 0	PHYS 172 Lec 2 0 0 0	PHYS 172 Lec 3 0 0 0	CGT 141 Lec 1 4 5 0	AGEC 217 Lec 2 0 0 0	AGEC 217 Lec 3 0 0 0	PSY 200 Lec 1 24 38 0										
Wed	CGT 163 LabP 2 4 5 4	PHYS 214 Lec 1 0 0 0	ANTH 205 Lec 1 16 61 0	ENGR 100H Lec 1a 4 6 0 Week 1	MA 165 Lec 5 0 15 0	PHYS 218 Lec 1 0 0 0	PHYS 218 Lec 2 0 0 0	AGEC 217 Lec 2 0 0 0	AGEC 217 Lec 3 0 0 0	PSY 200 Lec 1 24 38 0										
				ENGR 100H Lec 1b 4 6 0 Week 4																
				ENGR 100H Lec 1																

Рисунок 2 – пример расписания в UniTime

Третий аналог – это 1С:Расписание учебных занятий.

Система «1С:Расписание учебных занятий» является отечественным программным обеспечением [5], которая так же предназначена для автоматизации планирования учебного расписания.

Она разработана на базе «1С:Предприятие» и получила широкое распространение в российских образовательных учреждениях. Данная система нацелена на внедрение в существующие модули образовательной организации и позволяет использовать единое информационное пространство. Фрагмент расписания, составленного в данной системе, представлен на рисунке 3

Преимущества системы:

- адаптация к Российским стандартам организации учебного процесса и нормативным требованиям;
- хранение, структурирование больших объемов данных.

Недостатки решения:

- высокая стоимость внедрения и использования;
- невысокая гибкость при решении нестандартных ситуаций;
- ориентирована на комплексную автоматизацию всего образовательного процесса.

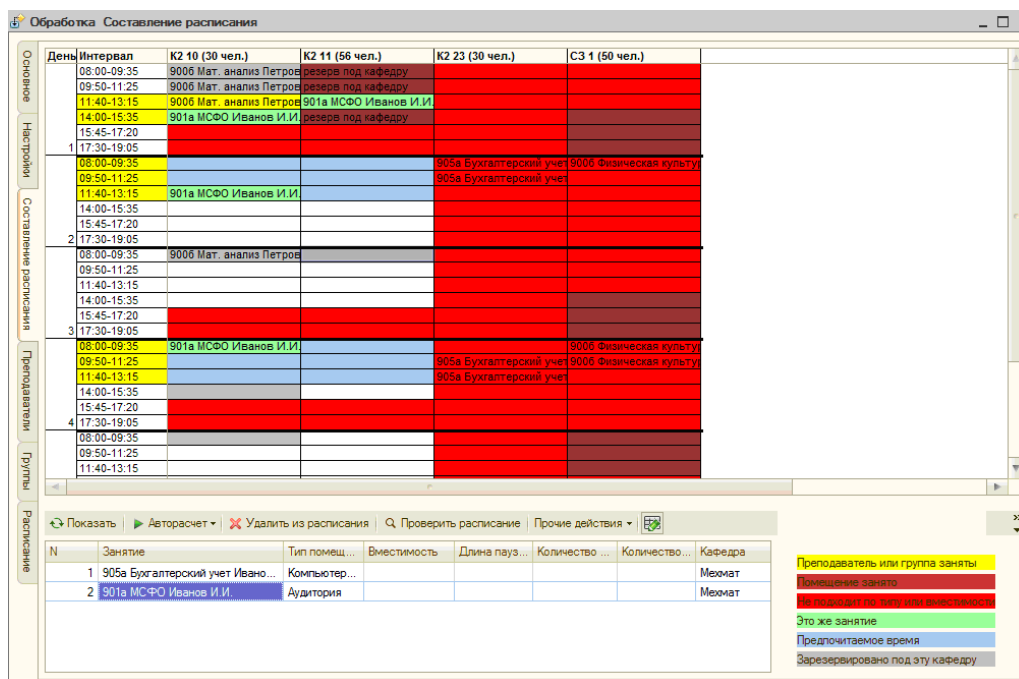


Рисунок 3 – Фрагмент расписания в системе «1С:Расписание учебных занятий»

Для более наглядного сравнения, в таблице 1 собрана основная информация о рассматриваемых аналогах.

Таблица 1 – Сравнение аналогов

Критерий сравнения	aSc TimeTables	UniTime	1С:Расписание учебных занятий
Целевое применение	Школы и учреждения среднего профессионального образования	Высшие учебные заведения	Все образовательные учреждения
Назначение системы	Автоматизированное составление учебного расписания	Планирование расписания с учетом большого количества параметров и ресурсов	Автоматизация планирования учебных занятий в пределах платформы 1С
Учет нагрузки преподавателей	Поддерживается	Поддерживается	Поддерживается
Учет ограничений и дополнительных настроек	Позволяет задавать различные требования, ограничения и правила	Учитывает временные, ресурсные, аудиторные и индивидуальные ограничения	Учитывает нормативные и организационные требования учебного процесса
Способ составления расписания	Перебор возможных вариантов с использованием эвристических методов	Применение методов математической оптимизации	Применение механизмов платформы 1С:Предприятие

Критерий сравнения	aSc TimeTables	UniTime	1С:Расписание учебных занятий
Гибкость настройки	Достаточно высокая	Высокая, но требует сложной настройки	Ограничена типовой логикой платформы и комплексной автоматизацией
Масштабируемость	Подходит для школ и СПО	Масштабируется для различных вузов	Подходит для организаций, уже использующих продукты 1С
Преимущества	Понятный интерфейс, широкий набор ограничений, большое практическое применение	Эффективные методы оптимизации, хорошая масштабируемость	Адаптация к российским стандартам, хранение больших объемов данных
Недостатки	Коммерческая лицензия, невозможность учета всех специфических требований, недоступность в ряде стран	Сложность настройки, трудность внедрения, большие временные затраты на адаптацию	Высокая стоимость внедрения, невысокая гибкость, ориентация на комплексную автоматизацию
Вывод	Хороша для базовых задач, но не всегда подходит для особых требований	Хорошее решение для вузов, но требует серьезной настройки и внедрения	Подходит для организаций уже сотрудничающих с 1С, но может быть избыточной для отдельной задачи составления расписания

В результате анализа стало ясно, что имеющиеся системы автоматизации составления расписания, безусловно, имеют ряд преимуществ, однако обладают существенными недостатками и ограничениями.

Таким образом, выявляется потребность в разработке такой системы, которая будет максимально оптимизирована под конкретную задачу учебного заведения, сможет учитывать все ресурсные ограничения, будет использовать эффективные методы оптимизации и поддерживать гибкую работу с требованиями и правилами.

1.1.3 Обоснование актуальности разработки программы

Разрабатываемая в рамках данной работы система автоматизации составления учебного расписания направлена на применение в условиях образовательного учреждения и учитывает необходимые требования, ограничения и особенности организации учебного процесса. В отличие от рассмотренных программных продуктов, разрабатываемое решение предполагает адаптацию под

текущие условия функционирования учебного заведения, включая структуру учебных программ, особенности распределения предметов и действующие организационные ограничения.

Одна из важнейших особенностей данной системы является использование методов оптимизации при составлении расписания [1, 4]. В текущих условиях формирование расписания происходит вручную и с использованием простых цифровых инструментов, что влечет за собой неравномерное распределение нагрузки, сложность учета индивидуальных пожеланий и тд. Применение методов оптимизации позволит решить эту задачу путем поиска наилучшего решения из множества возможных вариантов. Такой подход позволяет добиться более качественного итогового результата, уменьшить влияние человеческого фактора, снизить нагрузку на сотрудников и учесть индивидуальные возможности преподавателей.

Реализуемая система будет способна учитывать не только базовые требования, но и более специфические ресурсные ограничения. Например, доступность аудитории и пригодность её для конкретного занятия, наличие специального оборудования для проведения дисциплины и тд. В результате становится возможным более эффективное использование материально-технической базы учебного заведения.

Система также будет иметь высокую степень гибкости и адаптивности для работы с различным набором входных данных, такими как состав учебных групп, список дисциплин, количество учебных часов и тд. Изменение этих данных не потребует изменения алгоритма, что позволит использовать систему в различных условиях. При этом пользователи смогут легко редактировать данные и получить изменения в кратчайшие сроки.

Важная особенность данной системы – это независимость от коммерческих программных продуктов. Применение общедоступных, открытых технологий и самостоятельная разработка позволяет минимизировать расходы на внедрение и последующее использование, а также облегчает дальнейшее совершенствование и модификацию программного решения без лицензионных ограничений.

Система будет ориентирована на представление результата в принятом формате представления, что позволяет не обрабатывать дополнительно выходные данные и сразу использовать полученный результат в образовательном процессе.

Таким образом данная система обладает множеством существенных преимуществ в сравнении с имеющимися аналогами. Она объединяет в себе адаптивность, гибкость и методы оптимизации, что позволяет эффективно использовать её для планирования учебного процесса в текущих условиях функционирования образовательного учреждения.

1.1.4 Постановка задачи

Целью данной работы является разработка системы автоматизации составления учебного расписания с использованием методов оптимизации и учетом ресурсных ограничений.

Для реализации данной цели необходимо решить следующие задачи:

- проанализировать предметную область и существующие аналоги, для выявления эффективных технологий и актуальных функций;
- определить основные требования и ограничения к разрабатываемой системе;
- разработать модель представления данных, включающую описание учебных групп, преподавателей, дисциплин, аудиторий, временных промежутков;
- составить математическую модель задачи формирования расписания, включающую систему ограничений, множество переменных, критерии оптимальности;
- реализовать алгоритм учитывая методы оптимизации;
- разработать программное решение, позволяющее загружать исходные данные, генерировать расписание, отображать результаты в понятной форме;
- удостовериться в корректности работы системы, протестировав ее на примере различных входных данных;
- провести анализ итоговых результатов работы системы.

В результате выполнения работы должна быть разработана система, которая позволяет автоматически составлять расписание занятий, учитывая все

ограничения, правила и требования, минимизировать пересечения и конфликты, обеспечивать понятное представление результата.

На рисунке 4 представлена диаграмма вариантов использования системы.

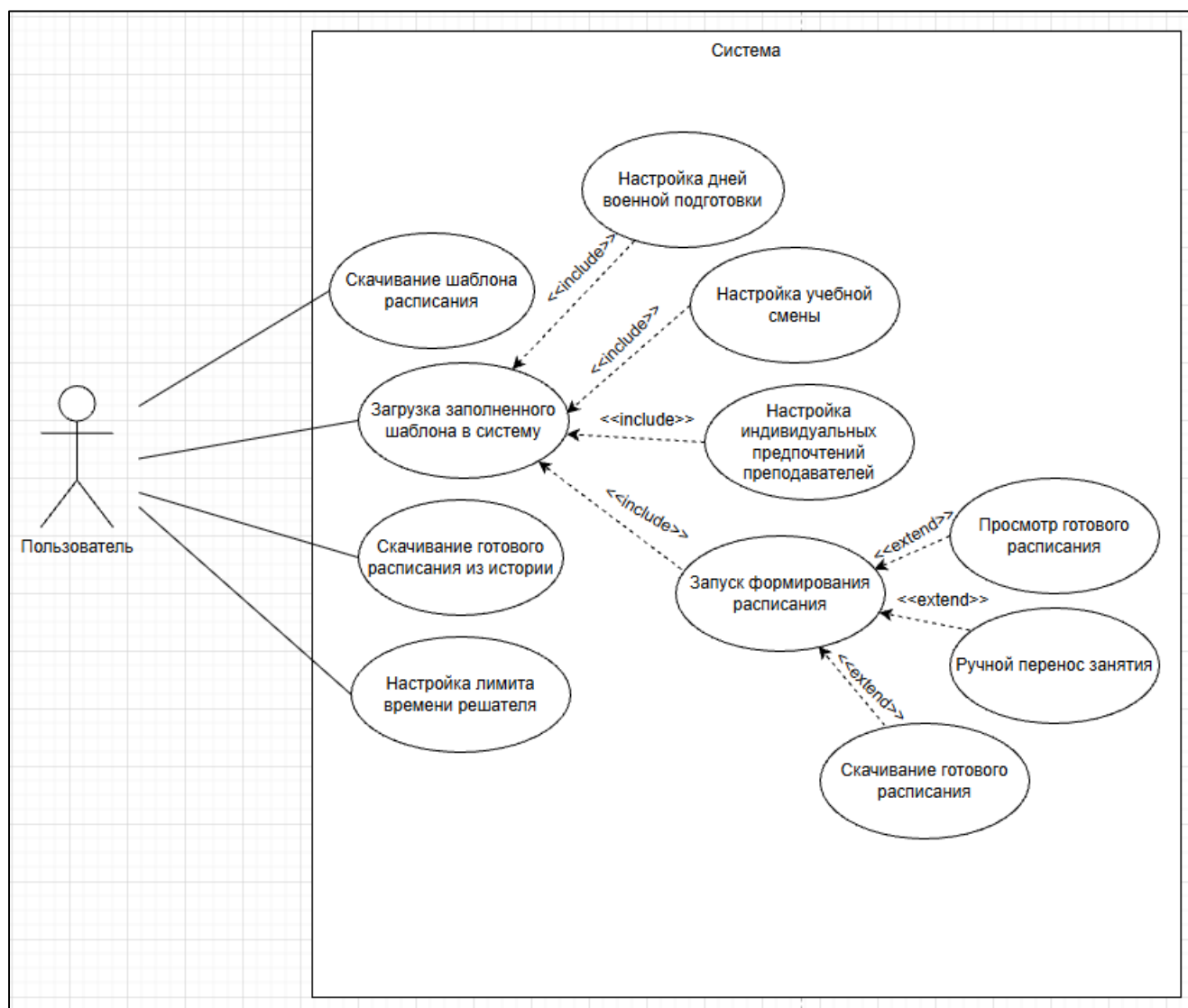


Рисунок 4 — Диаграмма вариантов использования системы

1.2 Выбор и настройка инструментов разработки

1.2.1 Выбор языка программирования

Одним из важнейших этапов подготовки к разработке системы является выбор языка программирования. Именно он влияет на то, насколько будет удобно и быстро реализовать логику программы, разработать графический интерфейс,

организовать взаимодействие с базой данных и в дальнейшем использовать, корректировать, совершенствовать систему.

Для реализации системы были рассмотрены несколько популярных языков программирования такие как C#, Java и Python.

Язык C# очень распространен в создании десктопных приложений [6], по большей части под ОС Windows. Он дает возможность разработчикам использовать удобные инструменты для реализации графического интерфейса, хорошо подходит для создания масштабных коммерческих приложений и систем, к тому же обладает хорошей производительностью. Однако его полноценное применения направлено больше на взаимодействие с экосистемой Microsoft, часть средств разработки и настройки направлена на нее. В системе, разрабатываемой в данной работе, такой подход может оказаться избыточным, так как целью является создание относительно быстро реализуемого приложения, легко сопровождаемого и по возможности компактного приложения.

Язык Java также является очень популярным средством разработки [7] различных прикладных систем. Его основными плюсами является кроссплатформенность, хорошая масштабируемость, строгая объектно-ориентированная модель, развитые инструменты разработки. На Java можно разработать различные десктопные приложения, распределенные системы и тд. Однако при решении задачи автоматизации составления расписания язык Java вносит в процесс некоторые усложнения. Например, возрастает объем служебного кода, усложняются требования к структуре системы. Также создание графического интерфейса и обработка основных пользовательских сценариев требует очень большого количества подготовительных действий.

В свою очередь язык Python позволяет сфокусироваться в основном на создании логики программы [8, 9]. Его особенность – это простой синтаксис, хорошая читаемость готового кода, малое количество служебного кода, разнообразие готовых библиотек для работы с различного типа данными, файлами, таблицами и графическим интерфейсом. Для системы автоматизации составления расписания это очень важно, так как большая часть логики системы

связана именно с сортировкой, фильтрацией, сравнением, проверкой ограничений и визуальном представлении информации.

Еще одним преимуществом языка Python является низкий порог входа и удобство ведения проекта. Код, который написан на Python, легко читать, анализировать и совершенствовать. Это влияет и на этап первичной реализации и на дальнейшее развитие программы, так как может возникнуть потребность добавить новые правила, отчеты, формы, или вообще изменить какую-то структуру.

Также большое количество готовых библиотек, подходящих под предметную область данной работы, послужило весомым аргументом для выбора языка Python. При помощи него можно удобно реализовать хранение информации в базе данных, загрузку и выгрузку информации в Excel, сделать автоматическую обработку табличных данных, сформировать графический интерфейс. Таким образом, язык Python дает возможность сразу охватить несколько областей системы: основная логика, работа с данными и отображение результата.

Учитывая все перечисленные особенности, для разработки системы в качестве основного языка программирования был выбран Python. Его использование позволяет реализовать относительно компактную систему с необходимым функционалом, без сложных дополнительных настроек, а также дает возможность удобно редактировать программу.

1.2.2 Выбор платформы и среды разработки

После того как выбран язык программирования необходимо определиться, какую программную среду лучше использовать для разработки. От этого во многом зависит удобство и скорость программирования, отладки, скорость поиска ошибок, использование библиотек и общая организация проекта.

Для разработки на Python существуют различные программные среды, например: PyCharm, Visual Studio Code, IDLE, Jupyter Notebook, Spyder и т.д. У каждой из них есть свои преимущества и недостатки.

PyCharm это одна из самых мощных программных сред. Она дает возможность использовать развитые средства навигации по коду, автодополнение,

рефакторинг кода, интеграцию с системами контроля версий и широкий спектр настроек проекта. Однако для системы, разрабатываемой в данной работе, такие разнообразные возможности могут быть излишними. К тому же, интерфейс программы достаточно непрост, следовательно, требует немало времени на изучение.

Visual Studio Code представляет собой универсальный редактор, который при использовании определенных расширений становится удобной средой разработки. Его основным преимуществом является гибкость и большое количество плагинов. Вместе с тем для стабильной работы проекта требуется предварительно настроить много инструментов, таких как: интерпретатор, расширения, отладчик, окружение и дополнительные компоненты, что увеличивает подготовительный этап разработки.

Spyder является интегрированной средой разработки, ориентированной на Python и входящей в состав платформы Anaconda. Ее важным достоинством является легкость освоения, простой редактор кода, наличие встроенной консоли, обозревателя переменных и основных средств для отладки программы. Для разрабатываемой системы, в котором необходимо совместить программирование, работу с данными и проверку промежуточных результатов, такая среда оказывается особенно удобной.

В качестве программной платформы была выбрана Anaconda [10]. Данный дистрибутив Python содержит не только интерпретатор языка, но и средства управления пакетами, готовую инфраструктуру для работы с библиотеками и ряд уже установленных инструментов. Это упрощает первоначальную настройку среды, снижает вероятность проблем с зависимостями и позволяет быстро подготовить рабочее окружение разработчика.

В качестве среды разработки был выбран Spyder [11], входящий в состав Anaconda. Такой выбор обусловлен несколькими причинами. Во-первых, Spyder прост в установке и не требует сложной дополнительной настройки. Во-вторых, он предоставляет понятный интерфейс, удобный для поэтапной разработки. В-третьих, в нем удобно выполнять тестовые фрагменты программы, проверять

					ПГУ 1.090301.27.001 ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

результаты работы функций и быстро локализовать ошибки. Для системы автоматизации составления расписания, где необходимо многократно проверять корректность обработки списков, таблиц и временных промежутков, это является существенным преимуществом.

Настройка среды разработки включала несколько этапов. Сначала была установлена платформа Anaconda, содержащая Python и набор базовых библиотек. После этого было создано отдельное виртуальное окружение проекта, в котором были размещены все необходимые пакеты. Такой подход позволяет обеспечить хорошую адаптивность разработки: при переносе проекта на другой компьютер можно установить те же версии библиотек и получить одинаковое поведение программы.

Таким образом, комбинация Anaconda и Spyder была выбрана как наиболее удобная для данной системы, ориентированной на быстрое создание приложения, обработку табличных данных и поэтапную отладку функциональности.

1.2.3 Выбор фреймворка для разработки пользовательского интерфейса

Так как разрабатываемая система предназначена для автоматизации составления и ведения учебного расписания, одним из ключевых требований к ней является наличие логичного и наглядного пользовательского интерфейса. Пользователь системы должен иметь возможность вводить исходные данные, редактировать сведения о преподавателях, учебных группах, дисциплинах и аудиториях, запускать составление расписания, просматривать результат и при необходимости сохранять его в удобном виде. Следовательно, в рамках проекта требуется инструмент, позволяющий достаточно быстро реализовать интерфейс прикладной системы без излишнего усложнения архитектуры.

При разработке приложений на Python для решения подобных задач могут использоваться разные подходы. Один из них связан с созданием классических настольных приложений на базе библиотек Tkinter или PyQt. Такой вариант предполагает формирование отдельного оконного интерфейса, работающего как самостоятельная программа в операционной системе. Другой подход заключается в разработке веб-приложения, интерфейс которого отображается в браузере. Для

задач, связанных с обработкой таблиц, вводом параметров, фильтрацией данных и визуальным представлением результатов, второй вариант является особенно удобным, так как не требует проектирования сложного оконного интерфейса и обеспечивает более современный способ взаимодействия пользователя с системой.

В рамках данной работы в качестве основного средства разработки пользовательского интерфейса был выбран фреймворк Streamlit [12, 13]. Данный инструмент предназначен для создания веб-приложений на языке Python [13] и позволяет объединить пользовательский интерфейс и вычислительную логику в рамках одного программного проекта. Это особенно важно для рассматриваемой системы, так как основная сложность задачи заключается не в разработке сложного дизайна, а в организации обработки данных, связанных с формированием учебного расписания.

Выбор Streamlit обусловлен рядом причин. Во-первых, данный фреймворк позволяет реализовать интерфейс приложения значительно быстрее по сравнению с классическими веб-технологиями, требующими отдельной разработки клиентской и серверной частей. Во-вторых, Streamlit хорошо подходит для приложений, в которых пользователь взаимодействует с таблицами, формами, переключателями, списками и результатами вычислений. В-третьих, использование данного инструмента позволяет разрабатывать систему в рамках единого языка программирования, что упрощает сопровождение проекта и делает его архитектуру более понятной.

С практической точки зрения применение Streamlit в рамках данной работы удобно тем, что разработчик может сосредоточиться именно на функциональных возможностях программы. Интерфейс создается с помощью встроенных компонентов, таких как поля ввода, кнопки, выпадающие списки, таблицы и блоки вывода сообщений. Это дает возможность реализовать основные сценарии работы пользователя без необходимости отдельно изучать HTML, CSS и JavaScript. Для проекта по автоматизации расписания такой подход является оправданным, поскольку позволяет в краткие сроки создать рабочее прикладное решение, пригодное для демонстрации и дальнейшего развития.

					ПГУ 1.090301.27.001 ПЗ	Лист
						24
Изм.	Лист	№ докум.	Подпись	Дата		

Настройка Streamlit в составе проекта включает установку соответствующей библиотеки в рабочее окружение Python, проверку корректности запуска приложения, настройку структуры файлов проекта и организацию модулей, отвечающих за различные части логики. В ходе настройки также определяется основной файл запуска, содержащий пользовательский интерфейс и логику перехода между основными разделами системы. Дополнительно может быть настроена боковая панель навигации, через которую пользователь получает доступ к разделам работы со справочниками, параметрами генерации и просмотру готового расписания.

С точки зрения структуры интерфейса Streamlit позволяет удобно организовать систему в виде последовательности логических блоков. Например, в одном разделе могут располагаться средства ввода и редактирования исходных данных, в другом — настройки формирования расписания, а в третьем — визуализация уже сформированного результата. При необходимости приложение может выводить таблицы, предупреждения о конфликтах, сообщения об ошибках и результаты экспорта. Таким образом, Streamlit обеспечивает достаточный набор средств для реализации всех базовых функций пользовательского интерфейса разрабатываемой системы.

Таким образом, выбор Streamlit в качестве фреймворка пользовательского интерфейса является обоснованным. Он соответствует целям проекта, обеспечивает высокую скорость разработки, позволяет создать понятное веб-приложение без чрезмерного усложнения архитектуры и открывает возможности для дальнейшего расширения системы.

1.2.4 Выбор средства хранения данных

Для функционирования системы автоматизации расписания необходимо организовать хранение информации о преподавателях, дисциплинах, группах, аудиториях, учебных неделях, временных промежутках и уже сформированных вариантах расписания. Без использования структурированного хранилища реализация подобной системы была бы затруднительной, поскольку данные

должны не только сохраняться между запусками программы, но и поддерживать поиск, изменение, удаление, сортировку и проверку связей между объектами.

В качестве возможных решений для хранения данных могут быть рассмотрены текстовые файлы, электронные таблицы и системы управления базами данных.

Использование обычных текстовых файлов для таких задач неудобно, поскольку при увеличении числа записей усложняется контроль целостности данных, а внесение изменений требует дополнительной логики обработки. Электронные таблицы, например, Excel, удобны как формат обмена и представления результатов, однако они не являются полноценной заменой базе данных, когда требуется поддерживать взаимосвязанные сущности и обеспечивать корректность операций обновления.

Поэтому для проекта целесообразно использовать систему управления базами данных. Среди наиболее распространенных вариантов можно выделить MySQL, PostgreSQL и SQLite.

MySQL и PostgreSQL являются полноценными серверными СУБД. Они хорошо подходят для многопользовательских систем, распределенных решений и крупных проектов, где требуется одновременный доступ большого числа пользователей. Однако для данной системы автоматизации применение серверной базы не всегда оправдано. Это связано с необходимостью установки и настройки отдельного сервера, созданием подключения, администрированием пользователей и дополнительными требованиями к сопровождению.

SQLite представляет собой встроенную СУБД [14], которая хранит всю базу данных в одном файле и не требует отдельного серверного процесса. Программа может работать локально, установка упрощается, а сопровождение системы не требует наличия отдельного администратора базы данных. При этом SQLite поддерживает основные возможности реляционных баз: таблицы, ключи, запросы, сортировку, фильтрацию и транзакции.

С учетом характера разрабатываемой системы в качестве средства хранения данных была выбрана SQLite. Это решение соответствует нескольким важным

					ПГУ 1.090301.27.001 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

требованиям. Во-первых, приложение предназначено для локального использования на рабочем компьютере или в рамках ограниченного числа пользователей. Во-вторых, объем данных в системе расписания относительно невелик по сравнению с промышленными информационными системами. В-третьих, важно минимизировать сложность внедрения программы: сотруднику достаточно установить приложение, после чего база данных будет создана автоматически или подключена из готового файла.

Дополнительным преимуществом SQLite является наличие встроенной поддержки в Python через модуль sqlite3 [14]. Это позволяет организовать взаимодействие с базой без подключения сторонних драйверов и без усложнения конфигурации проекта. Таким образом, вся работа с данными может быть реализована средствами, доступными непосредственно в используемой программной платформе.

Таким образом, выбор SQLite как средства хранения данных является обоснованным с точки зрения простоты внедрения, достаточной функциональности, минимальных требований к настройке и соответствия характеру разрабатываемой системы.

1.2.5 Выбор библиотек для обработки табличных данных и работы с файлами Excel

Особенностью задачи составления расписания является то, что большая часть исходной и итоговой информации имеет табличную форму. Сведения о преподавателях, группах, дисциплинах, учебной нагрузке, аудиториях и временных промежутках представляют собой структурированные таблицы. Кроме того, в практической деятельности образовательных учреждений обычно используются документы формата Excel, поэтому программное средство должно либо импортировать данные из таких файлов, либо формировать результат в удобном для печати и редактирования виде.

Для решения указанных задач в Python доступны различные библиотеки, среди которых особенно полезными являются pandas и openpyxl.

Библиотека `pandas` предназначена для обработки структурированных данных [15, 16]. Она позволяет загружать таблицы, фильтровать строки, сортировать значения, объединять данные из разных источников, группировать записи и выполнять иные операции анализа и преобразования. Для задачи автоматизации составления расписания это означает, что с ее помощью удобно подготавливать данные для проверки, искать пересечения, формировать сводные представления и упорядочивать готовое расписание по дням, группам или преподавателям.

Библиотека `openpyxl` используется для чтения и записи файлов формата Excel [17]. Она позволяет создавать книги и листы, заполнять ячейки, задавать стили, объединять области, устанавливать ширину столбцов и сохранять итоговые документы в удобном для пользователя виде. Для данной системы эта библиотека особенно ценна, поскольку итоговые таблицы удобно использовать в формате Excel.

Использование только встроенных средств Python для подобных операций значительно усложнило бы разработку. Поэтому для проекта было решено использовать именно `pandas` и `openpyxl` как наиболее подходящие библиотеки для табличной обработки и экспорта данных.

Библиотека `pandas` используется преимущественно на уровне обработки данных. С ее помощью могут быть сформированы промежуточные табличные представления, на основе которых выполняется проверка корректности составленного расписания. Например, можно определить, не назначен ли один и тот же преподаватель на две пары одновременно, не пересекаются ли занятия одной учебной группы, не используется ли одна и та же аудитория в один временной слот для разных дисциплин. Такие проверки особенно удобны именно в табличной форме.

Библиотека `openpyxl` применяется на этапе формирования итогового документа. С ее помощью создается электронная таблица расписания, в которой информация может быть представлена в наглядном виде: по дням недели, парам, учебным группам или преподавателям. При необходимости могут быть настроены объединенные заголовки, рамки таблиц, выравнивание содержимого ячеек и

другие элементы оформления. Это делает результат работы системы удобным для непосредственного практического использования.

Для повышения надежности импорта и экспорта данных были также предусмотрены меры контроля корректности форматов. При загрузке файлов необходимо проверять наличие обязательных столбцов, допустимость пустых значений, корректность типов данных и отсутствие критических ошибок в исходной таблице. Аналогично при экспорте результата формируется структура, соответствующая принятому шаблону документа. Такая настройка позволяет избежать ситуации, когда программа создает файл, неудобный для чтения или не соответствующий требованиям.

Таким образом, выбор библиотек `pandas` и `openruhl` обусловлен спецификой предметной области. Они обеспечивают удобную обработку структурированных данных, позволяют организовать обмен информацией в табличной форме и значительно расширяют практическую ценность создаваемой программы.

1.2.6 Настройка инструментов разработки

После того как были выбраны инструменты разработки необходимо выполнить их настройку для использования в проекте. На данном этапе задача заключается в формировании рабочего окружения, которое обеспечит удобную разработку, запуск, тестирование и ведение проекта.

Сначала была установлена программная платформа Anaconda [10], которая содержит в себе интерпретатор Python, основные средства для работы с библиотеками и менеджер пакетов. Официальный сайт, с которого можно установить платформу Anaconda представлен на рисунке 5.

Применение Anaconda в проекте позволяет облегчить и ускорить первичную подготовку среды разработки и уменьшить вероятность конфликтов между пакетами проекта.

Далее при помощи Anaconda Navigator была установлена среда разработки Spyder [11]. На рисунке 6 представлено окно приложения с доступными инструментами.

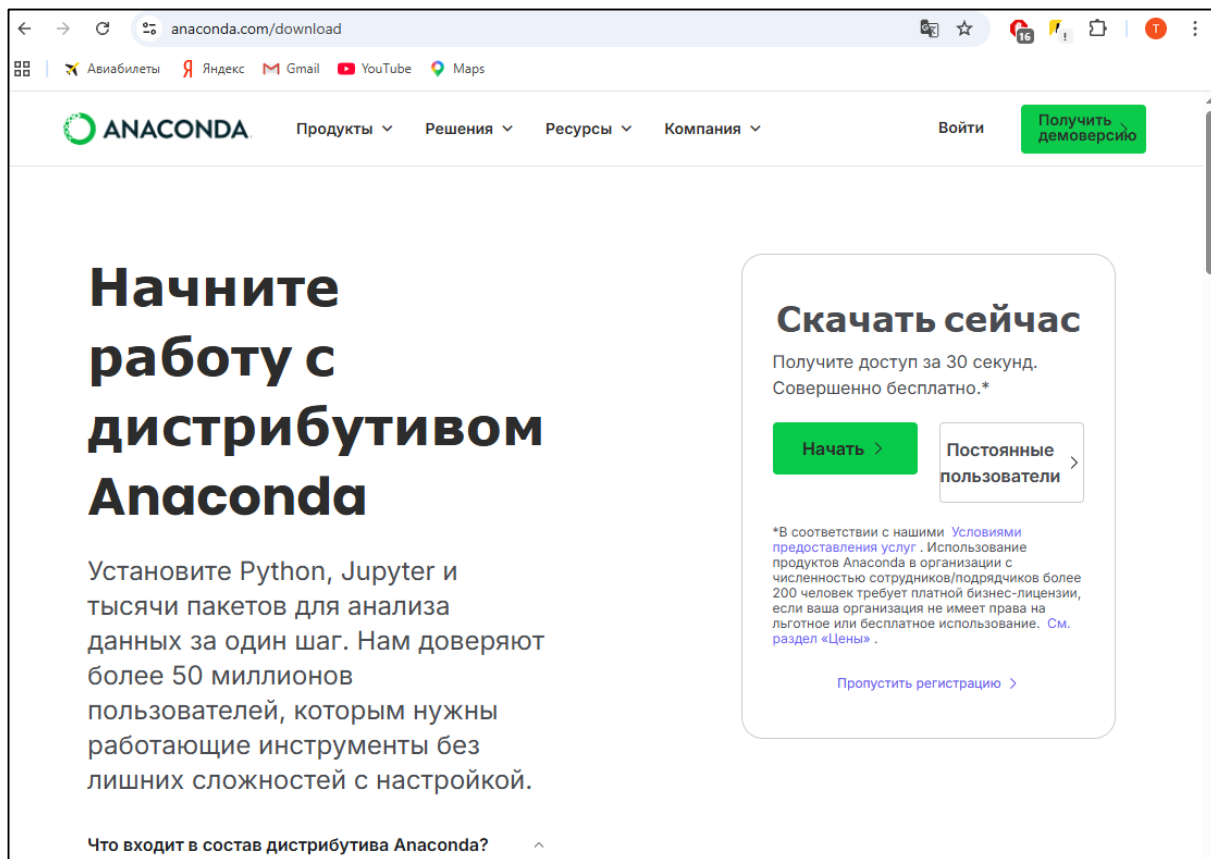


Рисунок 5 – Официальный сайт с дистрибутивом

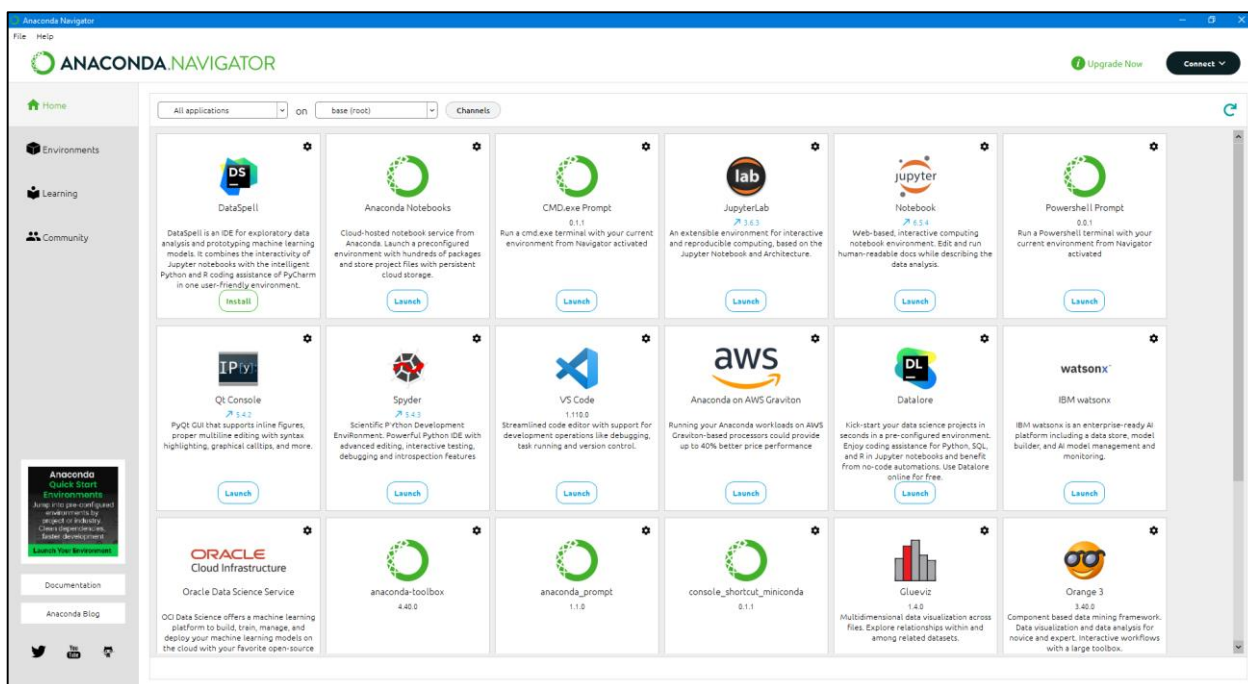


Рисунок 6 – Окно Anaconda Navigator с доступными инструментами разработки

					ПГУ 1.090301.27.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		30

После установки Anaconda было создано виртуальное окружение для системы, что продемонстрировано на рисунке 7. Такой подход дает возможность изолировать проект и используемые в нем библиотеки от других проектов, а также обеспечивает воспроизводимость среды разработки.

```

Anaconda Prompt
tzdata      pkgs/main/noarch::tzdata-2026a-he532380_0
ucrt        pkgs/main/win-64::ucrt-10.0.22621.0-haa95532_0
vc          pkgs/main/win-64::vc-14.3-h2df5915_10
vc14_runtime pkgs/main/win-64::vc14_runtime-14.44.35208-h4927774_10
vs2015_runtime pkgs/main/win-64::vs2015_runtime-14.44.35208-ha6b5a95_10
xz          pkgs/main/win-64::xz-5.8.2-h53af0af_0
zlib        pkgs/main/win-64::zlib-1.3.1-h02ab6af_0
zstd        pkgs/main/win-64::zstd-1.5.7-h56299aa_0

Proceed ([y]/n)? y

Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate schedule
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\Ant>
  
```

Рисунок 7 – Завершено создание виртуального окружения

Далее в созданном окружении были установлены библиотеки, которые потребуются для реализации функциональности системы, что можно увидеть на рисунке 8. В их числе Streamlit, pandas, openpyxl, sqlite3 [13, 16, 17, 14]. В результате было сформировано программное окружение, которое включает в себя необходимые инструменты для реализации и тестирования системы.

```

(schedule) C:\Users\Ant>conda install pandas openpyxl -y
Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(schedule) C:\Users\Ant>pip install streamlit
Successfully installed MarkupSafe-3.0.3 altair-6.1.0 attrs-26.1.0 blinker-1.9.0 cachetools-7.0.6 certifi-2026.2.25 chars
et_normalizer-3.4.7 click-8.3.2 colorama-0.4.6 gitdb-4.0.12 gitpython-3.1.46 idna-3.12 Jinja2-3.1.6 jsonschema-4.26.0 js
onschema-specifications-2025.9.1 narwhals-2.20.0 packaging-26.1 pillow-12.2.0 protobuf-7.34.1 pyarrow-24.0.0 pydeck-0.9.
2 referencing-0.37.0 requests-2.33.1 rpds-py-0.30.0 smmap-5.0.3 streamlit-1.56.0 tenacity-9.1.4 toml-0.10.2 tornado-6.5.
5 typing-extensions-4.15.0 urllib3-2.6.3 watchdog-6.0.0

(schedule) C:\Users\Ant>
  
```

Рисунок 8 – Успешная установка библиотек

Следующим шагом стала проверка работоспособности веб приложения на базе Streamlit. Был выполнен тестовый запуск приложения, в результате которого было подтверждено корректное отображение интерфейса в браузере, что позволило убедиться в работоспособности данного механизма. На рисунках 9 и 10 представлен результат тестирования.

```
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.18.0.1:8501
```

Рисунок 9 – Запуск фреймворка Streamlit

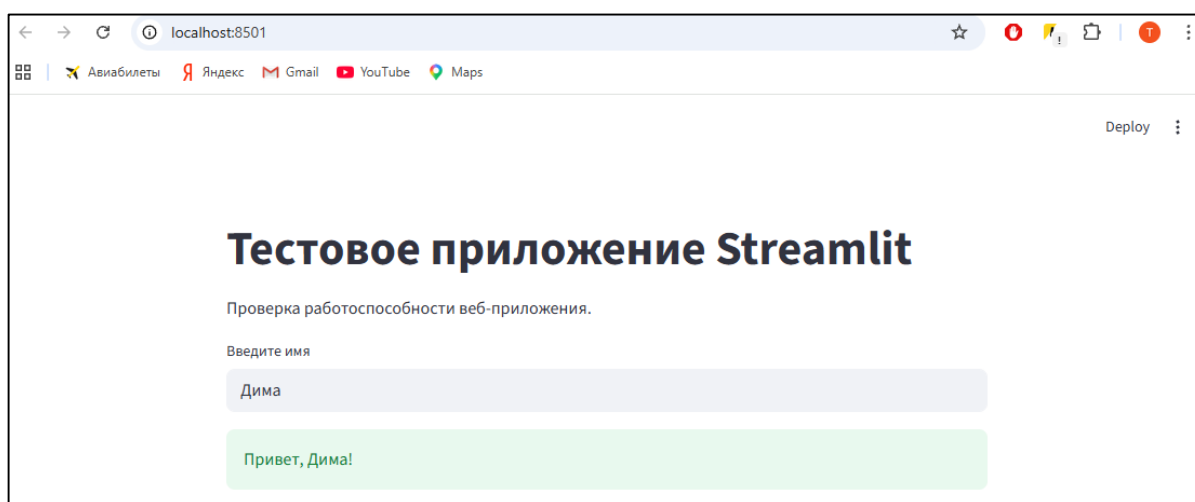


Рисунок 10 – Отображение интерфейса в браузере

Таким образом, в результате настройки инструментов разработки было сформировано рабочее окружение, позволяющее разрабатывать систему автоматизации составления расписания в удобной и стабильной среде.

1.3 Проектирование системы автоматизации

1.3.1 Архитектура приложения

Разрабатываемая система представляет собой приложение, пользовательский интерфейс которого отображается в браузере. Основным средством для создания интерфейса является фреймворк Streamlit [13], с помощью которого возможно в одном приложении объединить и вычислительную логику, и сам интерфейс. Такой подход очень эффективен, так как в данной СА приоритет отдается корректной

обработке входных данных и учету ограничений, а не созданию сложного интерфейса.

Архитектура основана на принципе модульной организации. Каждый такой блок отвечает за конкретный этап работы:

- ввод данных;
- проверка корректности;
- выбор ограничений;
- построение модели оптимизации;
- запуск алгоритма формирования расписания;
- представление результата;
- экспорт данных и хранение итоговых экземпляров.

Исходные данные поступают в систему в виде файла Excel. Для удобного взаимодействия с приложением, система предоставляет пользователю возможность загрузить шаблон входного файла с подсказками по заполнению. В шаблоне содержатся листы Groups, Activities и Rooms, предназначенные для записи учебных групп, занятий и аудиторий.

Когда заполненный шаблон загружен в систему, приложение его обрабатывает и проверяет. Проверяется структура файла, а именно наличие основных листов, корректность заполнения данных и соответствие данных между листами. Если во время проверки будут обнаружены какие-либо ошибки, система уведомляет об этом пользователя, что позволяет обнаружить и устранить проблему до запуска самого алгоритма.

Следующим элементом архитектуры является модуль настройки ограничений. С помощью него пользователь может задать определенные параметры, которые влияют на составление расписания:

- тип учебной смены;
- дни военной кафедры;
- индивидуальные предпочтения преподавателей.

С использованием этих параметров система строит множества подходящих вариантов расстановки занятий.

Основной вычислительный блок представлен модулем оптимизационного решения. В нем входные данные преобразуются во внутренние объекты, которые учитывают группы, дисциплины, аудитории и все остальные параметры. Затем для каждого занятия составляется набор возможных комбинаций временного промежутка и аудитории. Во время этого учитываются ограничения, которые заданы в предыдущем блоке. Для решения задачи используется открытая библиотека OR-Tools CP-SAT [18], благодаря которой возможно найти решение, удовлетворяющее наибольшему количеству ограничений.

Когда алгоритм завершается формируется итоговый вариант расписания. Затем он передается в модуль представления результата, где пользователь уже может посмотреть его в удобном формате. Также предусмотрено обнаружение ситуаций, в которых по тем или иным причинам расписание не может быть составлено. В таком случае пользователь видит набор возможных причин, мешающих формированию расписания.

Схематичное представление взаимодействия модулей программы представлено на UML-диаграмме компонентов в Приложении А.

1.3.2 Модель входных данных

Для того чтобы СА работала корректно и результат был пригоден к использованию, необходимо продумать структуру входных данных. В данной системе входные данные поступают в виде Excel-файла [16, 17]. Такой формат файла был выбран исходя из удобства и распространенности.

Применение Excel-файла дает возможность не разрабатывать сложную систему ввода данных через интерфейс приложения. Пользователь подготавливает нужную информацию, записывает в соответствии с шаблоном и загружает файл в систему. Это делает приложение удобным для применения и упрощает разработку.

Общая структура шаблона приведена в Таблице 2.

Таблица 2 — Структура входного Excel-файла

Лист	Назначение	Основные данные
Groups	Данные об учебных группах	Наименование группы, курс, численность, количество подгрупп

Лист	Назначение	Основные данные
Activities	Описание учебных занятий	Одна группа или поток, учебная дисциплина, тип занятия, преподаватель, частота проведения занятия, требования к оснащению аудитории
Rooms	Информация об аудиториях	Номер аудитории, наличие проектора, наличие компьютеров, возможность проведения лекций или лабораторных занятий, вместимость

В листе Groups содержится описание учебных групп, которые участвуют в формировании расписания. В нем записаны название группы, номер курса, численность и количество подгрупп. Название группы является ключевым идентификатором для системы, по которому происходит назначение дисциплин студентам.

Структура листа Groups представлена в таблице 3

Таблица 3 — Структура листа Groups

Поле	Назначение	Пример значения
group	Название учебной группы	22ВВП1
course	Номер курса	4
size	Количество студентов	29
lab_subgroups	Количество подгрупп	1
comment	Дополнительные заметки	Группа очного обучения

В листе Activities содержится основная информация о дисциплинах и конкретных занятиях, которые необходимо распределить по группам. Каждая строка в этой таблице — это отдельное занятие. В системе также предусмотрена возможность обработки диапазона групп, что дает возможность вносить в расписание потоковые занятия.

Структура листа Activities представлена в таблице 4

Таблица 4 — Атрибуты таблицы employees

Поле	Назначение	Пример значения
group_or_stream	Одна группа или поток	22ВВП1 или 22ВВП1-22ВВП2
discipline	Наименование дисциплины	Программирование

Поле	Назначение	Пример значения
kind	Тип занятия	ЛК или ЛБ
teacher	Преподаватель	Иванов И.И
freq	Частота проведения занятия	0.5 или 1.0 или 1.5 или 2.0
need_pc	Требуются ли компьютеры для занятия	0 или 1
need_projector	Требуется ли проектор	0 или 1
notes	Заметки	Потоковая лекция

Лист Rooms содержит информацию об аудиториях. Это необходимо, чтобы система могла расставлять занятия, ограничиваясь списком кабинетов, а также определять, какие помещения совместимы с определенными занятиями. К примеру, некоторые занятия требуют наличия компьютеров или проектора, при этом у каждой аудитории своя вместимость, поэтому для каждой аудитории рекомендуется указать ее характеристики. При этом, если не задана вместимость, то она считается достаточно вместимой при нахождении вариантов распределения.

Структура листа Rooms представлена в таблице 5.

Таблица 5 — Структура листа Rooms

Поле	Назначение	Пример значения
room	Номер или название аудитории	7а-322
has_projector	Наличие проектора	ДА или НЕТ
has_computer	Наличие компьютеров	ДА или НЕТ
lecture_possible	Возможность проведения лекций	ДА или НЕТ
lab_possible	Возможность проведения лабораторных занятий	ДА или НЕТ
capacity	Вместимость аудитории	50
notes	Заметки	Лекционная аудитория

На рисунке 12 отображена общая логика обработки входных данных

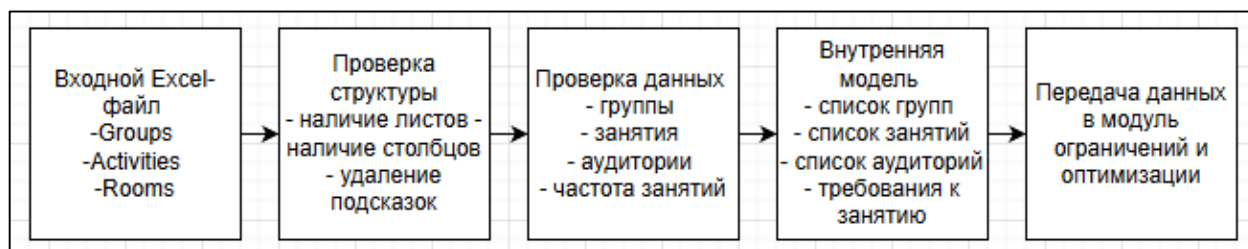


Рисунок 12 – Схема обработки входных данных

Как видно из рисунка, файл с исходными данными используется в системе не напрямую. Сначала он проверяется и система его преобразует во внутреннюю модель, для корректной работы.

Такой подход позволяет системе быть гибкой, ведь при изменении входных данных не придется менять код, а достаточно лишь обновить входной Excel-файл и заново загрузить его в программу.

1.3.3 Модель ограничений расписания

Когда информация загружена в систему, прежде чем приступить к формированию расписания нужно определить, какие варианты размещения будут считаться допустимыми, а какие нет. Для этого в приложении существует модель ограничений [1]. Она служит для описания правил, которые должны учитываться при распределении занятий. Особенность задачи составления расписания заключается в том, что любое занятие одновременно связано с множеством параметров, такие как: учебная группа, аудитория, тип занятия, преподаватель, временной интервал, день недели и сама неделя (четная или нечетная). Поэтому всегда при размещении приходится учитывать комплекс ограничений.

Основные ограничения, используемые при составлении расписания представлены в таблице 6.

Таблица 6 — Основные ограничения

Ограничение	Назначение
Одна учебная группа не может находиться одновременно на двух занятиях	Исключить пересечения у студентов
Один преподаватель не может одновременно проводить два занятия	Исключить пересечения у преподавателей

Ограничение	Назначение
В одной аудитории не могут проводиться два разных занятия одновременно	Исключить пересечения аудиторий
Аудитория должна соответствовать типу занятия	Распределение лекций и лабораторных занятий только в подходящие для этого помещения
Аудитория должна иметь необходимое оборудование	Обеспечить студентов и преподавателя необходимыми устройствами для проведения занятия
Вместимость аудитории должна соответствовать количеству групп	Исключить назначения занятия в неподходящую для этого аудиторию
В день военной подготовки не должно быть иных занятий	Учесть организационные ограничения
Временные промежутки, когда преподаватель не может проводить занятия, не могут участвовать в составлении расписания	Учесть индивидуальные ограничения преподавателя
Временные промежутки, когда преподаватель предпочитает провести занятия, имеют больший приоритет	Учесть индивидуальные предпочтения преподавателя
Максимальное количество пар в день ограничено	Предотвратить перегрузку студентов и преподавателей
Распределение занятий должно учитывать учебную смену	Эффективно распределить учебные занятия в течении дня
Желательно избегать неравномерной нагрузки	Повысить качество итогового расписания

Одним из самых важных ограничений является отсутствие пересечений у студентов. Если в определенный временной промежуток учебной группе уже назначено занятие, то другое занятие уже не может быть распределено на это время, так как студенты физически не могут присутствовать на двух разных занятиях. Аналогичное правило действует в отношении преподавателей.

Похожее ограничение связано с распределением аудиторий. Если в одну аудиторию уже назначено занятие одной группы, то другая группа не может занимать эту аудиторию, за исключением потоковых лекций.

Также действует ряд ограничений, связанных с техническим оснащением аудиторий. Каждая из них обладает характеристиками, которые были описаны в пункте 1.3.2, исходя из которых, если для занятия требуются компьютеры, то из доступных вариантов исключаются аудитории без компьютеров, если же требуется проектор, аналогично не учитываются аудитории без него.

Также для выбранных курсов или конкретных учебных групп могут быть назначены дни военной подготовки. При составлении расписания система исключит этот день из возможных вариантов и не будет назначать туда иные занятия.

Система также способна учитывать деление на учебные смены. Для первой смены алгоритм предпочитает более ранние временные промежутки, для второй, соответственно, более поздние. В случае невозможности распределения занятия с учетом смены, система в качестве исключения может назначить такое занятие в понедельник или субботу в то время, которое не соответствует учебной смене. Общая логика сортировки допустимых вариантов представлена на рисунке 13.

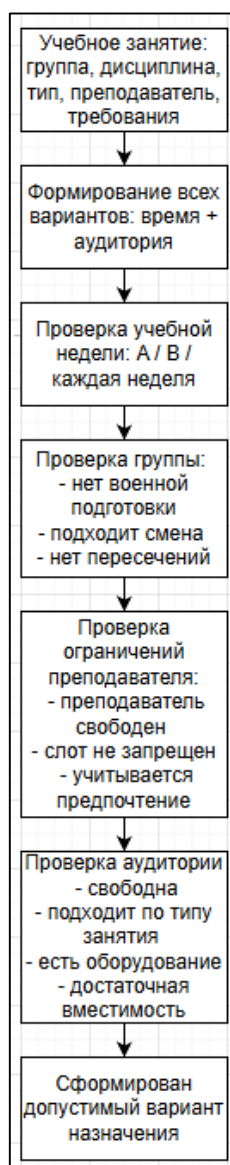


Рисунок 13 – Схема сортировки допустимых вариантов

Также для удобства реализована возможность учесть индивидуальные ограничения или предпочтения преподавателя. Для каждого временного промежутка есть три состояния:

- недоступно;
- предпочтительно;
- нейтрально.

Время, выбранное как «Недоступно» исключается из распределения. Промежуток отмеченный как «Предпочтительно» является приоритетным, но не обязательным. Статус «Нейтрально» не дает никаких особых свойств, система сама будет определять, задействовать этот временной промежуток или нет.

1.3.4 Математическая модель

Для работы алгоритма необходимо представить входные данные в формальном виде. Это позволяет избежать неточностей и двусмысленностей, а также построить оптимизационную модель, которую сможет обработать программа.

Задача формирования расписания заключается в распределении множества занятий по временным промежуткам так, чтобы не были нарушены ограничения и по возможности улучшены характеристики итогового расписания. В данной работе эта задача рассматривается как задача удовлетворения ограничений с элементами оптимизации [1, 18].

Пусть задано множество A учебных занятий:

$$A = \{a_1, a_2, \dots, a_n\}, \quad (1)$$

где: a_i – элемент множества соответствует одному учебному занятию.

У каждого элемента множества a_i есть следующие характеристики:

$$a_i = (G, D, K, T, E, W), \quad (2)$$

где G — набор учебных групп для этого занятия;

D — дисциплина;

K — тип занятия, например ЛК или ЛБ;

T — множество преподавателей, ведущих занятие;

E — требования к аудитории и оборудованию;

W — допустимые учебные недели проведения занятия.

Также в системе предусмотрено чередование учебных недель W :

$$W = \{A, B\}. \quad (3)$$

Это добавляет гибкости итоговому расписанию, так как дает возможность распределять занятие различными способами. Например, занятие можно проводить по обоим неделям, то есть и на неделе A и на неделе B , а можно на какой-то одной неделе, тогда на другой неделе этот временной промежуток будет свободен и туда можно разместить другое занятие.

Множество учебных групп обозначим так:

$$G = \{g1, g2, ..., gm\}. \quad (4)$$

Множество преподавателей:

$$T = \{t1, t2, ..., tp\}. \quad (5)$$

Множество аудиторий:

$$R = \{r1, r2, ..., rk\}. \quad (6)$$

Каждая аудитория обладает совокупностью характеристик:

$$rj = (Cj, Pj, Kj, Lj), \quad (7)$$

где Cj — вместимость;

Pj — наличие проектора;

Kj — наличие компьютеров;

Lj — подходящие типы занятий для данной аудитории.

Множество временных промежутков определяется как сочетание учебной недели, дня недели и времени занятия

$$S = W \times D \times H, \quad (8)$$

где W — множество недель;

D — множество учебных дней;

H — множество временных интервалов занятий.

Например, один временной интервал может быть представлен как:

$$S = (A, ПН, 09:50). \quad (9)$$

Такой набор означает, что занятие проводится на неделе А, в понедельник, в 9:50.

Для формирования расписания задействуется бинарная переменная:

$$x[i, s, r] \in \{0, 1\}, \quad (10)$$

где i — номер занятия;

s — временной интервал;

r — аудитория.

Переменная принимает значение:

$$x[i, s, r] = 1, \quad (11)$$

если занятие i распределено во временной интервал s и аудиторию r .

Иначе переменная принимает значение:

$$x[i, s, r] = 0. \quad (12)$$

Первое ограничение требует, чтобы занятие было назначено ровно один раз:

$$\sum x[i, s, r] = 1. \quad (13)$$

Суммирование происходит по всем подходящим временным интервалам s и аудиториям r для занятия i . Это ограничение нужно для того, чтобы занятие не было случайно пропущено или не назначено много раз, если оно уже существует как отдельный экземпляр во внутренней модели.

Далее идет ограничение, связанное с учебными группами. Одна группа не может находиться на двух занятиях в одно время, поэтому для каждой группы g и каждого временного интервала s должно выполняться условие:

$$\sum x[i, s, r] \leq 1. \quad (14)$$

Суммирование осуществляется по всем занятиям i , в которых задействована группа g , и по всем аудиториям r .

Похожее ограничение существует и в отношении преподавателей, в один момент времени преподаватель не может проводить 2 или более занятий:

$$\sum x[i, s, r] \leq 1 \quad (15)$$

Суммирование осуществляется по всем занятиям i , которые проводит преподаватель t , и по всем аудиториям r .

Также нужно учитывать, что аудитория может быть занята другой группой. Одна аудитория не может использоваться для проведения двух и более занятий одновременно:

$$\sum x[i, s, r] \leq 1. \quad (16)$$

Суммирование осуществляется по всем занятиям i , для конкретного временного интервала s , и по всем аудиториям r .

Существует также набор ограничений, связанный с соответствием аудитории занятию. Учебное занятие не может быть назначено в аудиторию, если в ней нет необходимого оборудования, например проектор и компьютеры, или если помещение не предназначено для проведения лекции или лабораторного занятия. Формально это можно представить так:

$$x[i, s, r] = 0, \text{ если } r \text{ не удовлетворяет требованиям } i. \quad (17)$$

Такие варианты исключаются еще до запуска решения.

Также в учет идет вместимость аудитории, если на занятии присутствуют несколько групп, то аудитория должна быть достаточно вместительной:

$$capacity(r) \geq groups_count(i), \quad (18)$$

где $capacity(r)$ — вместительность аудитории;

$groups_count(i)$ — количество групп на занятии.

Чтобы корректно учитывать ограничения, связанные с днем военной подготовки, введено правило, запрещающее размещать иные занятия в этот день. Если у группы g назначен день военной подготовки d , то для всех занятий i , где $g \in G_i$, выполняется условие:

$$x[i, s, r] = 0, \text{ если } day(s) = d. \quad (19)$$

Также выполняется контроль учебных смен. Если группе назначена определенная смена, то для назначения занятий доступны определенные временные интервалы:

$$x[i, s, r] = 0, \text{ если } slot(s) \text{ не соответствует смене группы.} \quad (20)$$

Ведется учет индивидуальных ограничений и предпочтений преподавателей. Каждый временной интервал может быть отмечен как:

недоступный, предпочтительный или нейтральный. Недоступные интервалы сразу исключаются:

$$x[i, s, r] = 0, \text{ если преподаватель занятия } i \text{ недоступен во время } s. \quad (21)$$

Нейтральные интервалы не имеют особых свойств, а предпочтительные являются частью целевой функции. Если алгоритму удастся разместить занятие в предпочтительный временной интервал, то такое размещение считается более приоритетным. Целевая функция служит инструментом подбора наиболее удобного итогового расписания, среди различных доступных вариантов. В данном случае ее можно представить, как максимизацию полезности:

$$\max F = P - Q, \quad (22)$$

где P — сумма баллов за выполнение предпочтений преподавателей и учет смены;

Q — штрафы за нежелательные факторы расписания.

К нежелательным факторам относятся:

- одиночные пары;
- неравномерная нагрузка по дням;
- распределение занятий в менее предпочтительные временные интервалы.

В упрощенном виде функцию можно записать так:

$$\max F = \sum preference_score[i, s, r] \cdot x[i, s, r] - \sum penalty[g, d], \quad (23)$$

где $preference_score[i, s, r]$ — оценка предпочтительности назначения занятия i во время s и аудиторию r ;

$penalty[g, d]$ — штраф за неудачное распределение нагрузки группы g в день d .

В программе эта модель передается в решатель OR-Tools CP-SAT [10]. Он принимает множество бинарных переменных, набор ограничений и целевую функцию. После этого начинается поиск такого распределения, в котором все ограничения выполнены, а значение целевой функции является наилучшим из найденных за отведенное время.

1.3.5 Проектирование хранения истории расписаний

В данной системе большая часть входной информации поступает в формате Excel-файла. Но после формирования готового расписания, может возникнуть

необходимость сохранить этот готовый результат, чтобы пользователь позже мог вернуться к нему, сравнить различные варианты, снова скачать какую-то более старую версию.

Для того чтобы решить эти задачи в системе предусмотрена возможность локального хранения истории составленных расписаний. В качестве инструмента хранения используется SQLite [14]. Для данной задачи эта БД хорошо подходит, так как не требует сервера, хранит все в одном файле и интегрируется прямо в код через встроенный модуль sqlite3.

В приложении используется файл базы данных schedule_history.sqlite3. Для защиты от перегрузки, максимальное количество вариантов расписаний, хранящихся в БД ограничено десятью вариантами.

Хранение готовых вариантов в БД не отменяет использование входного файла. При составлении расписания каждый раз используется исходный Excel-файл, а база данных хранит лишь уже готовые варианты расписания. Такой подход выбран, чтобы не нагружать систему сложной структурой базы данных и дать возможность пользователю удобно работать с расписанием.

Для хранения истории проектируется таблица schedule_history. В ней содержится информация о времени создания расписания, названии, количестве распределенных занятий, групп, курсов, а также данные о самом Excel-файле с готовым расписанием. Эта таблица создается автоматически, если не была создана ранее. Благодаря этому возможно использовать приложение, не настраивая БД вручную. Основные поля таблицы БД приведены в Таблице 7.

Таблица 7 — Структура таблицы schedule_history

Поле	Тип данных	Назначение
id	INTEGER PRIMARY KEY AUTOINCREMENT	Уникальный идентификатор записи
created_at	TEXT	Дата и время создания расписания
title	TEXT	Название расписания
assignments_count	INTEGER	Количество распределенных занятий

Поле	Тип данных	Назначение
groups_count	INTEGER	Количество учебных групп
courses_count	INTEGER	Количество курсов
xlsx_data	BLOB	Содержимое Excel-файла с расписанием

Поле id применяется как первичный ключ таблицы. С его помощью можно однозначно определить каждую запись в истории. Поле created_at хранит дату и время создания расписания, чтобы было пользователю было удобно ориентироваться среди версий. Поле title содержит название расписания. Поля assignments_count, groups_count и courses_count содержат краткую информацию о конкретном варианте расписания. Поле xlsx_data содержит Excel-файл в бинарном виде [14]. Это позволяет удобно хранить все варианты расписания в одном файле БД, что позволяет не создавать и не хранить папку с готовыми файлами Excel, а также минимизирует риски потери информации

Логика сохранения расписания выглядит следующим образом. После того как расписание успешно составлено, в приложении формируется итоговый Excel-файл. Далее он преобразуется в набор байтов и сохраняется в таблице schedule_history. Также в таблице сохраняются характеристики этого файла. Для этого используется функция сохранения. После добавления новой записи происходит очистка истории. Если количество расписаний превышает лимит, то система удалит самую старую запись, чтобы база данных не росла до бесконечности. Для этого используется функция, которая удаляет записи, кроме последних, отсортированных по дате создания.

На рисунке 14 представлена работа модуля хранения истории.



Рисунок 14 — последовательность работы модуля хранения истории

1.4 Разработка системы автоматизации

После того, как была спроектирована архитектура и все модели, была выполнена программная реализация СА составления расписания. Были разработаны функциональные компоненты системы: загрузка исходного файла, проверка корректности входных данных, формирование внутренней модели расписания, настройка ограничений, запуск алгоритма, предоставление результата, экспорт итогового расписания и сохранение истории.

Разработка была выполнена на языке Python с использованием фреймворка Streamlit, что позволило создать приложение с веб-интерфейсом, открываемым в браузере. Основной файл содержит всю структуру программы – и функции и интерфейс.

В процессе разработки особое внимание уделялось как самому алгоритму составления расписания, так и подготовке входных данных. Этот фактор очень важен, так как качество итогового расписания напрямую зависит от качества исходных данных. Поэтому все данные проходят через несколько уровней проверки.

1.4.1 Реализация загрузки и проверки входных данных

Работа с приложением начинается с загрузки входных данных. Пользователь загружает в систему Excel-файл, в котором содержится информация об учебных группах, занятиях и аудиториях. Загрузка происходит через боковую панель интерфейса. В программной реализации используется компонент `file_uploader` [13], который принимает файлы `.xlsx` формата.

После этого содержимое файла передается в функцию проверки данных. Для этого разработана функция `read_dynamic`, которая принимает байтовое содержимое Excel-файла, открывая его с помощью библиотеки `pandas` [16] и проверяет наличие обязательных листов. Если хоть один из них отсутствует, то выводится сообщение об ошибке.

После того как эта проверка прошла успешно, начинается чтение таблиц, в таблице 8 показано соответствие листа входного файла и переменных в коде программы.

Таблица 8 – Соответствие листов и переменных

Название листа	Переменная в программе
Groups	groups_df
Activities	acts_df
Rooms	rooms_df

Из этих таблиц формируются внутренние структуры данных, которые в дальнейшем используются для формирования расписания. Также дополнительно реализована обработка строк-подсказок, которые содержатся в шаблоне исходного файла. Они нужны для того, чтобы пользователю было проще заполнить шаблон без ошибок, но не должны учитываться алгоритмом при распределении. Для этого в коде есть определенные маркеры, такие как: пример, example, __example__, строка-подсказка. Если значение в столбце соответствует одному из этих маркеров или начинается со слова «пример», то такая строка удаляется из таблицы перед обработкой. Этапы проверки представлены в таблице 9

Таблица 9 – Этапы проверки

Этап проверки	Реализация	Результат
Проверка формата файла	Загрузка только .xlsx файлов	Исключается загрузка неподходящих файлов
Проверка обязательных листов	Проверяются Groups, Activities, Rooms	Ошибка при отсутствии нужного листа
Удаление строк-подсказок	Функция удаления служебных строк	В алгоритм поступают только нужные данные
Проверка групп	Сопоставление групп из Activities с листом Groups	Ошибка при неизвестной группе
Проверка аудиторий	Преобразование листа Rooms в список объектов Room	Ошибка при отсутствии аудиторий

Важный момент проверки – это сопоставление групп, которые указаны в листе Groups с теми, что записаны в Activities. Для этого реализована функция `validate_activity_groups`. На основе листа Groups формируется список учебных групп. Далее каждая строка в листе Activities проверяется: если обнаруживается такая группа, которой нет в составленном списке, то формируется сообщение об ошибке. В сообщении указывается строка в файле и запись, которую нужно отредактировать. Это позволяет обнаружить ошибки на раннем этапе и не запускать алгоритм с неверными данными.

Для обработки потоков групп реализована функция, которая разбирает строки. Она поддерживает разные варианты записей, например: 25BBB1, 25BBB1-4, 253ВЭЭ51-253ВЭЭ52. При этом ошибочные варианты записи не преобразуются и не корректируются автоматически во избежание неверных трактовок. Запись просто остается в изначальном значении, для того, чтобы предотвратить ошибки.

Также на этапе загрузки происходит преобразование информации об аудиториях. Для каждой из них формируется объект Room, содержащий всю информацию об аудитории. Такое преобразование нужно для корректной работы решателя. Модуль оптимизации не работает со строками из Excel-файла. Для его работы нужны структурированные объекты, с четко заданными характеристиками. Поэтому данные из листа Rooms переводятся во внутренний формат, который выглядит следующим образом:

```
Room (
    room = "7a-316",
    has_pc = False,
    has_projector = True,
    lecture_possible = True,
    lab_possible = False,
    capacity_groups = 2
)
```

Полный листинг программы находится в Приложении Б.

После того, как данные были успешно считаны и проверены, система показывает краткую информацию об общем количестве групп, занятий и

аудиторий. Это позволяет пользователю сразу увидеть, что файл прочитан верно и в систему поступил должный объем данных.

1.4.2 Создание шаблона входного Excel-файла

Шаблон входного файла формируется в системе с применением библиотеки `openpyxl` [17], которая позволяет создавать Excel-книгу, добавлять листы, заполнять ячейки и делать оформление. Благодаря этому не нужно хранить готовый файл с шаблоном и предоставлять его пользователю, вместо этого система сама создает его при нажатии на кнопку загрузки шаблона. Структура самого шаблона уже была описана в пункте 1.3.2

С технической точки зрения, создание шаблона происходит с помощью функции `build_input_template_workbook()`. Внутри нее создается книга Excel с помощью объекта `Workboook` из библиотеки `openpyxl`. Внутри книги создаются листы: `Groups`, `Activities` и `Rooms`, а также столбцы в этих листах. Затем функция добавляет во вторую строку каждого листа пример того, как заполнять шаблон.

Для стилизации листов используется отдельная функция `style_template_sheet(ws)`, где `ws` – текущий лист. Эта функция задает оформление заголовков и строки с подсказками: выделяет первую строку, выделяет заголовки полужирным шрифтом, устанавливает границы ячеек и ширину столбцов, фиксирует верхнюю часть листа. С помощью этого шаблон становится более удобным к восприятию и вероятность ошибок снижается.

После заполнения шаблона он сохраняется во временный объект `BytesIO` и сразу передается в интерфейс `Streamlit`, чтобы пользователь мог его скачать.

Для обработки строки, которая является примером и не должна быть распределена, предусмотрены функции `is_template_hint_value()` и `drop_template_hint_rows()`. Функция `is_template_hint_value()` определяет, является ли строка подсказкой, например если она содержит маркеры: пример, `example`, `__example__` или строка-подсказка. Функция `drop_template_hint_rows()` удаляет такие строки из таблиц `Groups`, `Activities` и `Rooms`.

1.4.3 Разработка настройки дополнительных параметров

После того, как пользователь загрузил данные, он может перейти к настройке дополнительных параметров расписания. К ним относятся: дни военной подготовки, учебные смены, предпочтения преподавателей и время на поиск решения.

Настройка данных параметров реализована через интерфейс Streamlit и не изменяет входной файл, что позволяет пользователю используя один входной файл гибко настраивать параметры и получать различные варианты.

Параметры, задаваемые пользователем, представлены в таблице 10

Таблица 10 – Основные группы параметров

Группа параметров	Назначение	Где используется
Время поиска решения	Ограничивает длительность работы решателя	При запуске CP-SAT
Смены обучения	Определяют допустимые временные интервалы для групп	При формировании допустимых назначений
Дни военной подготовки	Исключают определенный день недели для выбранных групп	При фильтрации временных интервалов
Предпочтения преподавателей	Задают предпочтительные и недоступные интервалы	При построении ограничений и целевой функции

Время поиска решения передается в модуль оптимизации и ограничивает работу решателя [18]. Этот параметр очень важен, так как задача составления расписания является комбинаторной задачей и при большом количестве входных данных поиск лучшего решения может занимать очень много времени. Поэтому ограничение времени дает возможность сохранить управляемость работой приложения: пользователь сам решает, сколько времени можно выделить на поиск решения.

В программе время, отведенное на поиск решения, передается в функцию `solve_schedule()` через параметр `time_limit_sec`. Затем оно используется при настройке объекта `CpSolver`, где задается максимальное время решения.

Следующей группой параметров является настройка смен обучения. В программе для этого формируется структура `group_shift_map`, где каждой группе сопоставляется выбранный вариант смены. При составлении подходящих вариантов назначения занятия используется функция проверки смены. Если временной интервал не подходит группе по смене, такой вариант исключается из дальнейшего рассмотрения. Это позволяет не добавлять в модель заведомо неподходящие назначения.

Для настройки дней военной подготовки в коде предусмотрена структура `military_days_by_group`. Для групп, у которых задана такая настройка, иные занятия на ставятся в этот день.

При настройке предпочтений преподавателя используются три состояния временных интервалов: нейтрально, предпочтительно, недоступно. В коде эти значения представлены константами `PREF_NEUTRAL`, `PREF_PREFERRED`, `PREF_UNAVAILABLE` и объединены в список `PREF_OPTIONS`. Для преобразования пользовательского значения во внутренний код используются функции `pref_to_code()` и `code_to_pref()`. Например, предпочтительный слот преобразуется во внутреннее значение `preferred`, а недоступный — в `unavailable`. Эти данные записываются в структуру `teacher_pref_map`. Она связывает преподавателя с набором временных интервалов и их состояниями. Далее эта структура используется двумя способами.

Во-первых, если временной интервал отмечен как недоступный, система отбрасывает его из допустимых вариантов, такая проверка производится функцией `slot_is_teacher_unavailable`.

Во-вторых, если временной интервал отмечен как предпочтительный, и система может туда поставить занятие, то такой вариант получает дополнительный балл и становится более приоритетным, такая проверка производится функцией `slot_preference_score()`.

Для подготовки списка дней и времени используется функция `parse_days_and_times()`. Она позволяет получить итоговые списки дней и временных интервалов на основе настроек. Далее функция

effective_times_for_shifts() формирует итоговый набор временных слотов с учетом смен. Это необходимо для того, чтобы все используемые временные интервалы были доступны при построении расписания.

В таблице 11 представлены структуры, которые система получает для запуска алгоритма.

Таблица 11 – Основные структуры для работы алгоритма

Название структуры	Что хранит
days	Список учебных дней
effective_times	Список временных интервалов
group_shift_map	Смены групп
military_days_by_group	Дни военной подготовки
teacher_pref_map	Предпочтения преподавателей
time_limit_sec	Лимит времени решателя

Эти структуры используются при формировании допустимых вариантов в функции build_solver_artifacts() и при запуске решателя в функции solve_schedule().

1.4.4 Реализация алгоритма составления расписания

Основная часть системы – это алгоритм автоматического формирования расписания. При нажатии кнопки «Составить расписание» происходит его запуск. В коде за это отвечает последовательный запуск функций expand_activities_to_occurrences() и solve_schedule(). Первая функция формирует список экземпляров занятий, а вторая выполняет построение и решение оптимизационной модели.

Первым делом происходит преобразование строк из листа Activities во внутренние объекты с помощью функции expand_activities_to_occurrences(). Она обрабатывает информацию из каждой строки и основываясь на частоте занятия создает отдельные экземпляры для недель А и В. Это нужно, потому что одна строка в листе Activities не всегда соответствует одному назначению в расписании, ведь занятие может проводиться по одной неделе, по двум, а также чаще одного раза в неделю. Поэтому исходные данные разворачиваются в список внутренних объектов Occurrence.

В таблице 12 приведена структура объекта Occurrence, который содержит основную информацию для дальнейшего распределения.

Таблица 12 – Описание структуры объекта Occurrence.

Поле	Назначение
occ_id	Идентификатор экземпляра занятия
base_id	Идентификатор исходного занятия
groups	Список учебных групп, участвующих в занятии
discipline	Наименование дисциплины
kind	Тип занятия, например лекция или лабораторное занятие
teachers	Список преподавателей, ведущих занятие
need_pc	Признак необходимости компьютеров
need_projector	Признак необходимости проектора
fixed_room	Фиксированная аудитория, если она указана во входных данных
allowed_weeks	Список допустимых недель для проведения занятия

Далее выполняются подготовка данных для решателя с помощью функции `build_solver_artifacts()`. Она формирует список всех подходящих временных интервалов. В случае же, если не найдено ни одного допустимого варианта, приложение формирует ошибку `NoFeasibleOptions`.

После того, как список подходящих вариантов составлен, создается модель оптимизации. В функции `solve_schedule()` используется объект `cp_model.CpModel()` из библиотеки `OR-Tools` [18]. Помимо учета ограничений, описанных в пункте 1.3.3, важным фактором является отсутствие «окон» в расписании и учет дневной нагрузки [1, 4]. Для этого используется функция `contiguous_patterns()`, которая формирует варианты непрерывной занятости без разрывов в расписании. Дополнительно функция `daily_load_penalty_values()` задает «штрафы» за неудачное распределение нагрузки, например за одиночную пару или перегрузку расписания.

После создания переменных, ограничений и целевой функции запускается решатель `CpSolver`. В коде для него задается максимальное время поиска через параметр `max_time_in_seconds`, а также используется несколько рабочих потоков поиска. Если решатель находит оптимальное или допустимое решение, поиск

продолжается. Если решение не обнаружено, вызывается функция `build_infeasibility_diagnostics()`, а пользователю выводится сообщение о противоречии ограничений.

При успешном решении задачи вызывается функция `solve_schedule()`, которая проходит по всем занятиям и определяет, какой вариант был выбран решателем и затем строит для каждого выбранного назначения список с основными данными, которые представлены в таблице 13.

Таблица 13 – Параметры списка

Поле	Назначение
<code>occ_id</code>	Идентификатор экземпляра занятия
<code>base_id</code>	Идентификатор исходного занятия
<code>week</code>	Учебная неделя, на которую назначено занятие
<code>day</code>	День недели, выбранный для занятия
<code>time</code>	Время начала занятия
<code>discipline</code>	Наименование дисциплины
<code>kind</code>	Тип занятия, например лекция или лабораторное занятие
<code>teachers</code>	Преподаватель или список преподавателей, ведущих занятие
<code>room</code>	Аудитория, назначенная для проведения занятия
<code>groups</code>	Учебная группа или список групп на занятии
<code>text</code>	Текстовое представление занятия для отображения в расписании

Этот список необходим для работы интерфейса, на его основе строится таблица расписания, происходит демонстрация расписания пользователю, экспорт в Excel и сохранение в историю. После получения результата список назначений сохраняется в `st.session_state.schedule_assignments`, а контекст расписания — в `st.session_state.schedule_context`. Затем приложение пытается сохранить результат в БД SQLite с помощью функции `save_schedule_to_history()`.

В системе предусмотрено два варианта обработки ошибок составления расписания. В первом случае, если не нашлось ни одного варианта расстановки, то возникает исключение `NoFeasibleOptions`, и пользователь получает информацию по этому занятию. Если же некоторые варианты все-таки есть, но модель получается противоречивой, то тогда возникает другое исключение `NoScheduleFound`. Тогда

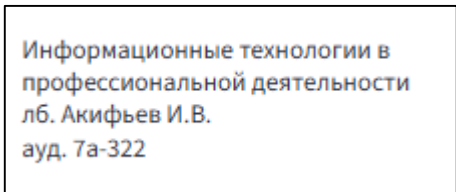
система выдает информацию о самых вероятных проблемах: перегруженные группы, преподаватели, самые ограниченные занятия и результаты проверки ослабления отдельных ограничений.

1.4.5 Разработка способа отображения и редактирования результата

После того, как алгоритм завершил свою работу, система получает список назначенных занятий. Для хранения результата используется механизм `st.session_state`. После того, как выполнится функция `solve_schedule()`, список назначений записывается в переменную `st.session_state.schedule_assignments`, а дополнительные сведения о расписании сохраняются в `st.session_state.schedule_context`. Это позволяет не терять данные при обновлении отдельных частей интерфейса и использовать одни и те же данные в разных частях системы.

Затем пользователь имеет возможность просмотреть составленное расписание в виде таблицы. В ячейках собрана основная информация о конкретном занятии, а именно название дисциплины, преподаватель и аудитория. Для формирования ячеек используется функция `format_assignment_text()`, которая объединяет информацию о занятии в удобный для отображения формат.

Пример содержимого ячейки представлен на рисунке 15.



Информационные технологии в
профессиональной деятельности
лб. Акифьев И.В.
ауд. 7а-322

Рисунок 15 – Одна из ячеек итогового расписания

Также в приложении выводится информация об общем количестве назначений. Благодаря этому можно сориентироваться, все ли занятия были распределены. Если все прошло успешно, система также выведет сообщение о том, что распределение успешно завершено.

Для удобства работы с расписанием его можно не только просмотреть, но и скачать. Так как данные хранятся структурированно, это дает возможность

использовать список назначений и для вывода в интерфейс и для экспорта в Excel-файл.

Если случается необходимость отредактировать исходное расписание, пользователь может прибегнуть к двум вариантам. Первый это заново сформировать расписания, изменив входные данные или ограничения. Вторым это вручную перенести занятие, выбрав его из списка и подобрать другой подходящий временной интервал из предложенных.

1.4.6 Реализация экспорта расписания

Экспорт выполняется на основе списка назначенных занятий, полученного после работы алгоритма оптимизации. Данный список хранится в `st.session_state.schedule_assignments`. При экспорте он преобразуется в табличный вид и записывается в книгу Excel. При формировании файла для экспорта используется библиотека `openpyxl` [17]. Сам экспорт выполняется функцией, которая собирает Excel-файл и возвращает его в виде объекта `BytesIO`.

При формировании ячеек, применяется тот же способ, что и при работе с интерфейсом. Также если занятие является потоковым, оно отображается в расписании каждой из групп, что позволяет пользователю видеть полноценное расписание каждой отдельно взятой группы.

В системе также реализована функция `safe_sheet_name()`, которая приводит название листов к допустимому виду, так как в Excel есть ограничения на длину имени, а также запрещены некоторые символы.

Экспорт связан и с модулем хранения истории. При сохранении расписания в историю используется тот же сформированный Excel-файл, который может быть скачан пользователем. Это позволяет хранить в SQLite не только данные о расписании, но и сам готовый файл.

1.4.7 Реализация хранения истории расписаний

Чтобы с приложением было удобнее взаимодействовать была разработана возможность автоматического хранения последних составленных расписаний в БД SQLite [14]. В коде программы путь к файлу базы данных задается константой `SCHEDULE_HISTORY_DB`, а количество сохраняемых вариантов ограничивается

константой SCHEDULE_HISTORY_LIMIT. В данной системе база данных хранит последние 10 составленных расписаний.

Для работы с историей расписаний разработаны функции, которые представлены в таблице 14.

Таблица 14 — Основные функции модуля истории расписаний

Функция	Назначение
init_schedule_history_db()	Создает SQLite-базу и таблицу schedule_history, если они еще не созданы
save_schedule_to_history()	Сохраняет сформированное расписание в историю
prune_schedule_history()	Удаляет старые записи и оставляет только последние
list_schedule_history()	Получает список сохраненных расписаний из базы данных
render_schedule_history_sidebar()	Отображает историю в боковой панели интерфейса

1.5 Тестирование и отладка

После реализации основных компонентов приложения было выполнено тестирование системы. Целью тестирования являлась проверка корректности работы основных функций приложения.

Так как разрабатываемая система является прикладным программным решением с пользовательским интерфейсом, основное внимание было уделено функциональному тестированию [18]. Проверялись действия, которые пользователь выполняет при обычной работе с приложением, а также ситуации, в которых были допущены ошибки.

Основные сценарии тестирования приведены в таблице 15

Таблица 15 — Тестовые сценарии проверки приложения

Проверяемая функция	Входные данные / действие	Ожидаемый результат	Фактический результат
Скачивание шаблона	Нажать кнопку «Скачать шаблон Excel»	Пользователь получает файл template_semester.xlsx с листами Groups, Activities, Rooms	Шаблон скачивается корректно
Загрузка корректного файла	Загрузить заполненный .xlsx	Приложение считывает данные и выводит короткую	Файл обработан корректно

Проверяемая функция	Входные данные / действие	Ожидаемый результат	Фактический результат
	файл с обязательными листами	сводку по группам, занятиям и аудиториям	
Проверка отсутствующего листа	Загрузить файл без листа Rooms	Приложение должно вывести сообщение об ошибке о нехватке обязательного листа	Ошибка отображается
Удаление строки-подсказки	Оставить строку-пример в шаблоне	Строка-подсказка не должна попасть в алгоритм распределения	Строка удаляется при обработке
Проверка неизвестной группы	Указать в Activities группу, которой нет в Groups	Приложение должно сообщить, какая группа не найдена	Ошибка определяется
Проверка диапазона групп	Указать поток, например 25BBB1-25BBB4	Приложение должно корректно разобрать группы потока	Диапазон обрабатывается
Проверка аудитории без оборудования	Для занятия с need_pc = 1 оставить только аудитории без компьютеров	Приложение должно определить отсутствие допустимых аудиторий	Выводится диагностическое сообщение
Проверка недоступности преподавателя	Отметить все подходящие слоты преподавателя как недоступные	Система не должна назначать занятия в запрещенные слоты	Запрещенные слоты исключаются
Запуск генерации расписания	Нажать кнопку «Составить расписание» при корректных данных	Запускается алгоритм, формируется расписание	Расписание формируется
Проверка невозможного расписания	Задать противоречивые ограничения	Приложение должно вывести диагностику причины проблемы	Отображается информация о конфликте
Просмотр результата	Нажать кнопку «Просмотреть / редактировать расписание»	Открывается модальное окно просмотра расписания	Расписание отображается
Экспорт результата	Нажать кнопку скачивания итогового расписания	Пользователь получает Excel-файл с расписанием	Файл скачивается
Сохранение истории	Сформировать расписание	Расписание сохраняется в SQLite в бинарном виде	Запись появляется в истории
Скачивание из истории	Нажать «Скачать из истории»	Пользователь получает ранее сформированный Excel-файл	Файл скачивается из истории

Во время тестов особое внимание было уделено обработке ошибок во входных данных, так как пользователь случайно может неверно заполнить файл, удалить

нужный лист, ошибиться в названии группы и тд. В таких случаях приложение не завершается аварийно, а верно обрабатывает ошибку.

Например, если в файле нет одного из обязательных листов, функция `read_dynamic()` прекращает обработку и формирует сообщение о нехватке листов. Если в занятии указана группа, отсутствующая на листе `Groups`, функция `validate_activity_groups()` формирует список ошибочных строк и указывает, какие группы не найдены. Такой подход упрощает исправление входного файла пользователем.

Для проверки корректности экспорта было сформировано расписание и скачан готовый файл-Excel. Было проверено, что файл успешно открывается, в нем присутствует необходимая информация, и она пригодна для использования.

В результате тестирования было установлено, что приложение корректно выполняет основные пользовательские сценарии. Система позволяет скачать шаблон, загрузить входной Excel-файл, проверить его структуру, настроить параметры расписания, запустить алгоритм составления расписания, просмотреть результат, экспортировать его в Excel и скачать ранее сохраненные варианты из истории.

Также было выявлено, что система корректно обрабатывает ситуации с различными ошибками, что повышает надежность системы и делает ее более удобной в эксплуатации.

					ПГУ 1.090301.27.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

2 Безопасность жизнедеятельности

Разработанная СА является программным решением, взаимодействие с которым происходит с помощью компьютера или ноутбука. Пользователь работает с приложением через веб-интерфейс. Основная работа пользователя связана со взаимодействием с Excel-таблицами, настройкой дополнительных параметров, просмотром и редактированием расписания.

При работе с такой системой отсутствуют многие опасные факторы, такие как: работа с механическим оборудованием, вредными веществами, физически тяжелый труд, повышенная или пониженная температура, высокий уровень шума и тд. Но при этом работа за компьютером [19], а в частности с данной системой, несет за собой ряд других вредных факторов:

- нагрузка на мышцы шеи, верхнего плечевого пояса и спины;
- зрительная нагрузка [19];
- усталость кистей рук;
- чрезмерное или недостаточное освещение;
- неправильная организация рабочего места;
- риск, связанный с использованием электрического оборудования.

В таблице 16 приведены основные опасные факторы и меры предосторожности для их минимизации.

Таблица 16 – Опасные факторы при работе с системой

Фактор	Возможное последствие	Меры снижения риска
Длительная работа за экраном	Утомление глаз, снижение концентрации	Перерывы, правильная яркость экрана, достаточное освещение
Неправильная посадка	Боль в спине, шее, плечах	Регулировка положения на стуле, высоты стола и расположения монитора
Длительная работа с мышью и клавиатурой	Усталость кистей рук	Удобное расположение устройств ввода, кратковременные разминки

Фактор	Возможное последствие	Меры снижения риска
Недостаточное или чрезмерное освещение	Повышенная зрительная нагрузка	Использование равномерного искусственного или естественного освещения
Электрический ток	Риск поражения электрическим током или возгорания	Использование исправной техники без видимых повреждений и качественных сетевых фильтров

Чтобы снизить зрительную нагрузку, экран монитора должен находиться примерно на расстоянии вытянутой руки. Лучше расположить монитор на уровне глаз или чуть ниже. Это позволяет также снизить нагрузку на шею при длительной работе.

Важную роль играет равномерное освещение. Монитор стоит разместить таким образом, чтобы на него не попадали прямые солнечные лучи. Также не стоит работать в помещении без освещения, так как это сильно увеличивает зрительную нагрузку.

Также при длительной работе с таблицами периодически стоит делать короткие перерывы. Нужно отвлечься от экрана, переключить внимание на какой-либо объект вдаль, размять кисти рук, шею и спину.

Важно следить за своим рабочим местом. У стула или рабочего кресла должна быть поддержка спины, а также верно отрегулирована высота, чтобы руки находились в естественном положении, а ноги были в устойчивом положении. Мышь и клавиатура должны быть расположены таким образом, чтобы при работе с ними не напрягались запястья, руки и плечи. Несоблюдение этих простых правил при длительной работе приводит к дополнительной утомляемости и снижению работоспособности.

Учитывая меры электробезопасности, перед началом работы нужно убедиться, что все оборудование исправно, включая розетки, сетевые фильтры, кабели питания и тд. Недопустимо использовать поврежденные провода, блоки питания, розетки. Подключение компьютера лучше производить с использованием сетевого фильтра или с источником бесперебойного напряжения.

Рядом с любым из перечисленных электрических устройств запрещено размещать любые жидкости, которые при попадании на приборы может привести к короткому замыканию, поломке техники или даже к поражению электрическим током.

Также не стоит подключать много устройств к одному удлинителю или же оставлять включенные устройства без присмотра. Помещения, в которых есть компьютеры, должны быть оборудованы средствами пожаротушения [20], а также должен быть обеспечен свободный доступ к ним. Пользователь должен быть ознакомлен с тем, как необходимо поступать при появлении запаха гари, задымлении или возгорании электрических приборов. При появлении любых признаков неисправности или поломки оборудования, нужно немедленно остановить работу и отключить устройство от сети.

Таким образом, при соблюдении этих правил, можно минимизировать опасные факторы и сделать взаимодействие с системой более удобным и продуктивным.

Заключение

В ходе выполнения данной работы была реализована СА составления учебного расписания с использованием методов оптимизации и учетом ресурсных ограничений. Ее основное назначение – это снижение трудоемкости задачи формирования расписания, повышение удобства при работе с ним, а также уменьшение количества ошибок за счет комплекса проверок.

В первой части работы был проведен анализ предметной области, рассмотрены основные нюансы задачи, выявлены основные элементы, а именно: учебные группы, дисциплины, преподаватели, временные интервалы, аудитории, дни и недели, а также дополнительные ограничения. Было определено, что задача составления расписания является трудоемкой комбинаторной задачей [1], которая требует одновременного учета множества факторов.

Также был проведен анализ существующих аналогов, после чего было выявлено, что многие системы, хоть и обладают своими плюсами, но при этом часто зависят от коммерческой лицензии, бывают излишне функционально нагруженными или сложными во внедрении [2, 3, 5]. В связи с этим был сделан вывод, что разработка данной системы является актуальной задачей.

Для реализации системы был выбран язык программирования Python, фреймворк Streamlit, библиотеки OR-Tools CP-SAT, pandas и openpyxl, база данных SQLite [9, 13, 14, 16, 17, 18].

В части проектирования системы была продумана архитектура, которая включает в себя несколько модулей: модуль пользовательского интерфейса, чтения и проверки данных, выбора и настройки ограничений, формирования модели оптимизации, составления расписания, отображения результата, экспорта и хранения истории.

В процессе реализации были созданы различные функции и механизмы, для корректной работы с файлами и обработки ошибок. Сам алгоритм формирования расписания основан на библиотеке OR-Tools CP-SAT, который в свою очередь

строит набор ограничений, составляет подходящие варианты распределений и ищет наиболее подходящее итоговое распределение.

После того как расписание успешно составлено, система дает возможность его просмотреть, отредактировать или скачать. Также реализовано хранение последних десяти вариантов расписания в базе данных SQLite, что дает возможность скачать более старые версии и снижает вероятность потери данных.

В части тестирования были сделаны различные проверки пользовательских сценариев и ошибочных ситуаций. Результаты тестирования показали, что система корректно обрабатывает и выполняет свои задачи, а в случае обнаружения ошибок выдает соответствующие сообщения.

В пункте безопасности жизнедеятельности были рассмотрены факторы, сопутствующие работе с данной системой. Изучены способы минимизации рисков, такие как: верная организация рабочего места, соблюдение правил работы за компьютером, понимание того, как действовать в аварийных ситуациях, соблюдение правил пожарной безопасности и электробезопасности. При выполнении этих правил возможно минимизировать вредные факторы при работе с приложением.

Проанализировав текущий функционал и возможности системы, был сделан вывод, что в дальнейшем приложение можно улучшить следующим образом:

- масштабировать систему, добавив поддержку полноценной клиент-серверной архитектуры;
- добавление поддержки ролей пользователей (администратор, пользователь, зритель);
- расширение списка ограничений;
- автоматическое составление дополнительных отчетов.

Таким образом, в результате выполнения выпускной квалификационной работы, была достигнута цель: разработана система автоматизации составления учебного расписания с использованием методов оптимизации и учетом ресурсных ограничений. В ходе выполнения были решены все поставленные задачи.

Список использованных источников

1. Bashab A., Ibrahim A. O., AbedElgabar E. E., et al. Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, and Open Issues // Computers, Materials & Continua. — 2023. — Vol. 74, No. 3. aSc Applied Software Consultants. aSc TimeTables [Электронный ресурс]. — URL: <https://www.asctimetables.com/> — Дата обращения: 11.04.2026.
2. aSc Applied Software Consultants. aSc TimeTables [Электронный ресурс]. — URL: <https://www.asctimetables.com/> — Дата обращения: 11.04.2026.
3. UniTime. University Timetabling [Электронный ресурс]. — URL: <https://www.unitime.org/index.php?tab=1> — Дата обращения: 11.04.2026.
4. Müller T., Rudová H., Barták R. University Course Timetabling: Solver Evolution // Proceedings of the 11th International Conference on the Practice and Theory of Automated Timetabling. — 2016.
5. Фирма «1С». 1С:Автоматизированное составление расписания [Электронный ресурс]. — URL: https://solutions.1c.ru/catalog/asp_univer/features — Дата обращения: 11.04.2026.
6. Microsoft. C# documentation [Электронный ресурс]. — URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/> — Дата обращения: 17.04.2026.
7. Oracle. Java Documentation [Электронный ресурс]. — URL: <https://docs.oracle.com/en/java/> — Дата обращения: 17.04.2026.
8. Лутц М. Изучаем Python. Том 1. 5-е изд. — СПб.: Диалектика, 2019.
9. Python Software Foundation. The Python Language Reference [Электронный ресурс]. — URL: <https://docs.python.org/3/reference/index.html> — Дата обращения: 17.04.2026.
10. Anaconda, Inc. Anaconda Navigator Documentation [Электронный ресурс]. — URL: <https://www.anaconda.com/docs/tools/anaconda-navigator/main> — Дата обращения: 17.04.2026.
11. Spyder Project Contributors. Spyder Documentation [Электронный ресурс]. — URL: <https://docs.spyder-ide.org/> — Дата обращения: 17.04.2026.

12. Конова П. С. Библиотека Streamlit как инструмент обработки и визуализации больших данных // Столыпинский вестник. — 2022. — № 3(2).

13. Streamlit, Inc. Streamlit Documentation [Электронный ресурс]. — URL: <https://docs.streamlit.io/> — Дата обращения: 17.04.2026.

14. Python Software Foundation. sqlite3 — DB-API 2.0 interface for SQLite databases [Электронный ресурс]. — URL: <https://docs.python.org/3/library/sqlite3.html> — Дата обращения: 24.04.2026.

15. Маккинни У. Python и анализ данных: первичная обработка данных с применением pandas, NumPy и Jupyter. 3-е изд. — М.: ДМК Пресс, 2023.

16. pandas. pandas.read_excel Documentation [Электронный ресурс]. — URL: https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html — Дата обращения: 09.05.2026.

17. openpyxl. openpyxl Documentation [Электронный ресурс]. — URL: <https://openpyxl.readthedocs.io/> — Дата обращения: 09.05.2026.

18. Google Developers. CP-SAT Solver — OR-Tools Documentation [Электронный ресурс]. — URL: https://developers.google.com/optimization/cp/cp_solver — Дата обращения: 24.04.2026.

19. Постановление Главного государственного санитарного врача РФ от 02.12.2020 № 40 «Об утверждении санитарных правил СП 2.2.3670-20 “Санитарно-эпидемиологические требования к условиям труда”» [Электронный ресурс]. — URL: https://www.consultant.ru/document/cons_doc_LAW_372741/ — Дата обращения: 21.05.2026.

20. Постановление Правительства РФ от 16.09.2020 № 1479 «Об утверждении Правил противопожарного режима в Российской Федерации» [Электронный ресурс]. — URL: https://www.consultant.ru/document/cons_doc_LAW_363263/ — Дата обращения: 21.05.2026.

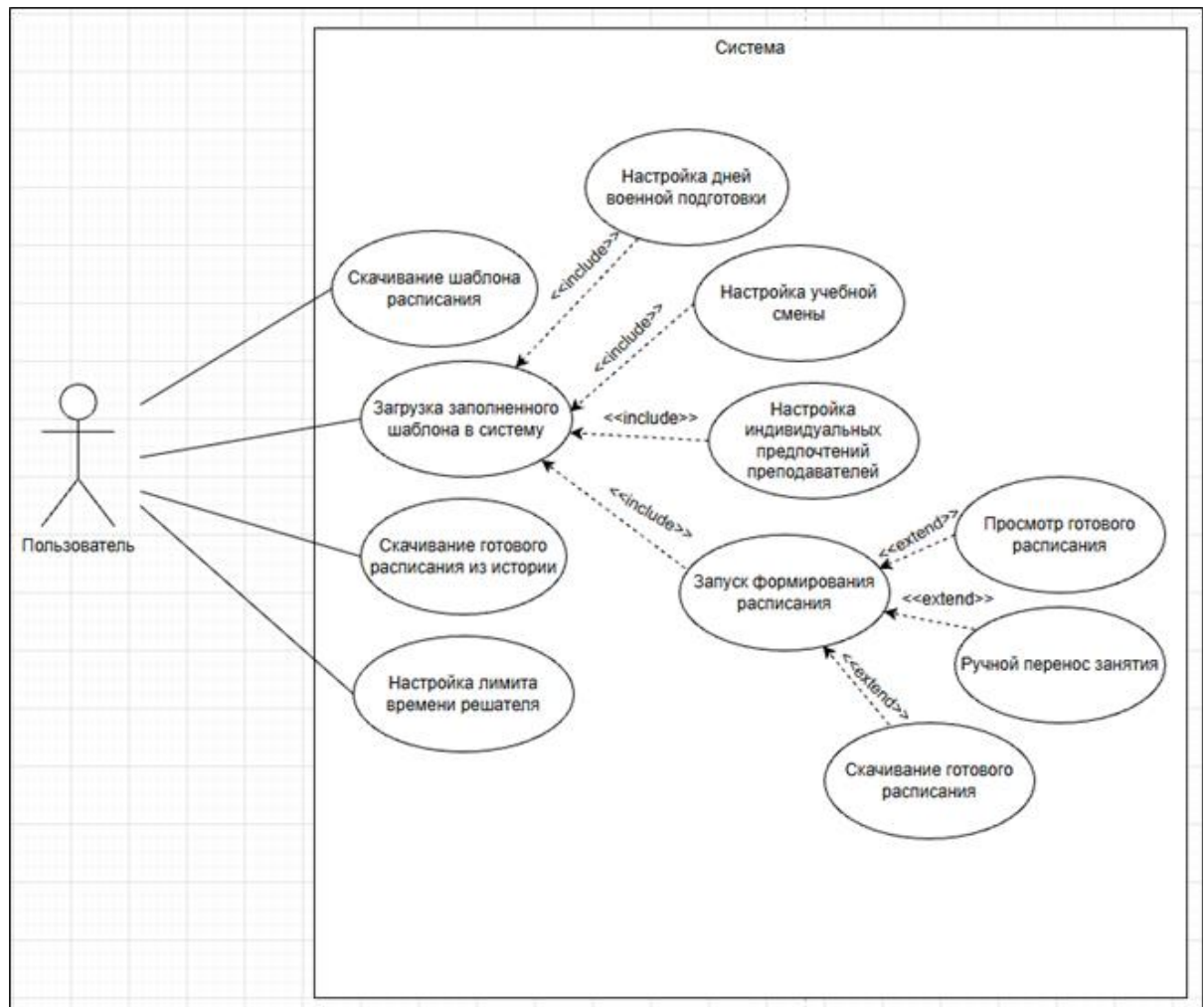
Приложение А

(обязательное)

Система автоматизации составления учебного расписания с использованием
методов оптимизации и учетом ресурсных ограничений

UML-диаграммы

					ПГУ 1.090301.27.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		



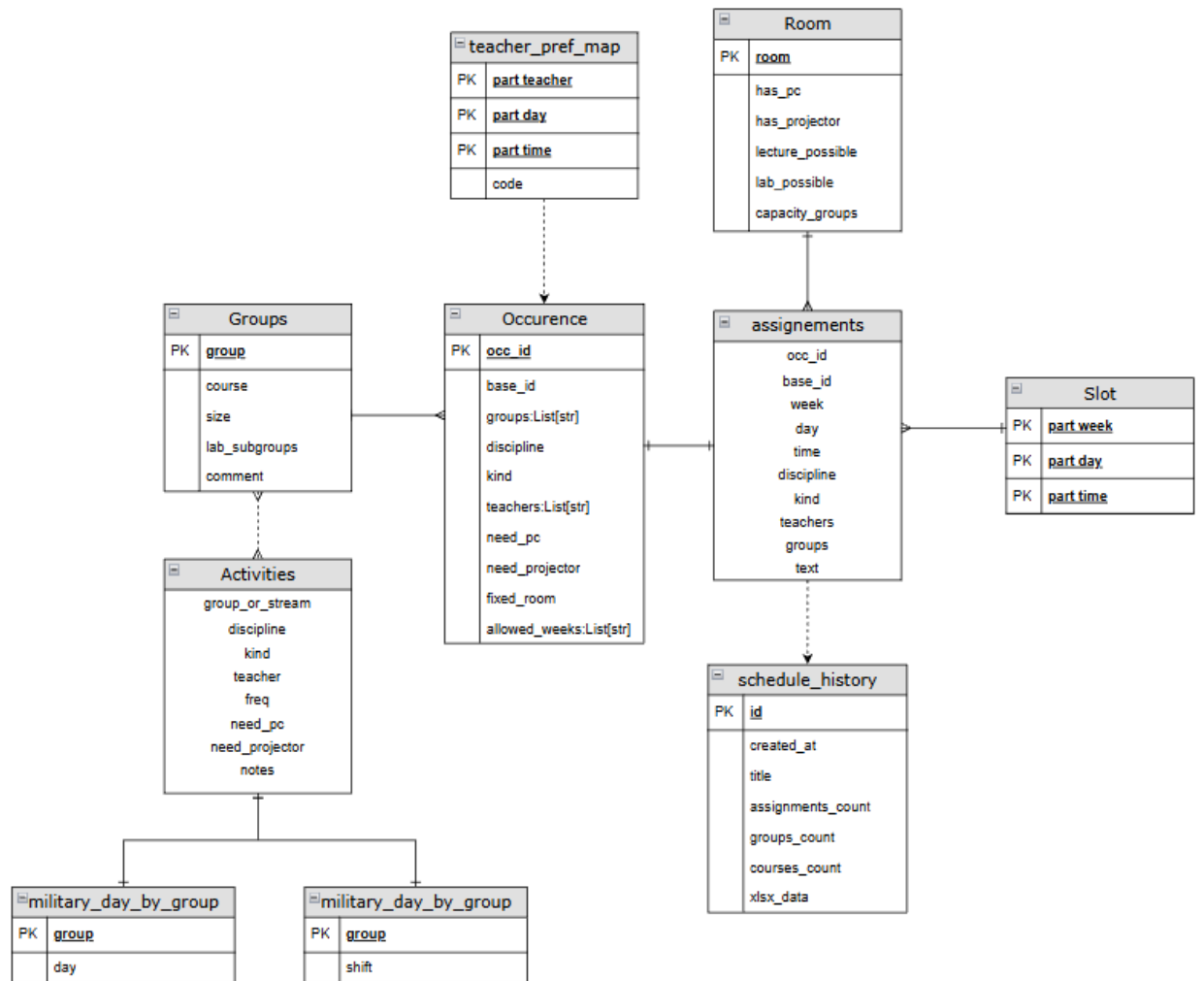
ПГУ1.090301.27.002 Д13									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Степанов	Иван Иванович	Петров	Сергей Сергеевич	Сидоров	Александр Александрович	Трофимов	Владимир Владимирович	Федотов	Андрей Андреевич
Группа	Группа	Группа	Группа	Группа	Группа	Группа	Группа	Группа	Группа
1 курс	1 курс	1 курс	1 курс	1 курс	1 курс	1 курс	1 курс	1 курс	1 курс
В. преподаватель	В. преподаватель	В. преподаватель	В. преподаватель	В. преподаватель	В. преподаватель	В. преподаватель	В. преподаватель	В. преподаватель	В. преподаватель
М. преподаватель	М. преподаватель	М. преподаватель	М. преподаватель	М. преподаватель	М. преподаватель	М. преподаватель	М. преподаватель	М. преподаватель	М. преподаватель

Изм.	Лист	№ докум.	Подпись	Дата

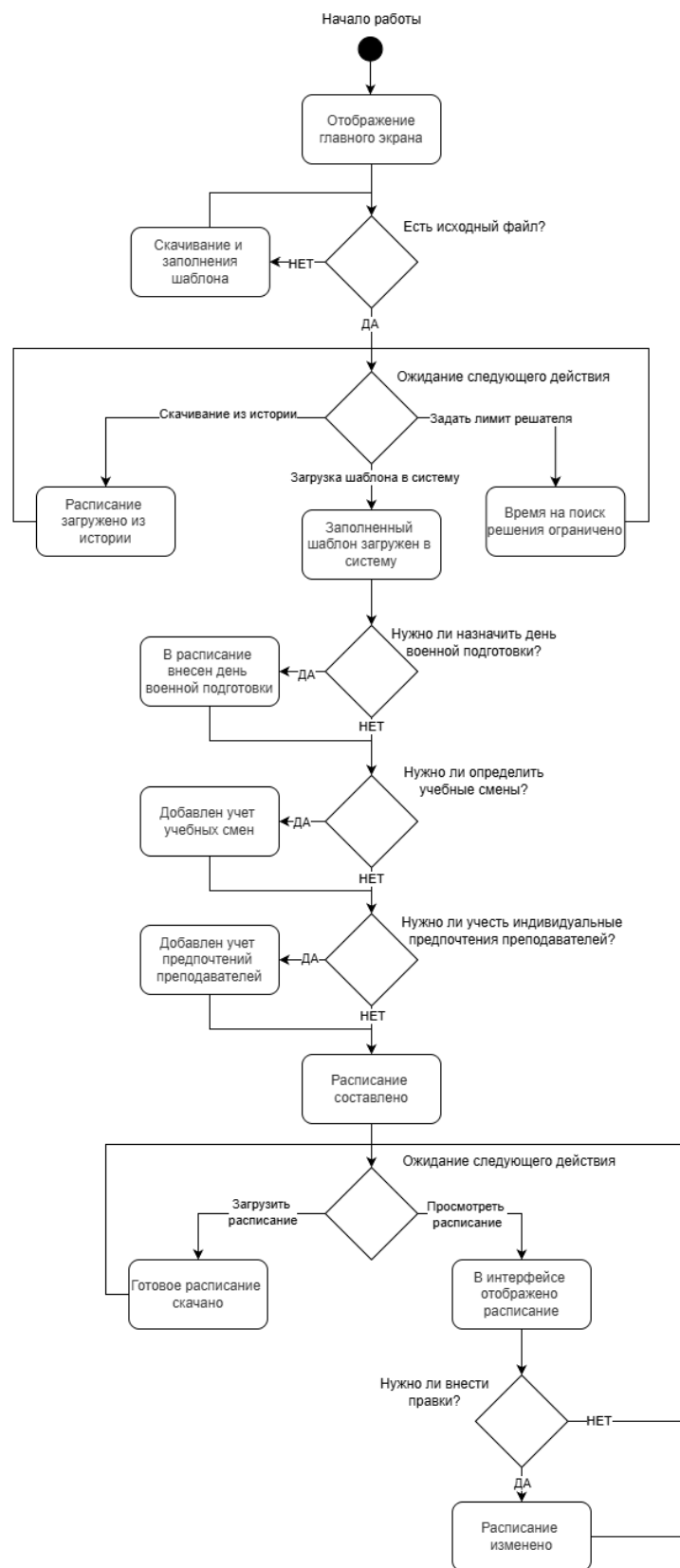
ПГУ 1.090301.27.001 ПЗ

Лист

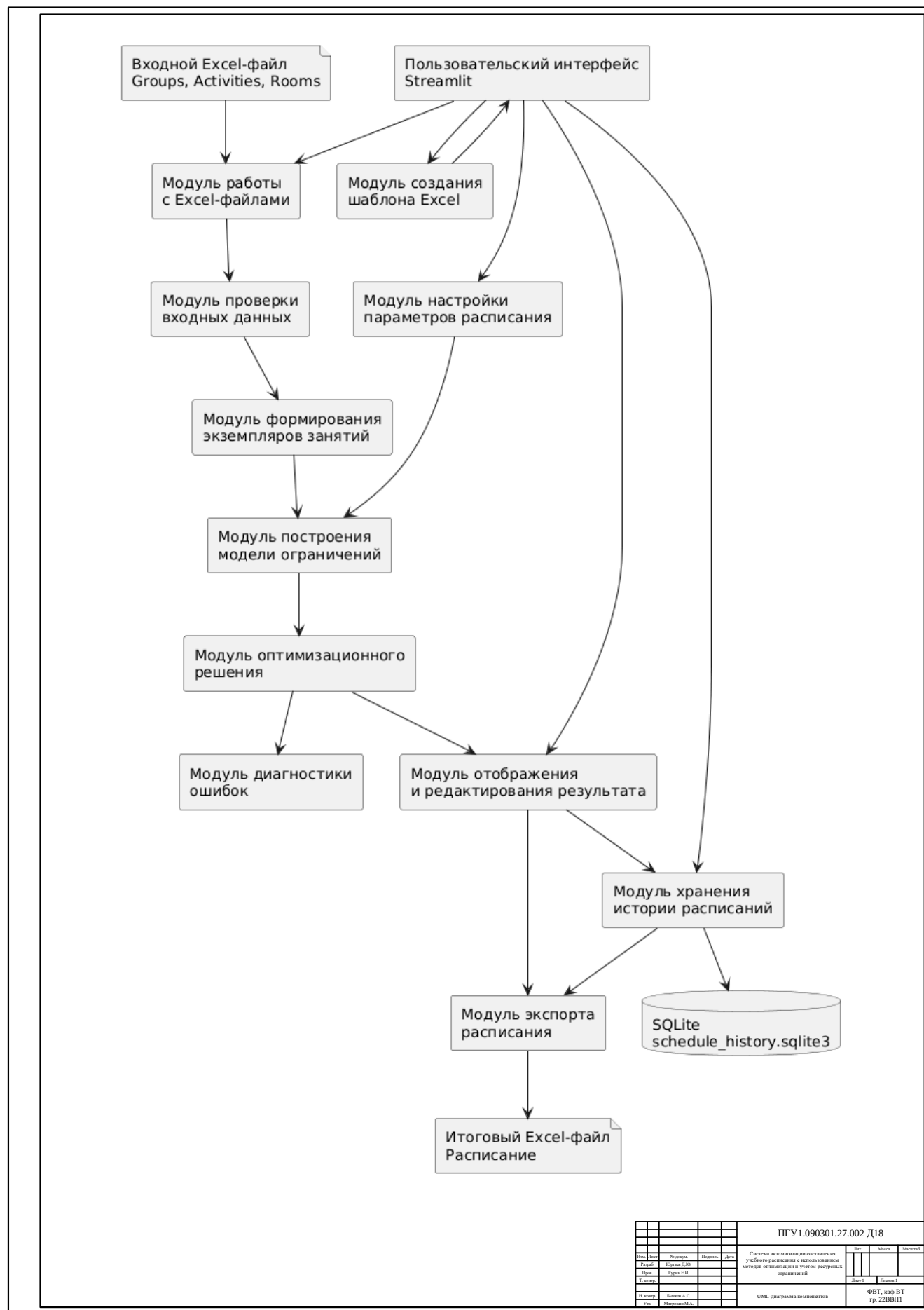
69



ПГУ1.090301.27.002 Д6									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия



ПГУ 1.090301.27.002 Д17									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия



ПГУ 1.090301.27.002 Д18									
Доп. лист	№ листа	Полное	Дата	Система автоматизации составления учебного расписания с использованием методов оптимизации и учетом ресурсов организаций					
Создан	Курсов ДЮ			Лист 1				Листов 1	
Дата	Группа Г.П.			ИМБ – инженерная школа					
Уч. центр	Департамент А.С.			ФЭТ, кафедра ВТ					
И.И. состав	Меркулов М.А.			г.р. 2288811					
Уч. центр									

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.27.001 ПЗ

Лист

72

Приложение Б

(обязательное)

Система автоматизации составления учебного расписания с использованием
методов оптимизации и учетом ресурсных ограничений

Листинг программы

					ПГУ 1.090301.27.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

Файл app.py:

```
from __future__ import annotations

import math
import re
import html
import json
import sqlite3
from datetime import datetime
from dataclasses import dataclass
from io import BytesIO
from pathlib import Path
from typing import Dict, List, Optional, Tuple

import pandas as pd
import streamlit as st
from openpyxl import Workbook
from openpyxl.styles import Alignment, Border, Font, PatternFill, Side
from openpyxl.utils import get_column_letter
from ortools.sat.python import cp_model

HERE = Path(__file__).resolve().parent
SCHEDULE_HISTORY_DB = HERE / "schedule_history.sqlite3"
SCHEDULE_HISTORY_LIMIT = 10
DEFAULT_DAYS = ["ПН", "ВТ", "СР", "ЧТ", "ПТ", "СБ"]
EARLY_TIME = "08:00"
DEFAULT_TIMES = [EARLY_TIME, "09:50", "11:40", "13:45", "15:35", "17:25"]
WEEKS = ["А", "В"]
MILITARY_TEXT = "День военной подготовки"
PREF_NEUTRAL = "-"
PREF_PREFERRED = "☐ Предпочтительно"
PREF_UNAVAILABLE = "☐ Недоступно"
PREF_OPTIONS = [PREF_NEUTRAL, PREF_PREFERRED, PREF_UNAVAILABLE]

TEMPLATE_HINT_MARKERS = {"пример", "example", "__example__", "строка-подсказка"}

def norm_str(x) -> str:
    if x is None or (isinstance(x, float) and pd.isna(x)):
        return ""
    return str(x).strip()

def parse_bool(v) -> bool:
    s = norm_str(v).lower()
    if not s:
        return False
```

					ПГУ 1.090301.27.001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

```

    return s in {"1", "да", "true", "+", "yes", "y", "ok", "скорее
да", "возможно"}

def normalize_activity_kind(kind: str) -> str:

    k = norm_str(kind).upper()
    if k in ("ЛК", "ЛЕК", "ЛЕКЦИЯ"):
        return "ЛК"
    if k in ("ЛБ", "ЛАБ", "ЛАБОРАТОРНАЯ", "ПЗ", "ПРАК", "ПРАКТИКА"):
        return "ЛБ"
    return k

def kind_to_short(kind: str) -> str:
    k = normalize_activity_kind(kind)
    if k == "ЛК":
        return "лек."
    if k == "ЛБ":
        return "лб."
    return k.lower() + "." if k else "зан."

def split_teachers(s: str) -> List[str]:
    s = norm_str(s)
    if not s:
        return []
    parts = re.split(r"\s*,\s*|\s*;\s*|\s+n\s+", s)
    return [p.strip() for p in parts if p.strip()]

def parse_groups_token(token: str) -> List[str]:
    token = norm_str(token)
    if not token:
        return []
    token = token.replace(",", " ")
    chunks = [c for c in token.split() if c.strip()]

    out: List[str] = []
    for ch in chunks:

        m_full = re.match(r"^(\\d{2}[A-Za-zA-Яа-яЁё]+)(\\d+)-(\\d{2}[A-
Za-zA-Яа-яЁё]+)(\\d+)$", ch)
        if m_full and m_full.group(1) == m_full.group(3):
            base = m_full.group(1)
            a_s, b_s = m_full.group(2), m_full.group(4)
            a, b = int(a_s), int(b_s)
            step = 1 if b >= a else -1
            width = max(len(a_s), len(b_s))
            for i in range(a, b + step, step):
                out.append(f"{base}{str(i).zfill(width)}")
            continue

```

```

        m_short = re.match(r"^(\d{2}[A-Za-zA-Яa-яЁё]+)(\d+)-(\d+)$",
ch)
        if m_short:
            base, a_s, b_s = m_short.group(1), m_short.group(2),
m_short.group(3)

            if len(a_s) == len(b_s):
                a, b = int(a_s), int(b_s)
                step = 1 if b >= a else -1
                width = len(a_s)
                for i in range(a, b + step, step):
                    out.append(f"{base}{str(i).zfill(width)}")
            else:
                out.append(ch)
        else:
            out.append(ch)

    seen = set()
    res = []
    for g in out:
        if g not in seen:
            res.append(g)
            seen.add(g)
    return res

def safe_sheet_name(name: str) -> str:
    bad = set(r'[:*?/\\'')
    cleaned = "".join("_" if ch in bad else ch for ch in name)
    return cleaned[:31] or "Sheet"

def parse_days_and_times(settings: dict) -> Tuple[List[str],
List[str]]:
    days = DEFAULT_DAYS
    times = DEFAULT_TIMES

    custom_days = norm_str(settings.get("days") or
settings.get("day_list"))
    if custom_days:
        parsed_days = [x.strip().upper() for x in re.split(r"[,;]",
custom_days) if x.strip()]
        if parsed_days:
            days = parsed_days

    custom_times = norm_str(settings.get("pair_times") or
settings.get("times") or settings.get("time_slots"))
    if custom_times:
        parsed_times = [x.strip() for x in re.split(r"[,;]",
custom_times) if x.strip()]
        if parsed_times:

```

```

        times = parsed_times

    times = sorted(list(dict.fromkeys(times)), key=time_to_minutes)
    return days, times

def effective_times_for_shifts(base_times: List[str],
course_shift_choices: Dict[int, str]) -> List[str]:

    times = list(base_times)
    if EARLY_TIME not in times:
        times = [EARLY_TIME] + times
    return sorted(list(dict.fromkeys(times)), key=time_to_minutes)

def export_times_for_course(course_groups: List[str],
group_shift_map: Dict[str, str], solver_times: List[str]) ->
List[str]:

    times = list(solver_times)
    return sorted(list(dict.fromkeys(times)), key=time_to_minutes)

def time_to_minutes(value: str) -> int:
    s = norm_str(value)
    m = re.match(r"^(\d{1,2}):(\d{2})$", s)
    if not m:
        return 24 * 60
    return int(m.group(1)) * 60 + int(m.group(2))

def format_assignment_text(discipline: str, kind: str, teachers:
List[str], room: str) -> str:
    discipline_s = norm_str(discipline) or "Дисциплина не указана"
    kind_s = kind_to_short(kind) or "тип не указан"
    teachers_s = ", ".join(teachers) if teachers else "преп. не
указан"
    room_s = norm_str(room) if norm_str(room) else "не указана"
    return f"{discipline_s}\n{kind_s} {teachers_s}\nауд. {room_s}"

def pref_to_code(value: str) -> str:
    value = norm_str(value)
    if value == PREF_PREFERRED:
        return "preferred"
    if value == PREF_UNAVAILABLE:
        return "unavailable"
    return "neutral"

def code_to_pref(value: str) -> str:

```

```

    if value == "preferred":
        return PREF_PREFERRED
    if value == "unavailable":
        return PREF_UNAVAILABLE
    return PREF_NEUTRAL

def is_template_hint_value(value) -> bool:

    s = norm_str(value).lower()
    if not s:
        return False
    return s in TEMPLATE_HINT_MARKERS or s.startswith("пример") or
s.startswith("строка-подсказка")

def drop_template_hint_rows(df: pd.DataFrame, marker_columns:
List[str]) -> pd.DataFrame:

    if df is None or df.empty:
        return df

    mask = pd.Series(False, index=df.index)
    for col in marker_columns:
        if col in df.columns:
            mask = mask | df[col].apply(is_template_hint_value)
    return df.loc[~mask].reset_index(drop=True)

def style_template_sheet(ws):
    header_fill = PatternFill("solid", fgColor="D9EAF7")
    hint_fill = PatternFill("solid", fgColor="FFF2CC")
    thin = Side(style="thin", color="B7B7B7")

    for row in ws.iter_rows():
        for cell in row:
            cell.alignment = Alignment(vertical="top",
wrap_text=True)
            cell.border = Border(left=thin, right=thin, top=thin,
bottom=thin)
            if cell.row == 1:
                cell.font = Font(bold=True)
                cell.fill = header_fill
                cell.alignment = Alignment(horizontal="center",
vertical="center", wrap_text=True)
            elif cell.row == 2:
                cell.fill = hint_fill
                cell.font = Font(italic=True, color="666666")

    for col_idx in range(1, ws.max_column + 1):
        col_letter = get_column_letter(col_idx)
        max_len = 12
        for row_idx in range(1, min(ws.max_row, 8) + 1):

```

```

        value = ws.cell(row_idx, col_idx).value
        if value is not None:
            max_len = max(max_len, min(len(str(value)), 42))
        ws.column_dimensions[col_letter].width = max_len + 2

ws.freeze_panes = "A2"
ws.row_dimensions[1].height = 24
if ws.max_row >= 2:
    ws.row_dimensions[2].height = 70

def build_input_template_workbook() -> BytesIO:

    wb = Workbook()

    ws_groups = wb.active
    ws_groups.title = "Groups"
    ws_groups.append(["group", "course", "size", "lab_subgroups",
"comment"])
    ws_groups.append([
        "ПРИМЕР: 25BVB1",
        "ПРИМЕР: 1",
        "необязательно: 25",
        "необязательно: 2",
        "Строка-подсказка не учитывается при распределении. Удалите
её или оставьте – программа пропустит.",
    ])

    ws_activities = wb.create_sheet("Activities")
    ws_activities.append(["group_or_stream", "discipline", "kind",
"teacher", "freq", "need_pc", "need_projector", "notes"])
    ws_activities.append([
        "ПРИМЕР: 25BVB1 или 25BVB1-25BVB4",
        "ПРИМЕР: Математика",
        "ПРИМЕР: ЛК или ЛБ",
        "ПРИМЕР: Иванов И.И.; Петров П.П.",
        "ПРИМЕР: 1 или 0.5",
        "0 или 1",
        "0 или 1",
        "Строка-подсказка не учитывается. ID занятия заполнять не
нужно – программа назначит его сама. freq: 1 – каждую неделю, 0.5 –
раз в две недели, 1.5 – чередование 1/2 пары.",
    ])

    ws_rooms = wb.create_sheet("Rooms")
    ws_rooms.append(["room", "has_projector", "has_computers",
"lecture_possible", "lab_possible", "capacity", "notes"])
    ws_rooms.append([
        "ПРИМЕР: 7a-316",
        "да или нет",
        "да или нет",
        "да или нет",
        "да или нет",
    ])

```

```

        "число мест или число групп, например 50 или 2",
        "Строка-подсказка не учитывается. capacity можно оставить
пустым, тогда аудитория считается вместительной.",
    ])

```

```

for ws in wb.worksheets:
    style_template_sheet(ws)

```

```

bio = BytesIO()
wb.save(bio)
bio.seek(0)
return bio

```

```

@dataclass(frozen=True)
class Room:
    room: str
    has_pc: bool
    has_projector: bool
    lecture_possible: bool
    lab_possible: bool
    capacity_groups: int

```

```

@dataclass(frozen=True)
class Slot:
    week: str
    day: str
    time: str

```

```

@dataclass
class Occurrence:
    occ_id: str
    base_id: str
    groups: List[str]
    discipline: str
    kind: str
    teachers: List[str]
    need_pc: bool
    need_projector: bool
    fixed_room: str
    allowed_weeks: List[str]

```

```

class NoFeasibleOptions(Exception):
    def __init__(self, occ: Occurrence, diag: dict):
        self.occ = occ
        self.diag = diag
        super().__init__(f"Нет допустимых слотов/аудиторий для
{occ.occ_id}")

```

```

class NoScheduleFound(Exception):
    def __init__(self, message: str, diag: Optional[dict] = None):
        self.diag = diag or {}
        super().__init__(message)

def slot_is_teacher_unavailable(teachers: List[str], day: str, time:
str, teacher_pref_map: Dict[str, Dict[Tuple[str, str], str]]) ->
bool:
    for teacher in teachers:
        if teacher_pref_map.get(teacher, {}).get((day, time)) ==
"unavailable":
            return True
    return False

def slot_preference_score(teachers: List[str], day: str, time: str,
teacher_pref_map: Dict[str, Dict[Tuple[str, str], str]]) -> int:
    score = 0
    for teacher in teachers:
        if teacher_pref_map.get(teacher, {}).get((day, time)) ==
"preferred":
            score += 1
    return score

def diagnose_no_options(
    occ: Occurrence,
    rooms: List[Room],
    available_slots: List[Slot],
    military_days_by_group: Dict[str, str],
    teacher_pref_map: Dict[str, Dict[Tuple[str, str], str]],
    group_shift_map: Dict[str, str],
) -> dict:
    gcount = len(occ.groups)
    k = norm_str(occ.kind).upper()

    slots_total = len(available_slots)
    slots_allowed_week = sum(1 for s in available_slots if s.week in
occ.allowed_weeks)
    slots_after_military = sum(
        1
        for s in available_slots
        if s.week in occ.allowed_weeks and
all(military_days_by_group.get(g, "") != s.day for g in occ.groups)
    )
    slots_after_teacher_restrictions = sum(
        1
        for s in available_slots
        if s.week in occ.allowed_weeks
        and all(military_days_by_group.get(g, "") != s.day for g in
occ.groups)

```

```

        and slot_allowed_by_shift(group_shift_map, occ.groups,
s.day, s.time)
        and not slot_is_teacher_unavailable(occ.teachers, s.day,
s.time, teacher_pref_map)
    )

def pass_fixed(r: Room) -> bool:
    return (not occ.fixed_room) or (r.room == occ.fixed_room)

def pass_pc(r: Room) -> bool:
    return (not occ.need_pc) or r.has_pc

def pass_proj(r: Room) -> bool:
    return (not occ.need_projector) or r.has_projector

def pass_kind(r: Room) -> bool:
    if k in ("ЛБ", "ЛАБ", "ЛАБОРАТОРНАЯ"):
        return r.lab_possible
    if k in ("ЛК", "ЛЕК", "ЛЕКЦИЯ"):
        return r.lecture_possible
    return True

def pass_cap(r: Room) -> bool:
    return r.capacity_groups >= gcount

rooms_fixed = [r for r in rooms if pass_fixed(r)]
rooms_pc = [r for r in rooms_fixed if pass_pc(r)]
rooms_proj = [r for r in rooms_pc if pass_proj(r)]
rooms_kind = [r for r in rooms_proj if pass_kind(r)]
rooms_cap = [r for r in rooms_kind if pass_cap(r)]

fail_details = []
for r in rooms:
    reasons = []
    if not pass_fixed(r):
        reasons.append("fixed_room!=room")
    if not pass_pc(r):
        reasons.append("need_pc")
    if not pass_proj(r):
        reasons.append("need_projector")
    if not pass_kind(r):
        reasons.append("kind_not_allowed")
    if not pass_cap(r):
        reasons.append(f"capacity<{gcount}groups")
    if reasons:
        fail_details.append({"room": r.room, "reasons":
reasons})

teacher_restrictions = {}
for teacher in occ.teachers:
    restricted = [f"{day} {time}" for (day, time), code in
teacher_pref_map.get(teacher, {}).items() if code == "unavailable"]

```

```

        preferred = [f"{day} {time}" for (day, time), code in
teacher_pref_map.get(teacher, {}).items() if code == "preferred"]
        teacher_restrictions[teacher] = {
            "unavailable": restricted,
            "preferred": preferred,
        }

    return {
        "occ_id": occ.occ_id,
        "discipline": occ.discipline,
        "kind": occ.kind,
        "teachers": occ.teachers,
        "groups": occ.groups,
        "fixed_room": occ.fixed_room,
        "allowed_weeks": occ.allowed_weeks,
        "military_days_for_groups": {g:
military_days_by_group.get(g, "") for g in occ.groups},
        "shift_for_groups": {g: group_shift_map.get(g, "Не
задавать") for g in occ.groups},
        "teacher_preferences": teacher_restrictions,
        "slots_total": slots_total,
        "slots_allowed_week": slots_allowed_week,
        "slots_after_military": slots_after_military,
        "slots_after_teacher_restrictions":
slots_after_teacher_restrictions,
        "rooms_total": len(rooms),
        "rooms_after_fixed_room": len(rooms_fixed),
        "rooms_after_pc": len(rooms_pc),
        "rooms_after_projector": len(rooms_proj),
        "rooms_after_kind": len(rooms_kind),
        "rooms_after_capacity": len(rooms_cap),
        "rooms_fail_examples": fail_details[:10],
    }

def rooms_df_to_rooms(rooms_df: pd.DataFrame) -> List[Room]:

    rooms_df = rooms_df.rename(columns={
        "has_pc": "has_computers",
        "computers": "has_computers",
        "pc": "has_computers",
        "projector": "has_projector",
        "lecture": "lecture_possible",
        "lab": "lab_possible",
    })
    rooms_df = drop_template_hint_rows(rooms_df, ["room"])

    rooms: List[Room] = []
    seen = set()
    for _, r in rooms_df.iterrows():
        room = norm_str(r.get("room"))
        if not room or room in seen:
            continue

```

```

        seen.add(room)

        cap = r.get("capacity")
        if pd.isna(cap) or not norm_str(cap):
            cap_groups = 999
        else:
            try:
                cap_int = int(float(str(cap).replace(",", ""
".").strip()))

                if cap_int <= 10:
                    cap_groups = max(1, cap_int)
                else:
                    cap_groups = max(1, math.ceil(cap_int / 25))
            except Exception:
                cap_groups = 999

        rooms.append(
            Room(
                room=room,
                has_pc=parse_bool(r.get("has_computers")),
                has_projector=parse_bool(r.get("has_projector")),
                lecture_possible=parse_bool(r.get("lecture_possible", "да")) or not
norm_str(r.get("lecture_possible")),
                lab_possible=parse_bool(r.get("lab_possible", "да"))
or not norm_str(r.get("lab_possible")),
                capacity_groups=cap_groups,
            )
        )

    return rooms

def build_group_to_course_from_sheet(groups_df: pd.DataFrame) ->
Dict[str, int]:

    result: Dict[str, int] = {}
    if groups_df is None or groups_df.empty:
        return result

    for _, row in groups_df.iterrows():
        raw_group = norm_str(row.get("group"))
        if not raw_group:
            continue
        course_raw = pd.to_numeric(row.get("course"),
errors="coerce")
        if pd.isna(course_raw):
            continue
        course = int(course_raw)
        if course <= 0:
            continue

```

```

        for g in parse_groups_token(raw_group) or [raw_group]:
            if g:
                result[g] = course
    return result

def validate_activity_groups(groups_df: pd.DataFrame, acts_df:
pd.DataFrame) -> None:
    known_groups =
set(build_group_to_course_from_sheet(groups_df).keys())
    if not known_groups:
        raise ValueError("На листе Groups не найдено ни одной группы
с указанным курсом.")

    problems = []
    for row_no, (_, row) in enumerate(acts_df.iterrows(), start=2):
        raw = norm_str(row.get("group_or_stream") or
row.get("group"))
        if not raw:
            continue
        parsed = parse_groups_token(raw)
        unknown = [g for g in parsed if g not in known_groups]
        if unknown:
            problems.append({"row": row_no, "raw": raw, "unknown":
unknown})

    if problems:
        lines = []
        for item in problems[:10]:
            lines.append(
                f"строка {item['row']}: '{item['raw']}' → не найдены
в Groups: {'', '.join(item['unknown'][:8])}"
            )
        suffix = "\n..." if len(problems) > 10 else ""
        raise ValueError(
            "В Activities есть группы, которых нет на листе Groups.
"
            "Проверьте запись потоков/диапазонов. Например, для двух
групп пишете "
            "253BЭЭ51-253BЭЭ52 или 253BЭЭ51-52, а не 253BЭЭ51-
2.\n\n"
            + "\n".join(lines)
            + suffix
        )

def read_dynamic(file_bytes: bytes) -> Tuple[dict, pd.DataFrame,
pd.DataFrame, pd.DataFrame, List[Room]]:
    xls = pd.ExcelFile(BytesIO(file_bytes))

    required_sheets = {"Groups", "Activities", "Rooms"}
    missing = sorted(required_sheets - set(xls.sheet_names))
    if missing:

```

```

        raise ValueError("Во входном файле не хватает листов: " + ",
".join(missing) + ". Скачайте новый шаблон и заполните его.")

```

```

settings = {
    "max_pairs_per_day": 4,
    "days": ", ".join(DEFAULT_DAYS),
    "times": ", ".join(DEFAULT_TIMES),
}

if "Settings" in xls.sheet_names:
    settings_df = pd.read_excel(xls, sheet_name="Settings")
    legacy_settings = {norm_str(r["param"]): r.get("value") for
_, r in settings_df.iterrows() if norm_str(r.get("param"))}
    for key in ("semester_name", "weeks_in_semester"):
        if key in legacy_settings:
            settings[key] = legacy_settings[key]

groups_df = pd.read_excel(xls, sheet_name="Groups")
acts_df = pd.read_excel(xls, sheet_name="Activities")
rooms_df = pd.read_excel(xls, sheet_name="Rooms")

groups_df = drop_template_hint_rows(groups_df, ["group"])
acts_df = drop_template_hint_rows(acts_df, ["activity_id", "id",
"group_or_stream", "discipline", "kind"])
rooms_df = drop_template_hint_rows(rooms_df, ["room"])

acts_df = acts_df.rename(columns={
    "teachers": "teacher",
    "group": "group_or_stream",
})

validate_activity_groups(groups_df, acts_df)

rooms = rooms_df_to_rooms(rooms_df)
if not rooms:
    raise ValueError("На листе Rooms не найдено ни одной
аудитории. Добавьте аудитории в шаблон.")

return settings, groups_df, acts_df, rooms_df, rooms

def parse_freq_per_week(raw_value, hours_semester=None,
weeks_in_semester: int = 17) -> Optional[float]:

    raw = norm_str(raw_value).replace(",", ".").strip()
    if not raw:
        if hours_semester is None or pd.isna(hours_semester):
            return None
        try:
            return float(hours_semester) / 2.0 / max(1,
weeks_in_semester)
        except Exception:

```

					ПГУ 1.090301.27.001 ПЗ	Лист
						86
Изм.	Лист	№ докум.	Подпись	Дата		

```

        return None

    upper = raw.upper()
    legacy_map = {
        "W": 1.0,
        "WEEKLY": 1.0,
        "ЕЖЕНЕДЕЛЬНО": 1.0,
        "AB": 0.5,
        "A": 0.5,
        "B": 0.5,
        "РАЗ В 2 НЕДЕЛИ": 0.5,
        "РАЗ В ДВЕ НЕДЕЛИ": 0.5,
    }
    if upper in legacy_map:
        return legacy_map[upper]

    try:
        value = float(raw)
        if value <= 0:
            return None

        return round(value * 2) / 2
    except Exception:
        return None

def expand_activities_to_occurrences(settings: dict, acts_df:
pd.DataFrame) -> List[Occurrence]:
    weeks = int(settings.get("weeks_in_semester", 17) or 17)
    occs: List[Occurrence] = []
    warn = []

    for row_no, (_, a) in enumerate(acts_df.iterrows(), start=1):

        act_id = f"A{row_no:04d}"

        if not any(norm_str(a.get(col)) for col in
["group_or_stream", "group", "discipline", "kind", "teacher",
"teachers", "freq"]):
            continue

        groups = parse_groups_token(a.get("group_or_stream") or
a.get("group"))
        if not groups:
            warn.append(f"activity {act_id}: не разобрал группы")
            continue

        discipline = norm_str(a.get("discipline"))
        kind = normalize_activity_kind(a.get("kind"))
        teachers = split_teachers(a.get("teacher") or
a.get("teachers"))
        fixed_room = norm_str(a.get("room"))

```

```

need_pc = int(a.get("need_pc") or 0) == 1
need_proj = int(a.get("need_projector") or 0) == 1

freq_per_week = parse_freq_per_week(
    a.get("freq"),
    hours_semester=a.get("hours_semester"),
    weeks_in_semester=weeks,
)
if freq_per_week is None:
    warn.append(f"activity {act_id}: не удалось понять
freq='{norm_str(a.get('freq'))}'")
    continue

total_pairs_in_two_weeks = int(round(freq_per_week * 2))
weekly_pairs = total_pairs_in_two_weeks // 2
extra_single_pair = total_pairs_in_two_weeks % 2

def add_occ(base_id: str, suffix: str, allowed_weeks:
List[str]):
    occs.append(
        Occurrence(
            occ_id=f"{base_id}_{suffix}",
            base_id=base_id,
            groups=groups,
            discipline=discipline,
            kind=kind,
            teachers=teachers,
            need_pc=need_pc,
            need_projector=need_proj,
            fixed_room=fixed_room,
            allowed_weeks=allowed_weeks,
        )
    )

for i in range(weekly_pairs):
    base = f"{act_id}#W{i+1}"
    add_occ(base, "A", ["A"])
    add_occ(base, "B", ["B"])

if extra_single_pair:
    add_occ(f"{act_id}#ALT", "ALT", ["A", "B"])

if warn:
    st.warning("Есть замечания по входным данным:\n\n" +
"\n".join(warn[:20]) + ("\n..." if len(warn) > 20 else ""))

return occs

def build_solver_artifacts(
    occs: List[Occurrence],

```

```

rooms: List[Room],
days: List[str],
times: List[str],
military_days_by_group: Dict[str, str],
teacher_pref_map: Dict[str, Dict[Tuple[str, str], str]],
group_shift_map: Dict[str, str],
use_military: bool = True,
use_teacher_unavailable: bool = True,
use_shift: bool = True,
):
    available_slots = [Slot(week=w, day=d, time=t) for d in days for
t in times for w in WEEKS]
    slot_index = {(s.week, s.day, s.time): i for i, s in
enumerate(available_slots)}

    local_rooms = list(rooms)
    known_rooms = {r.room for r in local_rooms}
    for o in occs:
        if o.fixed_room and o.fixed_room not in known_rooms:
            local_rooms.append(Room(o.fixed_room, True, True, True,
True, 999))
            known_rooms.add(o.fixed_room)

    allowed: Dict[int, List[Tuple[int, int]]] = {}
    base_to_occs: Dict[str, List[int]] = {}
    preference_scores: Dict[Tuple[int, int, int], int] = {}

    for i, o in enumerate(occs):
        base_to_occs.setdefault(o.base_id, []).append(i)
        allowed[i] = []
        gcount = len(o.groups)
        for s_idx, s in enumerate(available_slots):
            if s.week not in o.allowed_weeks:
                continue
            if use_military and any(military_days_by_group.get(g,
"") == s.day for g in o.groups):
                continue
            if use_shift and not
slot_allowed_by_shift(group_shift_map, o.groups, s.day, s.time):
                continue
            if use_teacher_unavailable and
slot_is_teacher_unavailable(o.teachers, s.day, s.time,
teacher_pref_map):
                continue
            for r_idx, r in enumerate(local_rooms):
                if o.fixed_room and r.room != o.fixed_room:
                    continue
                if o.need_pc and not r.has_pc:
                    continue
                if o.need_projector and not r.has_projector:
                    continue
                k = norm_str(o.kind).upper()

```

```

        if k in ("ЛБ", "ЛАБ", "ЛАБОРАТОРНАЯ") and not
r.lab_possible:
            continue
        if k in ("ЛК", "ЛЕК", "ЛЕКЦИЯ") and not
r.lecture_possible:
            continue
        if r.capacity_groups < gcount:
            continue
        allowed[i].append((s_idx, r_idx))
        preference_scores[(i, s_idx, r_idx)] =
slot_preference_score(o.teachers, s.day, s.time, teacher_pref_map) +
slot_shift_preference_score(group_shift_map, o.groups, s.day,
s.time)

    return available_slots, slot_index, local_rooms, allowed,
base_to_occs, preference_scores

def build_infeasibility_diagnostics(
    occs: List[Occurrence],
    rooms: List[Room],
    days: List[str],
    times: List[str],
    military_days_by_group: Dict[str, str],
    teacher_pref_map: Dict[str, Dict[Tuple[str, str], str]],
    group_shift_map: Dict[str, str],
    max_pairs_per_day: int,
    time_limit_sec: int,
    allowed: Dict[int, List[Tuple[int, int]]],
):
    diag: Dict[str, object] = {}

    tight = []
    for i, o in enumerate(occs):
        tight.append({
            "occ_id": o.occ_id,
            "discipline": o.discipline,
            "kind": o.kind,
            "groups": ", ".join(o.groups),
            "teachers": ", ".join(o.teachers),
            "allowed_slot_room_variants": len(allowed.get(i, [])),
        })
    tight.sort(key=lambda x: x["allowed_slot_room_variants"])
    diag["Самые ограниченные занятия"] = tight[:15]

    group_rows = []
    all_groups = sorted({g for o in occs for g in o.groups})
    for g in all_groups:
        for w in WEEKS:
            need = sum(1 for o in occs if g in o.groups and w in
o.allowed_weeks)

```

```

        avail = sum(1 for d in days for t in times if (not
military_days_by_group.get(g, "") == d) and
slot_allowed_by_shift(group_shift_map, [g], d, t))
        group_rows.append({"group": g, "week": w, "need_pairs":
need, "available_slots": avail, "deficit": need - avail})
        group_rows.sort(key=lambda x: (x["deficit"], x["need_pairs"]),
reverse=True)
        diag["Потенциально перегруженные группы"] = [r for r in
group_rows[:20] if r["deficit"] > 0 or r["need_pairs"] >= max(1,
r["available_slots"] - 1)]

teacher_rows = []
all_teachers = sorted({t for o in occs for t in o.teachers})
for teacher in all_teachers:
    blocked = teacher_pref_map.get(teacher, {})
    for w in WEEKS:
        need = sum(1 for o in occs if teacher in o.teachers and
w in o.allowed_weeks)
        avail = sum(1 for d in days for t in times if
blocked.get((d, t)) != "unavailable")
        teacher_rows.append({"teacher": teacher, "week": w,
"need_pairs": need, "available_slots": avail, "deficit": need -
avail})
        teacher_rows.sort(key=lambda x: (x["deficit"], x["need_pairs"]),
reverse=True)
        diag["Потенциально перегруженные преподаватели"] = [r for r in
teacher_rows[:20] if r["deficit"] > 0 or r["need_pairs"] >= max(1,
r["available_slots"] - 1)]

relaxation_results = []
tests = [
    ("Убрать правило без окон", dict(use_no_windows=False)),
    ("Игнорировать красные слоты преподавателей",
dict(use_teacher_unavailable=False)),
    ("Игнорировать военную кафедру", dict(use_military=False)),
    ("Игнорировать смены", dict(use_shift=False)),
    ("Отключить синхронизацию А/В",
dict(use_weekly_sync=False)),
]
for title, kwargs in tests:
    ok = quick_feasibility_check(
        occs=occs,
        rooms=rooms,
        days=days,
        times=effective_times,
        max_pairs_per_day=max_pairs_per_day,
        military_days_by_group=military_days_by_group,
        teacher_pref_map=teacher_pref_map,
        group_shift_map=group_shift_map,
        time_limit_sec=min(10, int(time_limit_sec)),
        **kwargs,
    )
    relaxation_results.append({"check": title, "feasible": ok})

```

```

diag["Проверка ослаблений"] = relaxation_results

hints = []
for item in relaxation_results:
    if item["feasible"]:
        hints.append(f"Если {item['check'].lower()}, решение
находится.")
    if not hints:
        hints.append("Даже после ослабления основных правил решение
не находится. Вероятно, не хватает слотов, аудиторий или есть
пересечения по преподавателям/группам.")
diag["Вывод"] = hints
return diag

def quick_feasibility_check(
    occs: List[Occurrence],
    rooms: List[Room],
    days: List[str],
    times: List[str],
    max_pairs_per_day: int,
    military_days_by_group: Dict[str, str],
    teacher_pref_map: Dict[str, Dict[Tuple[str, str], str]],
    group_shift_map: Dict[str, str],
    time_limit_sec: int = 8,
    use_no_windows: bool = True,
    use_weekly_sync: bool = True,
    use_military: bool = True,
    use_teacher_unavailable: bool = True,
    use_shift: bool = True,
) -> bool:
    available_slots, slot_index, local_rooms, allowed, base_to_occs,
_ = build_solver_artifacts(
    occs, rooms, days, times, military_days_by_group,
teacher_pref_map, group_shift_map,
    use_military=use_military,
use_teacher_unavailable=use_teacher_unavailable, use_shift=use_shift
)
    if any(len(v) == 0 for v in allowed.values()):
        return False

    model = cp_model.CpModel()
    y: Dict[Tuple[int, int, int], cp_model.IntVar] = {}
    for i, opts in allowed.items():
        for s_idx, r_idx in opts:
            y[(i, s_idx, r_idx)] =
model.NewBoolVar(f"y_{i}_{s_idx}_{r_idx}")
            model.Add(sum(y[(i, s_idx, r_idx)] for (s_idx, r_idx) in
opts) == 1)

    all_groups = sorted({g for o in occs for g in o.groups})
    all_teachers = sorted({t for o in occs for t in o.teachers})

```

```

def occ_in_slot(i: int, s_idx: int) -> List[cp_model.IntVar]:
    return [y[(i, s_idx, r_idx)] for (ss, r_idx) in allowed[i]
if ss == s_idx]

for g in all_groups:
    occ_ids = [i for i, o in enumerate(occs) if g in o.groups]
    for s_idx in range(len(available_slots)):
        vars_ = []
        for i in occ_ids:
            vars_.extend(occ_in_slot(i, s_idx))
        if vars_:
            model.Add(sum(vars_) <= 1)

for t in all_teachers:
    occ_ids = [i for i, o in enumerate(occs) if t in o.teachers]
    for s_idx in range(len(available_slots)):
        vars_ = []
        for i in occ_ids:
            vars_.extend(occ_in_slot(i, s_idx))
        if vars_:
            model.Add(sum(vars_) <= 1)

for r_idx in range(len(local_rooms)):
    for s_idx in range(len(available_slots)):
        vars_ = [y[(i, s_idx, r_idx)] for i in range(len(occs))
if (i, s_idx, r_idx) in y]
        if vars_:
            model.Add(sum(vars_) <= 1)

if use_weekly_sync:
    for base_id, idxs in base_to_occs.items():
        if len(idxs) != 2:
            continue
        iA, iB = idxs
        oA, oB = occs[iA], occs[iB]
        if oA.allowed_weeks != ["A"] or oB.allowed_weeks !=
["B"]):
            continue
        for d in days:
            for t in times:
                sA = slot_index.get(("A", d, t))
                sB = slot_index.get(("B", d, t))
                if sA is None or sB is None:
                    continue
                model.Add(sum(occ_in_slot(iA, sA)) ==
sum(occ_in_slot(iB, sB)))
                for r_idx in range(len(local_rooms)):
                    varsA = [y[(iA, s_idx, r_idx)] for (s_idx, rr) in
allowed[iA] if rr == r_idx]
                    varsB = [y[(iB, s_idx, r_idx)] for (s_idx, rr) in
allowed[iB] if rr == r_idx]
                    if varsA or varsB:
                        model.Add(sum(varsA) == sum(varsB))

```

```

    if use_no_windows:
        patterns_cache: Dict[int, List[List[int]]] = {}
        group_to_occ_ids = {g: [i for i, o in enumerate(occs) if g
in o.groups] for g in all_groups}
        occ_var: Dict[Tuple[str, int], cp_model.IntVar] = {}
        for g in all_groups:
            occ_ids = group_to_occ_ids[g]
            for s_idx in range(len(available_slots)):
                v = model.NewBoolVar(f"occ_{g}_{s_idx}")
                vars_ = []
                for i in occ_ids:
                    vars_.extend(occ_in_slot(i, s_idx))
                if vars_:
                    model.Add(sum(vars_) == v)
                else:
                    model.Add(v == 0)
            occ_var[(g, s_idx)] = v
        for g in all_groups:
            for w in WEEKS:
                for d in days:
                    s_list = [slot_index[(w, d, t)] for t in times]
                    vec = [occ_var[(g, s_idx)] for s_idx in s_list]
                    T = len(vec)
                    if T not in patterns_cache:
                        patterns_cache[T] = contiguous_patterns(T,
max_pairs_per_day)
                    model.AddAllowedAssignments(vec,
patterns_cache[T])

    solver = cp_model.CpSolver()
    solver.parameters.max_time_in_seconds = float(time_limit_sec)
    solver.parameters.num_search_workers = 8
    status = solver.Solve(model)
    return status in (cp_model.OPTIMAL, cp_model.FEASIBLE)

def contiguous_patterns(T: int, max_ones: int) -> List[List[int]]:
    patterns = [[0] * T]
    for L in range(1, min(max_ones, T) + 1):
        for start in range(0, T - L + 1):
            v = [0] * T
            for i in range(start, start + L):
                v[i] = 1
            patterns.append(v)
    return patterns

def daily_load_penalty_values(T: int, max_pairs_per_day: int) ->
List[int]:

    values: List[int] = []
    for count in range(T + 1):

```

```

        if count == 0:
            penalty = 1
        elif count == 1:
            penalty = 10
        elif count in (2, 3):
            penalty = 0
        elif count == 4:
            penalty = 6
        else:
            penalty = 12 + 4 * max(0, count - max_pairs_per_day)
        values.append(penalty)
    return values

def solve_schedule(
    occs: List[Occurrence],
    rooms: List[Room],
    days: List[str],
    times: List[str],
    max_pairs_per_day: int,
    military_days_by_group: Dict[str, str],
    teacher_pref_map: Dict[str, Dict[Tuple[str, str], str]],
    group_shift_map: Dict[str, str],
    time_limit_sec: int = 30,
) -> List[dict]:
    available_slots, slot_index, local_rooms, allowed, base_to_occs,
    preference_scores = build_solver_artifacts(
        occs, rooms, days, times, military_days_by_group,
        teacher_pref_map, group_shift_map,
        use_military=True, use_teacher_unavailable=True,
        use_shift=True
    )

    model = cp_model.CpModel()
    y: Dict[Tuple[int, int, int], cp_model.IntVar] = {}
    preference_terms: List[cp_model.LinearExpr] = []
    balance_penalty_terms: List[cp_model.IntVar] = []

    for i, o in enumerate(occs):
        if not allowed[i]:
            raise NoFeasibleOptions(o, diagnose_no_options(o,
                local_rooms, available_slots, military_days_by_group,
                teacher_pref_map, group_shift_map))
        for (s_idx, r_idx) in allowed[i]:
            var = model.NewBoolVar(f"y_o{i}_s{s_idx}_r{r_idx}")
            y[(i, s_idx, r_idx)] = var
            pref_score = preference_scores.get((i, s_idx, r_idx), 0)
            if pref_score > 0:
                preference_terms.append(pref_score * var)
            model.Add(sum(y[(i, s_idx, r_idx)] for (s_idx, r_idx) in
                allowed[i]) == 1)

    all_groups = sorted({g for o in occs for g in o.groups})

```

```

all_teachers = sorted({t for o in occs for t in o.teachers})

def occ_in_slot(i: int, s_idx: int) -> List[cp_model.IntVar]:
    return [y[(i, s_idx, r_idx)] for (ss, r_idx) in allowed[i]
if ss == s_idx]

for g in all_groups:
    occ_ids = [i for i, o in enumerate(occs) if g in o.groups]
    for s_idx in range(len(available_slots)):
        vars_ = []
        for i in occ_ids:
            vars_.extend(occ_in_slot(i, s_idx))
        if vars_:
            model.Add(sum(vars_) <= 1)

for t in all_teachers:
    occ_ids = [i for i, o in enumerate(occs) if t in o.teachers]
    for s_idx in range(len(available_slots)):
        vars_ = []
        for i in occ_ids:
            vars_.extend(occ_in_slot(i, s_idx))
        if vars_:
            model.Add(sum(vars_) <= 1)

for r_idx in range(len(local_rooms)):
    for s_idx in range(len(available_slots)):
        vars_ = [y[(i, s_idx, r_idx)] for i in range(len(occs))
if (i, s_idx, r_idx) in y]
        if vars_:
            model.Add(sum(vars_) <= 1)

for base_id, idxs in base_to_occs.items():
    if len(idxs) != 2:
        continue
    iA, iB = idxs
    oA, oB = occs[iA], occs[iB]
    if oA.allowed_weeks != ["A"] or oB.allowed_weeks != ["B"]:
        continue
    for d in days:
        for t in times:
            sA = slot_index.get(("A", d, t))
            sB = slot_index.get(("B", d, t))
            if sA is None or sB is None:
                continue
            model.Add(sum(occ_in_slot(iA, sA)) ==
sum(occ_in_slot(iB, sB)))
        for r_idx in range(len(local_rooms)):
            varsA = [y[(iA, s_idx, r_idx)] for (s_idx, rr) in
allowed[iA] if rr == r_idx]
            varsB = [y[(iB, s_idx, r_idx)] for (s_idx, rr) in
allowed[iB] if rr == r_idx]
            if varsA or varsB:
                model.Add(sum(varsA) == sum(varsB))

```

```

patterns_cache: Dict[int, List[List[int]]] = {}
group_to_occ_ids = {g: [i for i, o in enumerate(occs) if g in
o.groups] for g in all_groups}
occ_var: Dict[Tuple[str, int], cp_model.IntVar] = {}

for g in all_groups:
    occ_ids = group_to_occ_ids[g]
    for s_idx in range(len(available_slots)):
        v = model.NewBoolVar(f"occ_{g}_{s_idx}")
        vars_ = []
        for i in occ_ids:
            vars_.extend(occ_in_slot(i, s_idx))
        if vars_:
            model.Add(sum(vars_) == v)
        else:
            model.Add(v == 0)
        occ_var[(g, s_idx)] = v

penalty_values_cache: Dict[int, List[int]] = {}
for g in all_groups:
    for w in WEEKS:
        for d in days:
            s_list = [slot_index[(w, d, t)] for t in times]
            vec = [occ_var[(g, s_idx)] for s_idx in s_list]
            T = len(vec)
            if T not in patterns_cache:
                patterns_cache[T] = contiguous_patterns(T,
max_pairs_per_day)
            model.AddAllowedAssignments(vec, patterns_cache[T])

            if military_days_by_group.get(g, "") == d:
                continue
            count_var = model.NewIntVar(0, T,
f"daily_count_{g}_{w}_{d}")
            model.Add(count_var == sum(vec))
            if T not in penalty_values_cache:
                penalty_values_cache[T] =
daily_load_penalty_values(T, max_pairs_per_day)
            penalty_var = model.NewIntVar(0,
max(penalty_values_cache[T]), f"daily_penalty_{g}_{w}_{d}")
            model.AddElement(count_var, penalty_values_cache[T],
penalty_var)
            balance_penalty_terms.append(penalty_var)

if preference_terms or balance_penalty_terms:

    model.Maximize(
        sum(preference_terms)
        - 50 * sum(balance_penalty_terms)
    )

```

```

solver = cp_model.CpSolver()
solver.parameters.max_time_in_seconds = float(time_limit_sec)
solver.parameters.num_search_workers = 8
status = solver.Solve(model)
if status not in (cp_model.OPTIMAL, cp_model.FEASIBLE):
    diag = build_infeasibility_diagnostics(
        occs=occs,
        rooms=local_rooms,
        days=days,
        times=effective_times,
        military_days_by_group=military_days_by_group,
        teacher_pref_map=teacher_pref_map,
        group_shift_map=group_shift_map,
        max_pairs_per_day=max_pairs_per_day,
        time_limit_sec=time_limit_sec,
        allowed=allowed,
    )
    raise NoScheduleFound("Не удалось составить расписание:
ограничения противоречат друг другу.", diag=diag)

assignments: List[dict] = []
for i, o in enumerate(occs):
    chosen = None
    for (s_idx, r_idx) in allowed[i]:
        if solver.Value(y[(i, s_idx, r_idx)]) == 1:
            chosen = (s_idx, r_idx)
            break
    if chosen is None:
        continue
    s = available_slots[chosen[0]]
    r = local_rooms[chosen[1]]
    text = format_assignment_text(o.discipline, o.kind,
o.teachers, r.room)
    assignments.append(
        {
            "occ_id": o.occ_id,
            "base_id": o.base_id,
            "week": s.week,
            "day": s.day,
            "time": s.time,
            "discipline": o.discipline,
            "kind": o.kind,
            "teachers": ", ".join(o.teachers),
            "room": r.room,
            "groups": o.groups,
            "text": text,
        }
    )

return assignments

```

```

def normalize_day_label(day: str, days: List[str]) -> str:
    day = norm_str(day).upper()
    mapping = {
        "ПОНЕДЕЛЬНИК": "ПН",
        "ВТОРНИК": "ВТ",
        "СРЕДА": "СР",
        "ЧЕТВЕРГ": "ЧТ",
        "ПЯТНИЦА": "ПТ",
        "СУББОТА": "СБ",
        "ПН": "ПН",
        "ВТ": "ВТ",
        "СР": "СР",
        "ЧТ": "ЧТ",
        "ПТ": "ПТ",
        "СБ": "СБ",
    }
    short = mapping.get(day, day)
    return short if short in days else ""

def build_military_day_map(groups_df: pd.DataFrame, course_defaults:
Dict[int, str], editor_df: pd.DataFrame, days: List[str]) ->
Dict[str, str]:
    result: Dict[str, str] = {}
    for _, row in groups_df.iterrows():
        g = norm_str(row.get("group"))
        c = int(pd.to_numeric(row.get("course"), errors="coerce") or
0)
        default_day = normalize_day_label(course_defaults.get(c,
""), days)
        if g:
            result[g] = default_day

    if editor_df is not None and not editor_df.empty:
        for _, row in editor_df.iterrows():
            g = norm_str(row.get("group"))
            d = normalize_day_label(row.get("military_day"), days)
            if g:
                result[g] = d

    return result

def build_shift_map(groups_df: pd.DataFrame, course_shift_choices:
Dict[int, str]) -> Dict[str, str]:
    result: Dict[str, str] = {}
    for _, row in groups_df.iterrows():
        g = norm_str(row.get("group"))
        c = int(pd.to_numeric(row.get("course"), errors="coerce") or
0)
        if g:
            result[g] = course_shift_choices.get(c, "Не задавать")
    return result

```

```

def slot_allowed_by_shift(group_shift_map: Dict[str, str], groups:
List[str], day: str, time: str) -> bool:
    threshold = time_to_minutes("11:40")
    current = time_to_minutes(time)
    for g in groups:
        shift = group_shift_map.get(g, "Не задавать")
        if shift == "2 смена" and day not in {"ПН", "СБ"} and
current < threshold:
            return False
    return True

def slot_shift_preference_score(group_shift_map: Dict[str, str],
groups: List[str], day: str, time: str) -> int:
    score = 0
    current = time_to_minutes(time)
    earliest = {time_to_minutes("08:00"), time_to_minutes("09:50")}
    second_threshold = time_to_minutes("11:40")
    for g in groups:
        shift = group_shift_map.get(g, "Не задавать")
        if shift == "1 смена":
            if current in earliest:
                score += 2
        elif shift == "2 смена":
            if current >= second_threshold:
                score += 1
    return score

def extract_teacher_list(acts_df: pd.DataFrame) -> List[str]:
    teachers = set()
    teacher_col = "teacher" if "teacher" in acts_df.columns else
None
    if teacher_col is None:
        return []
    for value in acts_df[teacher_col].tolist():
        for t in split_teachers(value):
            teachers.add(t)
    return sorted(teachers)

def init_teacher_pref_store() -> None:
    if "teacher_pref_store" not in st.session_state:
        st.session_state.teacher_pref_store = {}

def get_teacher_pref_df(teacher: str, days: List[str], times:
List[str]) -> pd.DataFrame:
    store = st.session_state.teacher_pref_store
    teacher_map = store.get(teacher, {})
    rows = []

```

```

    for time in times:
        row = {"Папа": time}
        for day in days:
            row[day] = code_to_pref(teacher_map.get((day, time),
"neutral"))
            rows.append(row)
    return pd.DataFrame(rows)

def save_teacher_pref_df(teacher: str, df: pd.DataFrame, days:
List[str], times: List[str]) -> None:
    teacher_map: Dict[Tuple[str, str], str] = {}
    for _, row in df.iterrows():
        time = norm_str(row.get("Папа"))
        if time not in times:
            continue
        for day in days:
            teacher_map[(day, time)] = pref_to_code(row.get(day))
    st.session_state.teacher_pref_store[teacher] = teacher_map

def clear_teacher_preferences(teacher: str, days: List[str], times:
List[str]) -> None:
    st.session_state.teacher_pref_store[teacher] = {(day, time):
"neutral" for day in days for time in times}

def build_teacher_pref_map(days: List[str], times: List[str]) ->
Dict[str, Dict[Tuple[str, str], str]]:
    result: Dict[str, Dict[Tuple[str, str], str]] = {}
    for teacher, teacher_map in
st.session_state.teacher_pref_store.items():
        filtered = {}
        for (day, time), code in teacher_map.items():
            if day in days and time in times and code in
{"preferred", "unavailable"}:
                filtered[(day, time)] = code
        if filtered:
            result[teacher] = filtered
    return result

def summarize_teacher_preferences(teacher_pref_map: Dict[str,
Dict[Tuple[str, str], str]]) -> pd.DataFrame:
    rows = []
    for teacher, pref_map in sorted(teacher_pref_map.items()):
        preferred = sorted([f"{day} {time}" for (day, time), code in
pref_map.items() if code == "preferred"])
        unavailable = sorted([f"{day} {time}" for (day, time), code
in pref_map.items() if code == "unavailable"])
        rows.append(
            {
                "teacher": teacher,

```

```

        "предпочтительно": ", ".join(preferred),
        "недоступно": ", ".join(unavailable),
    }
)
return pd.DataFrame(rows)

def write_cell(ws, row: int, col: int, value, *, bold=False,
fill=None, center=False, wrap=True):
    cell = ws.cell(row, col, value)
    cell.font = Font(bold=bold)
    cell.alignment = Alignment(horizontal="center" if center else
"left", vertical="center", wrap_text=wrap)
    thin = Side(style="thin", color="000000")
    cell.border = Border(left=thin, right=thin, top=thin,
bottom=thin)
    if fill is not None:
        cell.fill = fill
    return cell

def set_medium_day_separator(ws, row: int, max_col: int):
    medium = Side(style="medium", color="000000")
    thin = Side(style="thin", color="000000")
    for c in range(1, max_col + 1):
        cell = ws.cell(row, c)
        cell.border = Border(
            left=cell.border.left if cell.border and
cell.border.left and cell.border.left.style else thin,
            right=cell.border.right if cell.border and
cell.border.right and cell.border.right.style else thin,
            top=cell.border.top if cell.border and cell.border.top
and cell.border.top.style else thin,
            bottom=medium,
        )

def course_sheet_layout(
    ws,
    course: int,
    course_groups: List[str],
    days: List[str],
    times: List[str],
    data_map: Dict[Tuple[str, str, str], Dict[str, str]],
    military_days_by_group: Dict[str, str],
):
    header_fill = PatternFill("solid", fgColor="D9EAF7")
    day_fill = PatternFill("solid", fgColor="E2F0D9")
    week_fill = PatternFill("solid", fgColor="FCE4D6")
    time_fill = PatternFill("solid", fgColor="E2F0D9")
    military_fill = PatternFill("solid", fgColor="FFF2CC")

    row = 1

```

```

        write_cell(ws, row, 1, "День", bold=True, fill=header_fill,
center=True)
        write_cell(ws, row, 2, "Время", bold=True, fill=header_fill,
center=True)
        write_cell(ws, row, 3, "Неделя", bold=True, fill=header_fill,
center=True)
        col = 4
        for g in course_groups:
            write_cell(ws, row, col, g, bold=True, fill=header_fill,
center=True)
            col += 1

        row += 1
        max_col = 3 + len(course_groups)
        for day in days:
            for time_idx, time in enumerate(times):
                row_a = row
                row_b = row + 1

                write_cell(ws, row_a, 1, day, fill=day_fill,
center=True)
                write_cell(ws, row_a, 2, time, fill=time_fill,
center=True)
                ws.merge_cells(start_row=row_a, start_column=1,
end_row=row_b, end_column=1)
                ws.merge_cells(start_row=row_a, start_column=2,
end_row=row_b, end_column=2)

                write_cell(ws, row_a, 3, "A", fill=week_fill,
center=True)
                write_cell(ws, row_b, 3, "B", fill=week_fill,
center=True)

                col = 4
                for g in course_groups:
                    military_day = military_days_by_group.get(g, "")
                    if military_day == day:
                        text_a = MILITARY_TEXT
                        text_b = MILITARY_TEXT
                        fill = military_fill
                    else:
                        text_a = data_map.get((g, day, time),
{ }).get("A", "")
                        text_b = data_map.get((g, day, time),
{ }).get("B", "")
                        fill = None

                    if text_a and text_a == text_b:
                        write_cell(ws, row_a, col, text_a, fill=fill)
                        write_cell(ws, row_b, col, None, fill=fill)
                        ws.merge_cells(start_row=row_a,
start_column=col, end_row=row_b, end_column=col)
                    else:

```

```

        write_cell(ws, row_a, col, text_a, fill=fill)
        write_cell(ws, row_b, col, text_b, fill=fill)
    col += 1

    if time_idx == len(times) - 1:
        set_medium_day_separator(ws, row_b, max_col)

    row += 2

for c in range(1, ws.max_column + 1):
    if c == 1:
        ws.column_dimensions[get_column_letter(c)].width = 9
    elif c == 2:
        ws.column_dimensions[get_column_letter(c)].width = 10
    elif c == 3:
        ws.column_dimensions[get_column_letter(c)].width = 9
    else:
        ws.column_dimensions[get_column_letter(c)].width = 26

for r in range(1, ws.max_row + 1):
    if r == 1:
        ws.row_dimensions[r].height = 24
    elif r % 2 == 0:
        ws.row_dimensions[r].height = 52
    else:
        ws.row_dimensions[r].height = 28

ws.freeze_panes = "D2"

def build_output_workbook(
    assignments: List[dict],
    groups_df: pd.DataFrame,
    days: List[str],
    solver_times: List[str],
    military_days_by_group: Dict[str, str],
    group_shift_map: Dict[str, str],
) -> BytesIO:
    wb = Workbook()
    ws_all = wb.active
    ws_all.title = "Все назначения"

    all_headers = ["Группа", "Курс", "Неделя", "День", "Время",
"Дисциплина", "Тип", "Преподаватели", "Аудитория"]
    for i, h in enumerate(all_headers, start=1):
        write_cell(ws_all, 1, i, h, bold=True,
fill=PatternFill("solid", fgColor="EDEDDED"), center=True)

    group_to_course = build_group_to_course_from_sheet(groups_df)

    flat_rows = []
    for a in assignments:
        for g in a["groups"]:

```

```

        flat_rows.append(
            {
                "group": g,
                "course": group_to_course.get(g, 0),
                "week": a["week"],
                "day": a["day"],
                "time": a["time"],
                "discipline": a["discipline"],
                "kind": kind_to_short(a["kind"]),
                "teachers": a["teachers"],
                "room": a["room"],
                "text": a["text"],
            }
        )

    day_order = {d: i for i, d in enumerate(days)}
    time_order = {t: i for i, t in enumerate(solver_times)}
    flat_rows = sorted(flat_rows, key=lambda x: (x["course"],
x["group"], day_order.get(x["day"], 99), time_order.get(x["time"],
99), x["week"]))

    r = 2
    for item in flat_rows:
        vals = [item["group"], item["course"], item["week"],
item["day"], item["time"], item["discipline"], item["kind"],
item["teachers"], item["room"]]
        for c, v in enumerate(vals, start=1):
            write_cell(ws_all, r, c, v)
        r += 1
    ws_all.freeze_panes = "A2"
    for c in range(1, len(all_headers) + 1):
        ws_all.column_dimensions[get_column_letter(c)].width = 20

    data_map: Dict[Tuple[str, str, str], Dict[str, str]] = {}
    for item in flat_rows:
        data_map.setdefault((item["group"], item["day"],
item["time"]), {})[item["week"]] = item["text"]

    course_to_groups: Dict[int, List[str]] = {}
    for g, c in group_to_course.items():
        if g and c > 0:
            course_to_groups.setdefault(c, []).append(g)

    for course in sorted(course_to_groups.keys()):
        ws = wb.create_sheet(title=safe_sheet_name(f"{course}
курс"))
        course_groups = sorted(course_to_groups[course])
        course_times = export_times_for_course(course_groups,
group_shift_map, solver_times)
        course_sheet_layout(ws, course, course_groups, days,
course_times, data_map, military_days_by_group)

    bio = BytesIO()

```

```

wb.save(bio)
bio.seek(0)
return bio

def assignments_to_flat_df(
    assignments: List[dict],
    groups_df: pd.DataFrame,
    days: List[str],
    times: List[str],
) -> pd.DataFrame:
    group_to_course = build_group_to_course_from_sheet(groups_df)
    rows = []
    for a in assignments:
        for g in a.get("groups", []):
            rows.append({
                "occ_id": a.get("occ_id", ""),
                "Группа": g,
                "Курс": group_to_course.get(g, 0),
                "Неделя": a.get("week", ""),
                "День": a.get("day", ""),
                "Время": a.get("time", ""),
                "Дисциплина": a.get("discipline", ""),
                "Тип": kind_to_short(a.get("kind", "")),
                "Преподаватели": a.get("teachers", ""),
                "Аудитория": a.get("room", ""),
            })
    if not rows:
        return pd.DataFrame()
    df = pd.DataFrame(rows)
    day_order = {d: i for i, d in enumerate(days)}
    time_order = {t: i for i, t in enumerate(times)}
    df["_day_order"] = df["День"].map(day_order).fillna(99)
    df["_time_order"] = df["Время"].map(time_order).fillna(99)
    df = df.sort_values(["Курс", "Группа", "_day_order",
        "_time_order", "Неделя"]).drop(columns=["_day_order",
        "_time_order"])
    return df.reset_index(drop=True)

def build_course_preview_df(
    assignments: List[dict],
    groups_df: pd.DataFrame,
    course: int,
    days: List[str],
    times: List[str],
    military_days_by_group: Dict[str, str],
) -> pd.DataFrame:
    group_to_course = build_group_to_course_from_sheet(groups_df)
    course_groups = sorted([g for g, c in group_to_course.items() if
        c == course])
    data_map: Dict[Tuple[str, str, str], Dict[str, str]] = {}
    for a in assignments:

```

```

        for g in a.get("groups", []):
            data_map.setdefault((g, a.get("day", ""), a.get("time",
"")), {})[a.get("week", "")] = a.get("text", "")

    rows = []
    for day in days:
        for time in times:
            for week in WEEKS:
                row = {"День": day, "Время": time, "Неделя": week}
                for g in course_groups:
                    if military_days_by_group.get(g, "") == day:
                        row[g] = MILITARY_TEXT
                    else:
                        row[g] = data_map.get((g, day, time),
{}).get(week, "")
                rows.append(row)
    return pd.DataFrame(rows)

def render_course_preview_html(
    assignments: List[dict],
    groups_df: pd.DataFrame,
    course: int,
    days: List[str],
    times: List[str],
    military_days_by_group: Dict[str, str],
) -> None:

    group_to_course = build_group_to_course_from_sheet(groups_df)
    course_groups = sorted([g for g, c in group_to_course.items() if
c == course])

    data_map: Dict[Tuple[str, str, str], Dict[str, str]] = {}
    for a in assignments:
        for g in a.get("groups", []):
            data_map.setdefault((g, a.get("day", ""), a.get("time",
"")), {})[a.get("week", "")] = a.get("text", "")

    def cell_text(value: str) -> str:
        value = norm_str(value)
        if not value:
            return ""
        return html.escape(value).replace("\n", "<br>")

    header_cells = "".join(f"<th class='group-
col'>{html.escape(g)}</th>" for g in course_groups)
    body = []
    for day in days:
        for time in times:
            values_by_week: Dict[str, Dict[str, str]] = {"A": {},
"B": {}

            for week in WEEKS:
                for g in course_groups:

```

```

        if military_days_by_group.get(g, "") == day:
            values_by_week[week][g] = MILITARY_TEXT
        else:
            values_by_week[week][g] = data_map.get((g,
day, time), {}).get(week, "")

    first_row = [
        f"<td class='meta day-cell'
rowspan='2'>{html.escape(day)}</td>",
        f"<td class='meta time-cell'
rowspan='2'>{html.escape(time)}</td>",
        "<td class='meta week-cell'>A</td>",
    ]
    second_row = ["<td class='meta week-cell'>B</td>"]

    for g in course_groups:
        value_a = values_by_week["A"].get(g, "")
        value_b = values_by_week["B"].get(g, "")
        if value_a and value_a == value_b:
            first_row.append(f"<td class='lesson-cell
merged' rowspan='2'>{cell_text(value_a)}</td>")
        else:
            first_row.append(f"<td class='lesson-
cell'>{cell_text(value_a)}</td>")
            second_row.append(f"<td class='lesson-
cell'>{cell_text(value_b)}</td>")

        body.append("<tr>" + "".join(first_row) + "</tr>")
        body.append("<tr class='slot-bottom'>" +
"".join(second_row) + "</tr>")
        if body:
            body[-1] = body[-1].replace("class='slot-bottom'",
"class='slot-bottom day-bottom'", 1)

    table_html = f"""
<style>
.course-schedule-wrapper {{
    width: 100%;
    max-height: 560px;
    overflow: auto;
    border: 1px solid #e5e7eb;
    border-radius: 10px;
    background: white;
}}
table.course-schedule {{
    border-collapse: collapse;
    width: max-content;
    min-width: 100%;
    table-layout: fixed;
    font-size: 13px;
}}
table.course-schedule th,
table.course-schedule td {{

```

```

        border: 1px solid #e5e7eb;
        padding: 8px 10px;
        vertical-align: middle;
        overflow: hidden;
        text-overflow: ellipsis;
    }}
    table.course-schedule th {{
        position: sticky;
        top: 0;
        z-index: 2;
        background: #f8fafc;
        font-weight: 600;
        text-align: center;
        white-space: nowrap;
    }}
    table.course-schedule .meta {{
        background: #f8fafc;
        text-align: center;
        white-space: nowrap;
    }}
    table.course-schedule .day-cell {{ width: 54px; min-width: 54px;
}}
    table.course-schedule .time-cell {{ width: 72px; min-width:
72px; }}
    table.course-schedule .week-cell {{ width: 58px; min-width:
58px; }}
    table.course-schedule .group-col,
    table.course-schedule .lesson-cell {{
        width: 230px;
        min-width: 230px;
        max-width: 230px;
    }}
    table.course-schedule .lesson-cell {{
        height: 48px;
        line-height: 1.25;
    }}
    table.course-schedule .merged {{
        background: #ffffff;
    }}
    table.course-schedule .day-bottom td {{
        border-bottom: 2px solid #cbd5e1;
    }}
</style>
<div class="course-schedule-wrapper">
    <table class="course-schedule">
        <thead>
<tr><th>День</th><th>Время</th><th>Неделя</th>{header_cells}</tr>
        </thead>
        <tbody>{' '.join(body)}</tbody>
    </table>
</div>
"""

```

```

st.markdown(table_html, unsafe_allow_html=True)

def selected_assignment_ids_for_move(selected_occ_id: str,
assignments: List[dict], occs: List[Occurrence]) -> List[str]:

    occs_by_id = occ_lookup(occs)
    assigns_by_id = assignment_lookup(assignments)
    selected_assignment = assigns_by_id.get(selected_occ_id)
    if selected_assignment is None:
        return [selected_occ_id]

    same_base = [a for a in assignments if a.get("base_id") ==
selected_assignment.get("base_id")]
    synced_pair = (
        len(same_base) == 2
        and {a.get("week") for a in same_base} == {"A", "B"}
        and all(occs_by_id.get(a.get("occ_id")) and
occs_by_id[a.get("occ_id")].allowed_weeks in (["A"], ["B"]) for a in
same_base)
    )
    return [a.get("occ_id") for a in same_base] if synced_pair else
[selected_occ_id]

def move_mode_hint(selected_occ_id: str, assignments: List[dict],
occs: List[Occurrence]) -> str:
    selected_ids = selected_assignment_ids_for_move(selected_occ_id,
assignments, occs)
    if len(selected_ids) == 2:
        return "Эта пара проводится каждую неделю. При переносе
недели А и В будут перенесены вместе на один день, время и
аудиторию."
    selected_assignment =
assignment_lookup(assignments).get(selected_occ_id, {})
    return f"Эта пара проводится только на неделе
{selected_assignment.get('week', '')}. При переносе она останется
только на этой неделе."

def assignment_label(a: dict) -> str:
    groups = ", ".join(a.get("groups", []))
    return f"{a.get('occ_id', '')} | {a.get('week', '')}
{a.get('day', '')} {a.get('time', '')} | {groups} |
{a.get('discipline', '')} | ауд. {a.get('room', '')}"

def assignment_lookup(assignments: List[dict]) -> Dict[str, dict]:
    return {a.get("occ_id", ""): a for a in assignments if
a.get("occ_id")}

def room_lookup(rooms: List[Room]) -> Dict[str, Room]:

```

```

    return {r.room: r for r in rooms}

def occ_lookup(occs: List[Occurrence]) -> Dict[str, Occurrence]:
    return {o.occ_id: o for o in occs}

def same_slot(a: dict, b: dict) -> bool:
    return a.get("week") == b.get("week") and a.get("day") ==
b.get("day") and a.get("time") == b.get("time")

def current_assignments_have_no_conflicts(
    assignments: List[dict],
    days: List[str],
    times: List[str],
    max_pairs_per_day: int,
) -> bool:

    seen_group = set()
    seen_teacher = set()
    seen_room = set()

    for a in assignments:
        slot_key = (a.get("week"), a.get("day"), a.get("time"))
        for g in a.get("groups", []):
            key = (g, *slot_key)
            if key in seen_group:
                return False
            seen_group.add(key)
        for t in split_teachers(a.get("teachers", "")):
            key = (t, *slot_key)
            if key in seen_teacher:
                return False
            seen_teacher.add(key)
        room = norm_str(a.get("room"))
        if room:
            key = (room, *slot_key)
            if key in seen_room:
                return False
            seen_room.add(key)

    all_groups = sorted({g for a in assignments for g in
a.get("groups", [])})
    time_index = {t: i for i, t in enumerate(times)}
    allowed_patterns = contiguous_patterns(len(times),
max_pairs_per_day)
    allowed_pattern_set = {tuple(p) for p in allowed_patterns}

    for g in all_groups:
        for week in WEEKS:
            for day in days:
                vec = [0] * len(times)

```

```

        for a in assignments:
            if g in a.get("groups", []) and a.get("week") ==
week and a.get("day") == day:
                idx = time_index.get(a.get("time"))
                if idx is not None:
                    vec[idx] = 1
            if tuple(vec) not in allowed_pattern_set:
                return False

    return True

def replace_assignment_slot(
    assignments: List[dict],
    selected_ids: List[str],
    day: str,
    time: str,
    room: str,
    occs_by_id: Dict[str, Occurrence],
) -> List[dict]:
    new_assignments = []
    for a in assignments:
        if a.get("occ_id") in selected_ids:
            occ = occs_by_id[a.get("occ_id")]
            updated = dict(a)
            updated["day"] = day
            updated["time"] = time
            updated["room"] = room
            updated["text"] =
format_assignment_text(updated.get("discipline", ""),
updated.get("kind", ""), occ.teachers, room)
            updated["teachers"] = ", ".join(occ.teachers)
            new_assignments.append(updated)
        else:
            new_assignments.append(dict(a))
    return new_assignments

def find_move_options(
    selected_occ_id: str,
    assignments: List[dict],
    occs: List[Occurrence],
    rooms: List[Room],
    days: List[str],
    times: List[str],
    military_days_by_group: Dict[str, str],
    teacher_pref_map: Dict[str, Dict[Tuple[str, str], str]],
    group_shift_map: Dict[str, str],
    max_pairs_per_day: int,
) -> List[dict]:
    occs_by_id = occ_lookup(occs)
    assigns_by_id = assignment_lookup(assignments)
    selected_occ = occs_by_id.get(selected_occ_id)
    selected_assignment = assigns_by_id.get(selected_occ_id)

```

```

    if selected_occ is None or selected_assignment is None:
        return []

    available_slots, _, local_rooms, allowed, _, preference_scores =
build_solver_artifacts(
    occs, rooms, days, times, military_days_by_group,
teacher_pref_map, group_shift_map,
    use_military=True, use_teacher_unavailable=True,
use_shift=True,
)
    room_by_idx = {idx: r for idx, r in enumerate(local_rooms)}
    occ_index_by_id = {o.occ_id: i for i, o in enumerate(occs)}

    same_base = [a for a in assignments if a.get("base_id") ==
selected_assignment.get("base_id")]
    synced_pair = (
        len(same_base) == 2
        and {a.get("week") for a in same_base} == {"A", "B"}
        and all(occs_by_id.get(a.get("occ_id")) and
occs_by_id[a.get("occ_id")].allowed_weeks in (["A"], ["B"]) for a in
same_base)
    )
    selected_ids = [a.get("occ_id") for a in same_base] if
synced_pair else [selected_occ_id]

    options = []
    seen = set()

    if synced_pair:

        pair_ids = selected_ids
        pair_occ_indices = [occ_index_by_id[oid] for oid in
pair_ids]
        for day in days:
            for time in times:
                for room in local_rooms:
                    if all(
                        any(
                            available_slots[s_idx].week ==
assigns_by_id[oid].get("week")
                            and available_slots[s_idx].day == day
                            and available_slots[s_idx].time == time
                            and room_by_idx[r_idx].room == room.room
                            for s_idx, r_idx in
allowed.get(occ_index_by_id[oid], [])
                        )
                        for oid in pair_ids
                    ):
                        if all(assigns_by_id[oid].get("day") == day
and assigns_by_id[oid].get("time") == time and
assigns_by_id[oid].get("room") == room.room for oid in pair_ids):
                            continue

```

```

        candidate =
replace_assignment_slot(assignments, pair_ids, day, time, room.room,
occs_by_id)

        if not
current_assignments_have_no_conflicts(candidate, days, times,
max_pairs_per_day):
            continue
        score = 0
        for oid in pair_ids:
            i = occ_index_by_id[oid]
            week = assigns_by_id[oid].get("week")
            for s_idx, r_idx in allowed.get(i, []):
                s = available_slots[s_idx]
                if s.week == week and s.day == day
and s.time == time and room_by_idx[r_idx].room == room.room:
                    score +=
preference_scores.get((i, s_idx, r_idx), 0)
                    key = (day, time, room.room)
                    if key not in seen:
                        seen.add(key)
                        options.append({"day": day, "time":
time, "room": room.room, "selected_ids": pair_ids, "score": score,
"mode": "A/B вместе"})
                    else:
                        i = occ_index_by_id[selected_occ_id]
                        for s_idx, r_idx in allowed.get(i, []):
                            slot = available_slots[s_idx]
                            room = room_by_idx[r_idx]
                            if slot.week != selected_assignment.get("week"):
                                continue
                            if selected_assignment.get("day") == slot.day and
selected_assignment.get("time") == slot.time and
selected_assignment.get("room") == room.room:
                                continue
                            candidate = replace_assignment_slot(assignments,
[selected_occ_id], slot.day, slot.time, room.room, occs_by_id)
                            if not current_assignments_have_no_conflicts(candidate,
days, times, max_pairs_per_day):
                                continue
                            key = (slot.week, slot.day, slot.time, room.room)
                            if key not in seen:
                                seen.add(key)
                                options.append({
                                    "day": slot.day,
                                    "time": slot.time,
                                    "room": room.room,
                                    "selected_ids": [selected_occ_id],
                                    "score": preference_scores.get((i, s_idx,
r_idx), 0),
                                    "mode": "одна пара",
                                })

        day_order = {d: n for n, d in enumerate(days)}

```

```

        time_order = {t: n for n, t in enumerate(times)}
        options.sort(key=lambda x: (-x.get("score", 0),
day_order.get(x["day"], 99), time_order.get(x["time"], 99),
x["room"])))
        return options

def option_label(option: dict) -> str:
    return f"{option['day']} {option['time']} | ауд.
{option['room']} | режим: {option.get('mode', 'перенос')}}"

def render_schedule_preview_and_editor() -> None:
    assignments = st.session_state.get("schedule_assignments", [])
    context = st.session_state.get("schedule_context", {})
    occs = st.session_state.get("schedule_occs", [])
    if not assignments or not context:
        st.info("Сначала составьте расписание.")
        return

    groups_df = context["groups_df"]
    days = context["days"]
    times = context["times"]
    rooms = context["rooms"]
    military_days_by_group = context["military_days_by_group"]
    teacher_pref_map = context["teacher_pref_map"]
    group_shift_map = context["group_shift_map"]
    max_pairs_per_day = context["max_pairs_per_day"]

    if st.session_state.get("move_success_message"):
        st.success(st.session_state.move_success_message)
        st.session_state.move_success_message = ""

    flat_df = assignments_to_flat_df(assignments, groups_df, days,
times)
    group_to_course = build_group_to_course_from_sheet(groups_df)
    courses = sorted({c for c in group_to_course.values() if c > 0})

    def render_move_panel(selected_occ_id: str, panel_key: str) ->
None:
        selected_assignment =
assignment_lookup(assignments).get(selected_occ_id)
        if not selected_assignment:
            return

        st.markdown("---")
        st.markdown("***Перенос выбранной пары**")
        st.caption(
            f"Сейчас: {selected_assignment.get('week', '')}
{selected_assignment.get('day', '')} "
            f"{selected_assignment.get('time', '')}, ауд.
{selected_assignment.get('room', '')}. "

```

```

        f"{'', ' '.join(selected_assignment.get('groups', []))} -
{selected_assignment.get('discipline', '')}"
    )
    st.caption(move_mode_hint(selected_occ_id, assignments,
occs))

    if st.button("Найти варианты переноса",
key=f"find_move_options_{panel_key}_{selected_occ_id}"):
        options = find_move_options(
            selected_occ_id=selected_occ_id,
            assignments=assignments,
            occs=occs,
            rooms=rooms,
            days=days,
            times=times,
            military_days_by_group=military_days_by_group,
            teacher_pref_map=teacher_pref_map,
            group_shift_map=group_shift_map,
            max_pairs_per_day=max_pairs_per_day,
        )
        st.session_state.move_options = options
        st.session_state.move_options_for = selected_occ_id

        options = st.session_state.get("move_options", []) if
st.session_state.get("move_options_for") == selected_occ_id else []
        if options:
            option_labels = [option_label(o) for o in options]
            selected_option_label = st.selectbox("Подходящие
варианты", options=option_labels,
key=f"selected_move_option_{panel_key}_{selected_occ_id}")
            selected_option =
options[option_labels.index(selected_option_label)]
            if st.button("Перенести сюда",
key=f"apply_move_{panel_key}_{selected_occ_id}"):
                occs_by_id = occ_lookup(occs)
                updated_assignments = replace_assignment_slot(
                    assignments,
                    selected_option["selected_ids"],
                    selected_option["day"],
                    selected_option["time"],
                    selected_option["room"],
                    occs_by_id,
                )
                st.session_state.schedule_assignments =
updated_assignments
                move_message = "Пара перенесена. Таблица, файл для
скачивания и история SQLite обновлены."
                try:
                    save_schedule_to_history(updated_assignments,
context, title="Расписание после ручного переноса")
                except Exception as history_error:
                    move_message = f"Пара перенесена, но не удалось
сохранить историю SQLite: {history_error}"

```

```

        st.session_state.move_options = []
        st.session_state.move_options_for = ""
        st.session_state.reopen_schedule_dialog = True
        st.session_state.move_success_message = move_message
        st.rerun()
    elif st.session_state.get("move_options_for") ==
selected_occ_id:
        st.warning("Подходящих вариантов переноса для выбранной
пары не найдено.")

    tab_all, tab_courses = st.tabs(["Все назначения", "По курсам"])

    with tab_all:
        display_all_df = flat_df.drop(columns=["occ_id"],
errors="ignore")
        st.dataframe(display_all_df, use_container_width=True,
height=420)

        if not flat_df.empty:
            choices = {assignment_label(a): a.get("occ_id") for a in
assignments}
            selected_label = st.selectbox(
                "Пара для переноса",
                options=list(choices.keys()),
                key="move_selected_occ_label_all",
                help="Выберите пару из списка. Отдельной вкладки
переноса больше нет: перенос выполняется прямо из просмотра
расписания.",
            )
            render_move_panel(choices[selected_label], "all")

    with tab_courses:
        if not courses:
            st.info("В листе Groups не найдены курсы для
отображения.")
        else:
            selected_course = st.selectbox("Курс", options=courses,
format_func=lambda c: f"{c} курс", key="preview_course_select")
            render_course_preview_html(assignments, groups_df,
selected_course, days, times, military_days_by_group)
            st.caption("Если занятие одинаковое на неделях А и В,
оно показано одной объединённой ячейкой. При переносе такая пара
переносится сразу для обеих недель.")

            course_group_set = {g for g, c in
group_to_course.items() if c == selected_course}
            course_assignments = [
                a for a in assignments
                if any(g in course_group_set for g in
a.get("groups", []))
            ]
            if course_assignments:

```

```

        choices = {assignment_label(a): a.get("occ_id") for
a in course_assignments}
        selected_label = st.selectbox(
            "Па́ра для переноса из выбранного курса",
            options=list(choices.keys()),

key=f"move_selected_occ_label_course_{selected_course}",
            help="Выберите пару из текущего курса и найдите
допустимые варианты переноса.",
        )
        render_move_panel(choices[selected_label],
f"course_{selected_course}")

    out_bio = build_output_workbook(assignments, groups_df, days,
times, military_days_by_group, group_shift_map)
    st.download_button(
        label="Скачать текущее расписание (xlsx)",
        data=out_bio,
        file_name="raspisanie_by_course_edited.xlsx",
        mime="application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet",
        key="download_current_schedule_from_preview",
    )

if hasattr(st, "dialog"):
    @st.dialog("Просмотр и редактирование расписания",
width="large")
        def render_schedule_dialog():
            render_schedule_preview_and_editor()
else:
    def render_schedule_dialog():
        with st.expander("Просмотр и редактирование расписания",
expanded=True):
            render_schedule_preview_and_editor()

def init_schedule_history_db() -> None:

    with sqlite3.connect(SCHEDULE_HISTORY_DB) as conn:
        conn.execute(
            """
            CREATE TABLE IF NOT EXISTS schedule_history (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                created_at TEXT NOT NULL,
                title TEXT NOT NULL,
                assignments_count INTEGER NOT NULL,
                groups_count INTEGER NOT NULL,
                courses_count INTEGER NOT NULL,
                xlsx_data BLOB NOT NULL
            )
            """
        )

```

```

        conn.commit()

def prune_schedule_history(limit: int = SCHEDULE_HISTORY_LIMIT) ->
None:

    with sqlite3.connect(SCHEDULE_HISTORY_DB) as conn:
        conn.execute(
            """
            DELETE FROM schedule_history
            WHERE id NOT IN (
                SELECT id
                FROM schedule_history
                ORDER BY datetime(created_at) DESC, id DESC
                LIMIT ?
            )
            """,
            (int(limit),),
        )
        conn.commit()

def save_schedule_to_history(assignments: List[dict], context: dict,
title: str = "Составленное расписание") -> None:

    if not assignments or not context:
        return

    out_bio = build_output_workbook(
        assignments,
        context["groups_df"],
        context["days"],
        context["times"],
        context["military_days_by_group"],
        context["group_shift_map"],
    )
    xlsx_data = out_bio.getvalue()
    group_to_course =
build_group_to_course_from_sheet(context["groups_df"])
    groups_count = len(group_to_course)
    courses_count = len({c for c in group_to_course.values() if c >
0})

    init_schedule_history_db()
    with sqlite3.connect(SCHEDULE_HISTORY_DB) as conn:
        conn.execute(
            """
            INSERT INTO schedule_history
                (created_at, title, assignments_count, groups_count,
courses_count, xlsx_data)
            VALUES (?, ?, ?, ?, ?, ?)
            """,
            (

```

```

        datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        title,
        len(assignments),
        groups_count,
        courses_count,
        sqlite3.Binary(xlsx_data),
    ),
)
conn.commit()
prune_schedule_history()

def list_schedule_history() -> List[dict]:

    init_schedule_history_db()
    with sqlite3.connect(SCHEDULE_HISTORY_DB) as conn:
        conn.row_factory = sqlite3.Row
        rows = conn.execute(
            """
            SELECT id, created_at, title, assignments_count,
groups_count, courses_count, xlsx_data
            FROM schedule_history
            ORDER BY datetime(created_at) DESC, id DESC
            LIMIT ?
            """,
            (SCHEDULE_HISTORY_LIMIT,),
        ).fetchall()
    return [dict(row) for row in rows]

def render_schedule_history_sidebar() -> None:
    st.sidebar.markdown("----")
    st.sidebar.subheader("История")
    st.sidebar.caption(
        "В SQLite автоматически хранятся последние 10 составленных
расписаний. "
        "Когда появляется новое расписание, самое старое удаляется."
    )
    try:
        history_rows = list_schedule_history()
    except Exception as e:
        st.sidebar.warning(f"Не удалось открыть историю SQLite:
{e}")
    return

    if not history_rows:
        st.sidebar.caption("Пока сохранённых расписаний нет.")
        return

    for idx, row in enumerate(history_rows, start=1):
        label = f"{idx}. {row['created_at']}"
        with st.sidebar.expander(label):
            st.caption(

```

```

        f"{row['title']}\n\n"
        f"Назначений: {row['assignments_count']} · "
        f"Групп: {row['groups_count']} · "
        f"Курсов: {row['courses_count']}"
    )
    safe_created = str(row["created_at"]).replace(":", "-")
    ").replace(" ", "_")
    st.download_button(
        label="Скачать из истории",
        data=bytes(row["xlsx_data"]),
        file_name=f"raspisanie_history_{safe_created}.xlsx",
        mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
        key=f"download_history_schedule_{row['id']}",
    )

st.set_page_config(page_title="Генератор расписания кафедры",
layout="wide")
st.title("Генератор расписания кафедры – MVP v25")

st.markdown("""
<style>
div.stButton > button,
div.stDownloadButton > button {
    white-space: nowrap;
    min-height: 2.5rem;
}
</style>
""", unsafe_allow_html=True)

st.sidebar.header("Исходные данные")
st.sidebar.caption("Загрузите Excel-файл семестра с листами Groups, Activities и Rooms. Если файла ещё нет, можно скачать пустой шаблон ниже.")
st.sidebar.markdown("---")
st.sidebar.subheader("Шаблон")
st.sidebar.caption("Скачивание необязательно: можно загрузить уже заполненный файл. Первая строка после заголовков в каждом листе – пример, программа её пропускает.")
st.sidebar.download_button(
    label="Скачать шаблон Excel",
    data=build_input_template_workbook(),
    file_name="template_semester.xlsx",
    mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
    help="Шаблон содержит листы Groups, Activities и Rooms. Первая строка после заголовков – подсказка, она не учитывается при распределении.",
)
st.sidebar.markdown("---")
st.sidebar.subheader("Загрузка")

```

```

dyn_file = st.sidebar.file_uploader("Исходный файл семестра
(.xlsx)", type=["xlsx"], help="Файл с листами Groups, Activities и
Rooms. Лист Settings больше не нужен, внешний статический файл не
требуется.")
st.sidebar.markdown("---")
st.sidebar.subheader("Параметры расчёта")
time_limit = st.sidebar.slider("Время на поиск решения (сек)", 10,
120, 40, 10)
render_schedule_history_sidebar()

if dyn_file is None:
    st.caption("Загрузите исходный файл семестра, затем настройте
дополнительные параметры и нажмите «Составить расписание».")
    st.stop()

settings, groups_df, acts_df, rooms_df, rooms =
read_dynamic(dyn_file.getvalue())
days, base_times = parse_days_and_times(settings)
teachers = extract_teacher_list(acts_df)
init_teacher_pref_store()

col1, col2, col3 = st.columns(3)
with col1:
    st.metric("Групп", len(groups_df))
with col2:
    st.metric("Занятий", len(acts_df))
with col3:
    st.metric("Аудиторий", len(rooms))

st.subheader("Военная кафедра")
st.caption("Необязательная настройка. Можно задать день военной
подготовки сразу для целого курса и при необходимости изменить его
для отдельных групп.")

course_defaults: Dict[int, str] = {}
all_courses = sorted(pd.to_numeric(groups_df["course"],
errors="coerce").dropna().astype(int).unique().tolist())
cols = st.columns(min(4, max(1, len(all_courses))))
for idx, course in enumerate(all_courses):
    with cols[idx % len(cols)]:
        chosen = st.selectbox(
            f"{course} курс",
            options=["Нет"] + days,
            index=0,
            key=f"mil_course_{course}",
            help="Если выбран день, в этот день обычные пары для
групп курса ставиться не будут.",
        )
        course_defaults[course] = "" if chosen == "Нет" else chosen

editor_seed = groups_df[["group", "course"]].copy()
editor_seed["group"] = editor_seed["group"].astype(str).str.strip()

```

```

editor_seed["course"] = pd.to_numeric(editor_seed["course"],
errors="coerce").fillna(0).astype(int)
editor_seed["military_day"] = editor_seed["course"].map(lambda c:
course_defaults.get(int(c), ""))
editor_seed["military_day"] =
editor_seed["military_day"].replace({"": "Нет"})

editor_df = st.data_editor(
    editor_seed,
    key="military_editor",
    hide_index=True,
    use_container_width=True,
    column_config={
        "group": st.column_config.TextColumn("Группа",
disabled=True),
        "course": st.column_config.NumberColumn("Курс",
disabled=True),
        "military_day": st.column_config.SelectboxColumn("День
военной подготовки", options=["Нет"] + days),
    },
)

military_days_by_group = build_military_day_map(groups_df,
course_defaults, editor_df, days)
selected_military = {g: d for g, d in military_days_by_group.items()
if d}
if selected_military:
    with st.expander("Показать назначенные дни военной кафедры"):
        st.dataframe(pd.DataFrame([{"group": g, "military_day": d}
for g, d in sorted(selected_military.items())]),
use_container_width=True)

st.subheader("Смена по курсам")
st.caption("Необязательная настройка. Для 2 смены пары ставятся не
раньше 11:40, кроме понедельника и субботы, где ранние пары
допустимы как запасной вариант. Для 1 смены алгоритм предпочитает
ранние пары 08:00 и 09:50.")

course_shift_choices: Dict[int, str] = {}
shift_cols = st.columns(min(4, max(1, len(all_courses))))
for idx, course in enumerate(all_courses):
    with shift_cols[idx % len(shift_cols)]:
        chosen_shift = st.selectbox(
            f"{course} курс – смена",
            options=["Не задавать", "1 смена", "2 смена"],
            index=0,
            key=f"shift_course_{course}",
        )
        course_shift_choices[course] = chosen_shift

group_shift_map = build_shift_map(groups_df, course_shift_choices)
effective_times = effective_times_for_shifts(base_times,
course_shift_choices)

```

```

selected_shifts = [
    {"course": course, "shift": shift}
    for course, shift in sorted(course_shift_choices.items())
    if shift != "Не задавать"
]
if selected_shifts:
    with st.expander("Показать выбранные смены"):
        st.dataframe(pd.DataFrame(selected_shifts),
use_container_width=True)

st.subheader("Индивидуальные предпочтения преподавателей")
st.caption("Зелёный – предпочтительное время. Красный –
преподаватель не может вести пары в это время. Пустое значение –
алгоритм решает сам.")

if teachers:
    teacher_col1, teacher_col2 = st.columns([2, 1])
    with teacher_col1:
        selected_teacher = st.selectbox("Преподаватель",
options=teachers, key="selected_teacher_pref")
    with teacher_col2:
        st.write("")
        st.write("")
        if st.button("Очистить настройки преподавателя",
key="clear_selected_teacher"):
            clear_teacher_preferences(selected_teacher, days,
effective_times)
            st.rerun()

    with st.expander(f"Настройка предпочтений: {selected_teacher}",
expanded=True):
        teacher_pref_df = get_teacher_pref_df(selected_teacher,
days, effective_times)
        pref_editor = st.data_editor(
            teacher_pref_df,

key=f"teacher_pref_editor_{selected_teacher}_{'.'.join(days)}_{'.'.j
oin(effective_times)}",
            hide_index=True,
            use_container_width=True,
            column_config={
                "Папа": st.column_config.TextColumn("Папа",
disabled=True),
                **{day: st.column_config.SelectboxColumn(day,
options=PREF_OPTIONS) for day in days},
            },
        )
        if st.button("Сохранить предпочтения преподавателя",
key=f"save_teacher_pref_{selected_teacher}"):
            save_teacher_pref_df(selected_teacher, pref_editor,
days, effective_times)

```

```

        st.success(f"Предпочтения для преподавателя
«{selected_teacher}» сохранены.")
        st.rerun()
    else:
        st.caption("В файле Activities не найдены преподаватели для
настройки предпочтений.")

teacher_pref_map = build_teacher_pref_map(days, effective_times)
if teacher_pref_map:
    with st.expander("Показать сохранённые предпочтения
преподавателей"):

st.dataframe(summarize_teacher_preferences(teacher_pref_map),
use_container_width=True)

if st.button("Составить расписание"):
    try:
        occs = expand_activities_to_occurrences(settings, acts_df)
        assignments = solve_schedule(
            occs=occs,
            rooms=list(rooms),
            days=days,
            times=effective_times,
            max_pairs_per_day=int(settings.get("max_pairs_per_day",
4) or 4),
            military_days_by_group=military_days_by_group,
            teacher_pref_map=teacher_pref_map,
            group_shift_map=group_shift_map,
            time_limit_sec=int(time_limit),
        )

        st.session_state.schedule_assignments = assignments
        st.session_state.schedule_occs = occs
        st.session_state.schedule_context = {
            "groups_df": groups_df,
            "rooms": list(rooms),
            "days": days,
            "times": effective_times,
            "military_days_by_group": military_days_by_group,
            "teacher_pref_map": teacher_pref_map,
            "group_shift_map": group_shift_map,
            "max_pairs_per_day":
int(settings.get("max_pairs_per_day", 4) or 4),
        }
        st.session_state.move_options = []
        st.session_state.move_options_for = ""

        try:
            save_schedule_to_history(assignments,
st.session_state.schedule_context, title="Автоматически составленное
расписание")
        except Exception as history_error:

```

```

        st.warning(f"Расписание составлено, но не удалось
сохранить его в историю SQLite: {history_error}")

        st.success(f"Готово! Назначений: {len(assignments)}")

    except NoFeasibleOptions as e:
        st.error(f"Нет допустимых слотов/аудиторий для
{e.occ.occ_id}")
        st.json(e.diag)
    except NoScheduleFound as e:
        st.error(str(e))
        if getattr(e, "diag", None):
            st.subheader("Что, скорее всего, мешает")
            if e.diag.get("Вывод"):
                for item in e.diag["Вывод"]:
                    st.warning(item)
            for key in ["Проверка ослаблений", "Потенциально
перегруженные группы", "Потенциально перегруженные преподаватели",
"Самые ограниченные занятия"]:
                if e.diag.get(key):
                    st.markdown(f"***{key}***")
                    st.dataframe(pd.DataFrame(e.diag[key]),
use_container_width=True)
        except Exception as e:
            st.error(f"Ошибка: {e}")

if st.session_state.get("schedule_assignments") and
st.session_state.get("schedule_context"):
    assignments = st.session_state.schedule_assignments
    context = st.session_state.schedule_context
    st.subheader("Готовое расписание")

    flat_rows = []
    group_to_course =
build_group_to_course_from_sheet(context["groups_df"])
    for a in assignments:
        for g in a["groups"]:
            flat_rows.append({
                "group": g,
                "course": group_to_course.get(g, 0),
                "week": a["week"],
                "day": a["day"],
                "time": a["time"],
                "discipline": a["discipline"],
                "kind": kind_to_short(a["kind"]),
                "teachers": a["teachers"],
                "room": a["room"],
            })
    if flat_rows:
        df_preview = pd.DataFrame(flat_rows)
        stats = df_preview.groupby(["course",
"group"]).size().reset_index(name="назначений")
        st.caption("Краткая сводка по группам")

```

```

        st.dataframe(stats, use_container_width=True)

    col_view, col_download, col_spacer = st.columns([1.85, 1.35, 4])
    with col_view:
        if st.button("Просмотреть / редактировать расписание",
key="open_schedule_preview"):
            render_schedule_dialog()
    with col_download:
        out_bio = build_output_workbook(
            assignments,
            context["groups_df"],
            context["days"],
            context["times"],
            context["military_days_by_group"],
            context["group_shift_map"],
        )
        st.download_button(
            label="Скачать расписание (xlsx)",
            data=out_bio,
            file_name="raspisanie_by_course.xlsx",
            mime="application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet",
            key="download_current_schedule_main",
        )

    if st.session_state.get("reopen_schedule_dialog"):
        st.session_state.reopen_schedule_dialog = False
        render_schedule_dialog()

```

Приложение В (обязательное)

Система автоматизации составления учебного расписания с использованием
методов оптимизации и учетом ресурсных ограничений

Презентация

БАКАЛАВРСКАЯ РАБОТА

Система автоматизации составления учебного расписания с использованием методов оптимизации и учетом ресурсных ограничений

Автор: студент группы 22ВВП1 Юртаев Дмитрий Юрьевич
Руководитель: д.т.н., профессор кафедры «Вычислительная
техника» Гурин Евгений Иванович
Пенза, 2026

1

Актуальность разработки

Трудности, связанные с ручным составлением расписания:

Много взаимосвязанных ресурсов;

Высокая вероятность ошибок;

Сложность внесения изменений;

Необходимость учитывать индивидуальные
ограничения;

Трудоемкость быстро растет с увеличением
количества входных данных.



2

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.27.001 ПЗ

Лист

128

Цель и задачи работы

Цель: разработка системы автоматизации составления учебного расписания с учетом ресурсных ограничений и методов оптимизации.

- 1 Проанализировать предметную область и аналоги
- 2 Выбрать и настроить инструменты разработки
- 3 Спроектировать архитектуру, модели и ввести учет ограничений
- 4 Реализовать систему составления расписания
- 5 Провести тестирование и отладку

3

Анализ аналогов

aSc TimeTables



UniTime



1C:Расписание



Вывод после анализа

готовые решения функциональны, но могут быть избыточными для отдельной задачи

для работы важны адаптация под конкретные входные данные и ресурсные ограничения

разрабатываемая система ориентирована на простой сценарий: входные данные → проверка → оптимизация → готовое расписание

4

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.27.001 ПЗ

Лист

129

Преимущества разработанной системы

- **Гибкие входные данные** Excel-шаблон с листами Groups, Activities и Rooms
- **Проверка ошибок** Контроль структуры файла, групп, аудиторий и строк с подсказками
- **Учет ограничений** Группы, преподаватели, аудитории, оборудование, учебные смены, военная подготовка
- **Оптимизация** Использование OR-Tools CP-SAT для поиска допустимых решений
- **Практический результат** Просмотр, ручной перенос, экспорт в Excel и история в SQLite

5

Стек технологий

Python

основная логика приложения



Streamlit

интерфейс



OR-Tools CP-SAT

решение задачи ограничений



pandas

чтение и обработка таблиц



openpyxl

шаблон и экспорт Excel



SQLite

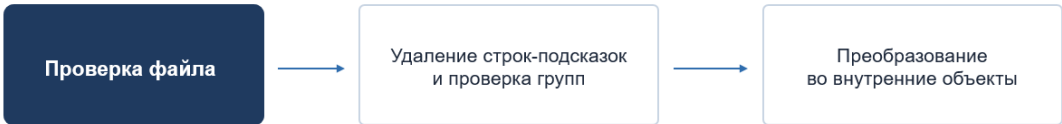
история расписаний



6

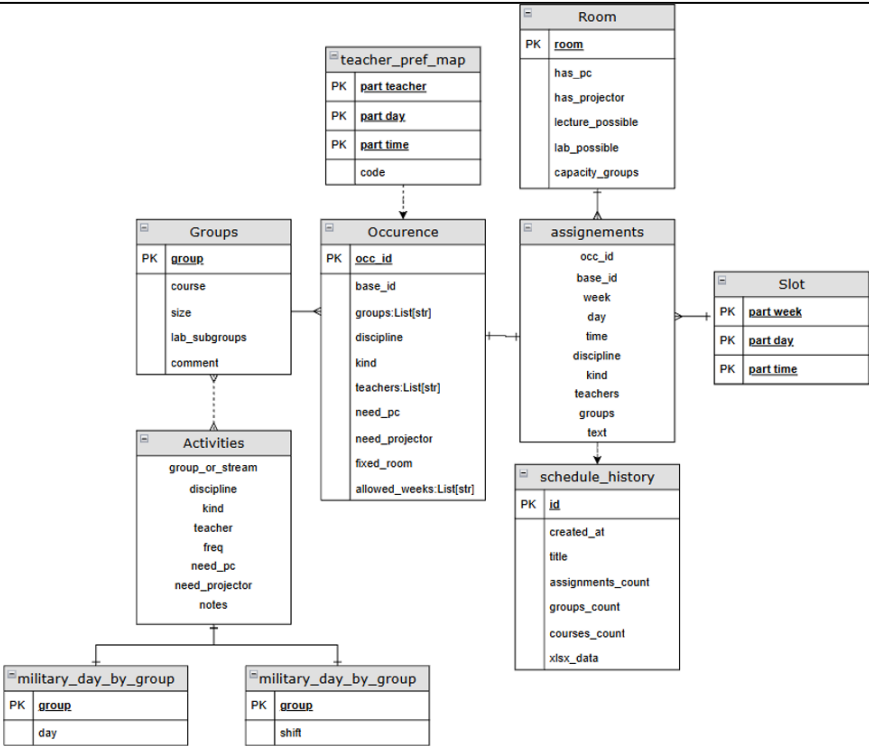
Модель входных данных

Входной файл содержит три обязательных листа



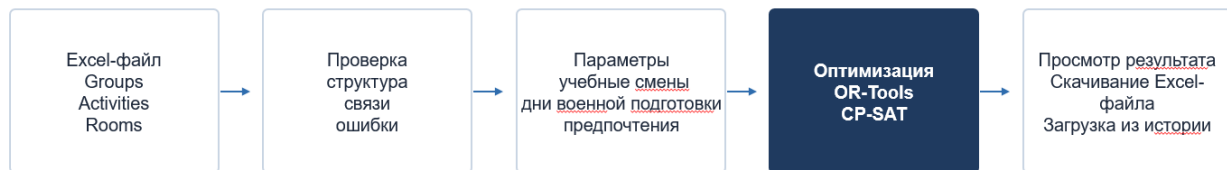
7

ER-модель



8

Архитектура приложения



Все ключевые действия реализованы внутри одного приложения: подготовка данных, построение модели, запуск решателя, отображение, экспорт и сохранение истории.

9

Алгоритм составления расписания

Основная переменная модели:

$x[i, s, r] = 1$
если занятие i поставлено
во время s и аудиторию r

Ограничения:

Занятие назначается один раз

Группа не имеет двух занятий одновременно

Преподаватель не ведет две пары одновременно

Аудитория занята не более одним занятием

Учтены оборудование, учебные смены и недоступные временные интервалы

Целевая функция

поощрение предпочтительных временных интервалов
штрафы за неудобное распределение нагрузки

10

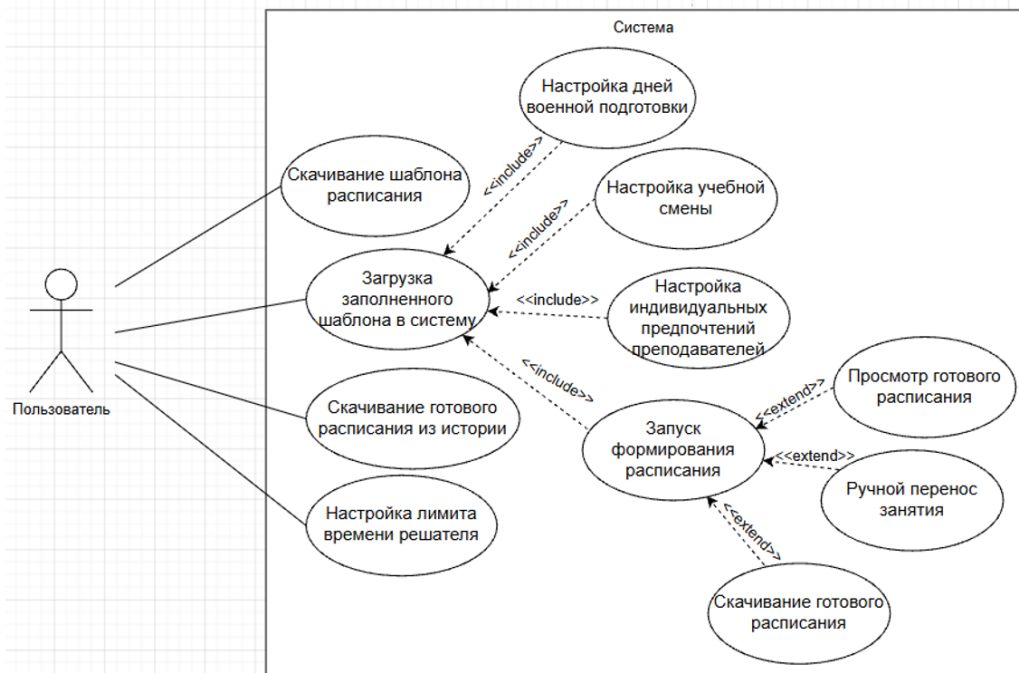
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.27.001 ПЗ

Лист

132

UML-диаграмма деятельности



11

Результат работы системы

Просмотр расписания

В ячейке отображаются дисциплина, тип занятия, преподаватель и аудитория.

Информационные технологии в профессиональной деятельности
лб. Акифьев И.В.
ауд. 7а-322

Экспорт в Excel

Сформированный файл можно открыть, передать или распечатать без дополнительной обработки.

.xlsx

История в SQLite

Последние 10 вариантов расписания хранятся в базе данных в бинарном виде.

BLOB

Сформированный результат можно просмотреть, вручную перенести занятие, скачать и повторно получить из истории.

12

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.27.001 ПЗ

Лист

133

Тестирование и итоги

Проверены ключевые сценарии:

Скачивание шаблона

Загрузка корректного файла

Обработка ошибок входных данных

Запуск генерации расписания

Просмотр, экспорт и история

Итоговый результат

Разработана система, которая автоматизирует составление учебного расписания, учитывает ресурсные ограничения, формирует результат в Excel и сохраняет историю вариантов.

Перспективы развития: клиент-серверная архитектура, роли пользователей, расширение ограничений и дополнительные отчеты.

Цель работы достигнута, поставленные задачи выполнены.

13

СПАСИБО ЗА ВНИМАНИЕ!

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.27.001 ПЗ

Лист

134