

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА/GRADUATION THESIS

Разработка интеллектуальной системы для отслеживания динамики
информационных потоков в образовательном подразделении «Техническая школа»
ГУП «Петербургский метрополитен»

Автор/ Author

Суздальцева Маргарита Вячеславовна

Направленность (профиль) образовательной программы/Major

Интеллектуальные системы в гуманитарной сфере 2017

Квалификация/ Degree level

Бакалавр

Руководитель ВКР/ Thesis supervisor

Добренко Наталья Викторовна, кандидат технических наук, Университет ИТМО, факультет
инфокоммуникационных технологий, доцент (квалификационная категория "ординарный
доцент")

Группа/Group

К3443

Факультет/институт/кластер/ Faculty/Institute/Cluster

факультет инфокоммуникационных технологий

Направление подготовки/ Subject area

45.03.04 Интеллектуальные системы в гуманитарной сфере

Обучающийся/Student

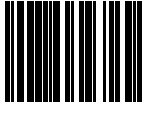
Документ подписан	
Суздальцева Маргарита Вячеславовна	
03.06.2021	

(эл. подпись/ signature)

Суздальцева
Маргарита
Вячеславовна

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Добренко Наталья Викторовна	
03.06.2021	

(эл. подпись/ signature)

Добренко
Наталья
Викторовна

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Суздальцева Маргарита Вячеславовна

Группа/Group K3443

Факультет/институт/кластер/

Faculty/Institute/Cluster

факультет

инфокоммуникационных технологий

Квалификация/ Degree level Бакалавр

Направление подготовки/ Subject area 45.03.04 Интеллектуальные системы в гуманитарной сфере

Направленность (профиль) образовательной программы/Major Интеллектуальные системы в гуманитарной сфере 2017

Специализация/ Specialization

Тема ВКР/ Thesis topic Разработка интеллектуальной системы для отслеживания динамики информационных потоков в образовательном подразделении «Техническая школа» ГУП «Петербургский метрополитен»

Руководитель ВКР/ Thesis supervisor Добренко Наталья Викторовна, кандидат технических наук, Университет ИТМО, факультет инфокоммуникационных технологий, доцент (квалификационная категория "ординарный доцент")

Срок сдачи студентом законченной работы до / Deadline for submission of complete thesis 28.05.2021

Техническое задание и исходные данные к работе/ Requirements and premise for the thesis

Проектирование и разработка системы для автоматизации информационных процессов в технической школе метрополитена. Реализация веб-приложения с помощью актуального стека технологий. Обеспечение безопасности персональных данных. Обеспечение адаптивности веб-приложения.

Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)/ Content of the thesis (list of key issues)

Анализ предметной области и существующих информационных потоков.

Выявление требований к системе, составление списка необходимых модулей и функционала.

Проектирование системы.

Разработка веб-приложения (все модули, интеллектуализация, адаптивность).

Тестирование и обеспечение безопасности.

Перечень графического материала (с указанием обязательного материала) / List of graphic materials (with a list of required material)

Презентация к докладу (обязательно);
 Диаграммы, созданные при проектировании системы (обязательно);
 Модель базы данных (обязательно);
 Интерфейсы веб-приложения (обязательно);
 Поясняющие текст схемы и изображения;
 Скриншоты кода.

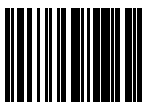
Исходные материалы и пособия / Source materials and publications

Django documentation. URL: <https://docs.djangoproject.com/en/3.1/>
 Документация Vue.js. URL: <https://ru.vuejs.org/v2/guide/>
 Django REST framework. URL: <https://www.django-rest-framework.org/>
 PostgreSQL: Documentation. URL: <https://www.postgresql.org/docs/>
 Сайт ресурсов UML. URL: <https://www.uml.org/>
 Python Documentation. URL: <https://www.python.org/doc/>
 Методология функционального моделирования IDEF0: руководящий документ. URL: <https://nsu.ru/smk/files/idef.pdf>

Дата выдачи задания/ Objectives issued on 28.04.2021

СОГЛАСОВАНО / AGREED:


Руководитель ВКР/
 Thesis supervisor

Документ подписан	
Добренко Наталья Викторовна	
28.04.2021	

(эл. подпись)

Добренко
Наталья
Викторовна

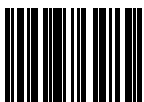
Задание принял к
 исполнению/ Objectives
 assumed by

Документ подписан	
Суздальцева Маргарита Вячеславовна	
29.04.2021	

(эл. подпись)

Суздальцева
Маргарита
Вячеславовна

Руководитель ОП/ Head
 of educational program

Документ подписан	
Хлопотов Максим Валерьевич	
29.05.2021	

(эл. подпись)

Хлопотов
Максим
Валерьевич

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ /
SUMMARY OF A GRADUATION THESIS**

Обучающийся/ Student

Суздальцева Маргарита Вячеславовна

Наименование темы ВКР / Title of the thesis

Разработка интеллектуальной системы для отслеживания динамики информационных потоков в образовательном подразделении «Техническая школа» ГУП «Петербургский метрополитен»

Наименование организации, где выполнена ВКР/ Name of organization

Университет ИТМО

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ/
DESCRIPTION OF THE GRADUATION THESIS**

1. Цель исследования / Research objective

Разработка интеллектуальной системы для отслеживания динамики информационных потоков в образовательном подразделении «Техническая школа» ГУП «Петербургский метрополитен»

2. Задачи, решаемые в ВКР / Research tasks

Анализ специфики организации и существующих информационных потоков. Выявление требований к интеллектуальной информационной системе для Технической школы, определение перечня модулей. Проектирование системы, описание возможностей интеллектуализации. Разработка веб-приложения с использованием актуальных технологий. Обеспечение базовой безопасности системы.

3. Краткая характеристика полученных результатов / Short summary of results/conclusions

Проведён анализ предметной области, на основании которого спроектированы и описаны модули будущей системы. Созданы структуры данных, обеспечивающие хранение и обработку информации. С помощью Django реализована пилотная версия системы для отслеживания динамики информационных потоков в организации. Предложены решения по масштабированию системы. Приведены требования к безопасности.

4. Наличие публикаций по теме выпускной работы/ Have you produced any publications on the topic of the thesis

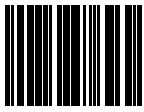
5. Наличие выступлений на конференциях по теме выпускной работы/ Have you produced any conference reports on the topic of the thesis

всероссийский)

6. Полученные гранты, при выполнении работы/ Grants received while working on the thesis

7. Дополнительные сведения/ Additional information

Обучающийся/Student

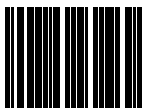
Документ подписан	
Суздальцева Маргарита Вячеславовна	
03.06.2021	

(эл. подпись/ signature)

Суздальцева
Маргарита
Вячеславовна

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Добренко Наталья Викторовна	
03.06.2021	

(эл. подпись/ signature)

Добренко
Наталья
Викторовна

(Фамилия И.О./ name
and surname)

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ	9
ВВЕДЕНИЕ.....	10
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	11
1.1 Анализ контекста и специфики организации	11
1.2 Основной бизнес-процесс	13
1.3 Описание информационных потоков	14
Выводы по первой главе	15
2 ВЫЯВЛЕНИЕ ТРЕБОВАНИЙ К СИСТЕМЕ	16
2.1 Общие требования к разрабатываемой системе	16
2.2 Требования к безопасности системы	16
2.3 Требования к функционалу системы.....	16
2.4 Возможность использования аналогов.....	17
Выводы по второй главе.....	19
3 ПРОЕКТИРОВАНИЕ СИСТЕМЫ.....	20
3.1 Общее видение	20
3.1.1 IDEF0.....	20
3.1.2 IDEF3.....	23
3.1.3 DFD.....	23
3.1.4 Диаграммы UML	24
3.1.5 Use case.....	24
3.1.6 Диаграмма развёртывания	25
3.1.7 Диаграмма компонентов	27
3.1.8 Диаграмма классов	28
3.2 Проектирование базы данных.....	29

3.3 Модуль «БД Преподавателей»	30
3.4 Модуль «Планирование».....	31
3.5 Модуль «Учёт часов»	34
3.6 Модуль «Электронный журнал»	35
3.7 Модуль «Расчёты»	36
3.8 Модуль «Документы».....	38
Выводы по третьей главе	38
4 РАЗРАБОТКА МОДУЛЕЙ СИСТЕМЫ	39
4.1 Статус разработки модулей системы.....	39
4.2 Используемые в проекте технологии и принятые решения	39
4.3 Модуль «БД Преподавателей»	41
4.5 Модуль «Планирование».....	47
4.5.1 MVP	47
4.5.2 Потенциал интеллектуализации	53
4.6 Модуль «Учёт часов»	53
4.7 Версионность планирования и учёта часов	57
4.8 Модуль «Электронный журнал»	59
4.8.1 Формирование расписания	59
4.8.2 Добавление оценок и просмотр информации занятий.....	60
4.8.3 Получение средних показателей студентов	62
4.9 Модуль «Расчёты»	64
Выводы по четвертой главе.....	68
5 БЕЗОПАСНОСТЬ И МАСШТАБИРОВАНИЕ СИСТЕМЫ	69
5.1 Обеспечение безопасности системы	69
5.1.1 Информационные меры.....	69

5.1.2 Разграничение доступа	69
5.1.3 Хранение данных	69
5.1.4 Выбор технологий разработки	69
5.1.5 Условия развёртывания системы	70
5.2 Возможность дальнейшего масштабирования системы.....	70
5.2.1 Интеграция модулей системы	70
5.2.2 Повторное использование данных для анализа.....	71
Выводы по пятой главе	71
ЗАКЛЮЧЕНИЕ	72
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	73
ПРИЛОЖЕНИЕ А Диаграмма IDEF0	75
ПРИЛОЖЕНИЕ Б Схема базы данных приложения	76
ПРИЛОЖЕНИЕ В Код notification_script.py	77

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

БД – База данных

ЖЦ – Жизненный цикл

ИИС – Интеллектуальная Информационная Система

СУБД – Система управления базами данных

ТШ – Техническая школа

Эндпоинт – конечная точка API

Backend (бэкенд) – серверная часть веб-приложения

Frontend (фронтенд) – клиентская часть веб-приложения

JS – JavaScript

MVP – Minimum Viable Product, минимальный жизнеспособный продукт

ВВЕДЕНИЕ

В современных реалиях хранение и дальнейшее использование данных редко обходится без применения информационных технологий. Очевидно, что использование интеллектуальных информационных систем в деятельности организации способно значительно повлиять на эффективность реализации её бизнес-процессов. Особенно остро проблема недостатка автоматизации проявляется в тех случаях, когда в организации существует много различных информационных потоков. Обращение к имеющейся информации может быть затруднено при отсутствии её упорядоченного, структурированного, целостного хранения. В результате потенциально ценная информация не может быть повторно использована для обработки и анализа. Документы накапливаются в аналоговом формате без сохранения структуры связей между ними.

«Техническая школа» ГУП «Петербургский метрополитен» – яркий пример образовательного подразделения, нуждающегося в оптимизации процессов, касающихся работы с информацией. Создание интеллектуальной информационной системы для Технической школы способно упростить и систематизировать администрирование учебного процесса этого образовательного подразделения. Более того, такая система позволит в будущем отслеживать динамику информационных потоков в организации.

Таким образом, задачами выпускной квалификационной работы являлись анализ информационных потоков, характерных для Технической школы, а затем – проектирование и разработка соответствующей интеллектуальной информационной системы (ИИС).

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Анализ контекста и специфики организации

Образовательное подразделение «Техническая школа» ГУП «Петербургский метрополитен» – учебное подразделение технической направленности, обучающее ряду узко-специфических профессий [9]. Специалистов набирают строго по количеству имеющихся рабочих мест, а процесс обучения может длиться от одного дня до шести месяцев.

Администратор учебного процесса занимается составлением документов, опираясь на определённые источники информации. Во-первых, поступающие извне нормативные документы и акты, регулирующие деятельность организации:

- учебные планы (по программам),
- лицензии преподавателей,
- график рабочих и выходных дней на год (производственный календарь),
- календарный план проведения обучения (на год).
- положение о стипендии студентам,
- положение об оплате преподавателей,
- приказы о начале обучения (по группам).

Иначе говоря, «Петербургский метрополитен» предоставляет своему подразделению данные о том, сколько необходимо и каких именно специалистов нужно подготовить, в какие сроки. В свою очередь, Техническая школа составляет собственные планы и передает их на утверждение.

Во-вторых, требующая постоянной актуализации информация о преподавателях. Потенциальными преподавателями являются не только непосредственные сотрудники Технической школы, но и сотрудники других подразделений метрополитена. Несмотря на наличие в педагогическом составе организации профессионалов отрасли, её узость создаёт дефицит кадров. Штат не является постоянным. Сотрудник, который может быть приглашён преподавать, должен иметь соответствующее образование, а также проходить регулярную переподготовку. Цифровые либо аналоговые копии

лицензий, подтверждающих завершённое обучение и его сроки, хранятся в организации. Естественно, имеет значение и возможность сотрудника преподавать в тот период, когда его услуги актуальны.

Для успешного осуществления образовательной деятельности требуется учитывать значения многих переменных. При этом в организации отсутствует структурированное, упорядоченное хранение данных, а бизнес-процессы осуществляются с минимумом автоматизации. Документы, логически связанные между собой, хранятся разрозненно и в аналоговой форме. В случае необходимости обращения к определённому документу могут возникнуть сложности с его поиском. Отдельные документы могут храниться в электронном формате, но, как правило, не имеют ссылок на другие, например, использованные при составлении.

Техническая школа метрополитена ведёт учёт посещаемости и успеваемости студентов. Преподаватели ведут традиционные бумажные журналы, затем они вручную обрабатываются администратором учебного процесса для расчёта стипендий. Электронные журналы в качестве основного хранилища данных об оценках учащихся используются во многих учебных учреждениях. Эта практика может быть применена и здесь.

Затруднительно избежать личного контакта администратора с сотрудниками для получения актуальных данных о их возможности преподавать. Тем не менее, хранение и отслеживание полученной информации может быть автоматизировано. Оптимальным решением проблемы является грамотное хранение и постоянный мониторинг данных преподавателей. При соблюдении этих условий, в необходимый момент всегда будет возможно найти подходящего преподавателя, независимо от того, является ли он сотрудником Технической школы или другого подразделения.

Описывая контекст, важно упомянуть, что документирование учебного процесса подразумевает хранение персональных данных его участников. Автоматизация в таком случае требует соблюдения ряда мер по обеспечению безопасности.

1.2 Основной бизнес-процесс

Для того, чтобы предложить какое-либо решение по автоматизации, нужно понять, как функционирует объект автоматизации.

Администратор получает календарный план проведения обучения на год, содержащий учебные программы, по которым требуются специалисты, и количество мест. Исходя из полученных чисел, администратор формирует планы подготовки и повышения квалификации рабочих кадров:

- 1) учебный план на год,
- 2) учебный план на месяц.

Они включают в себя группы, сформированные из количества студентов по учебным программам, и часы, взятые из учебных программ, необходимые для прохождения обучения. Планы формируются с учётом графика рабочих и выходных дней на год.

Исходя из дисциплин учебных программ и календарных планов требуется найти преподавателей. Для этого администратор обращается к аналоговой базе преподавателей, ранее уже работавших в Технической школе. Обычно хранятся следующие параметры:

- 1) наименование подразделения сотрудника,
- 2) дата прохождения им педагогической переподготовки,
- 3) дата прохождения повышения квалификации,
- 4) отсканированные копии лицензий, подтверждающих даты.

Сотрудники обязаны проходить повышение квалификации каждые три года, следовательно, необходимо учесть наличие лицензии у преподавателя на период преподавания, а также напоминать сотрудником о приближающейся дате истечения лицензии.

При составлении любых планов необходимо соответствовать нормам часов преподавания (в день, в месяц, год).

По завершении планирования формируются письма со сроками обучения и списками преподавателей по группам. Они отправляются

Технической школой в управление для утверждения (списков преподавателей, студентов, сроков) и получения приказа о начале обучения.

В процессе обучения преподаватели отмечают посещаемость студентов и ставят им оценки в журнал, который передаётся администратору для проверки и контроля.

Администратор рассчитывает суммы выплат как студентам, так и преподавателям, и передаёт эту информацию в управление. Стипендия формируется на основе ведомостей успеваемости и посещаемости студентов, а также из положения о стипендии студентам. Помимо этого, исходя из таблицы учёта часов преподавания и из табелей преподавателей, а также из положения об оплате преподавателей, администратор рассчитывает зарплаты преподавателей и передаёт эту информацию в управление.

В конце обучения по каждому предмету проводится экзамен, на который собирается комиссия из сотрудников Технической школы и метрополитена. Оценки за экзамены также вносятся в ведомости.

1.3 Описание информационных потоков

Информационным потоком в контексте данной работы считается совокупность сообщений, перемещающихся внутри системы (между ее сущностями), а также между системой и внешним миром (сущностями, не включенными в систему). Информационные потоки в системе, как правило, влияют на её функционирование.

Техническая школа выпускает около 4 тысяч специалистов в год, сама суть организации предполагает постоянную циркуляцию информации, касающейся ее деятельности. Все информационные потоки рассматриваемого образовательного подразделения – регулярны. Большинство видов ведомостей и отчётов формируются и утверждаются каждый месяц, некоторые – один или несколько раз в год. Обучение вышеупомянутого числа студентов создаёт значительные объёмы документации. Сама по себе такая информация только обеспечивает учебный процесс. Однако, ее систематизированное накопление могло бы помочь отслеживать динамику информационных процессов в

организации. Сопоставление данных за разные периоды позволило бы выявлять закономерности и использовать предыдущий опыт. Например, оптимизировать отбор кадров для преподавания и повысить скорость и эффективность остальных процессов.

Информационные потоки описываемой организации были систематизированы с помощью схемы информационных потоков (UML Information Flow Diagram) (рисунок 1). Эта поведенческая диаграмма визуализирует передвижение информации и анализ различных ситуаций в системе, показывает обмен данными между системами [8].



Рисунок 1 – Information Flow Diagram

Выводы по первой главе

Описание контекста и специфики организации, а также существующих в ней информационных потоков, позволяет сделать вывод о необходимости внедрения в неё информационной системы, которая бы отвечала потребностям сотрудников Технической школы, а также требованиям безопасности и актуальности применяемых технологий.

2 ВЫЯВЛЕНИЕ ТРЕБОВАНИЙ К СИСТЕМЕ

2.1 Общие требования к разрабатываемой системе

Было принято решение спроектировать и разработать, а затем интегрировать в организацию интеллектуальную информационную систему (ИИС). В процессе взаимодействия с представителем организации были выявлены основные требования к системе.

Во-первых, веб-интерфейс как основной способ взаимодействия с системой. Нежелательно использовать стороннее ПО.

Во-вторых, адаптивность веб-интерфейса модуля, который будут использовать преподаватели, под мобильные устройства.

Два вида пользователей: администратор (он же руководитель) и рядовые преподаватели. Администратору для работы требуется вся полнота функционала, а преподавателям – инструменты для ведения электронного журнала. Студенты системой не пользуются.

Имеют значение расширяемость, возможность дальнейшего масштабирования системы. Важна версионность (хранение версий) там, где это необходимо.

Наконец, полная или частичная имитация той логики (бизнес-процессов), которая уже действует в организации. Связность, наличие логических связей между объектами.

2.2 Требования к безопасности системы

Как было сказано ранее, предполагается хранение персональных данных сотрудников и студентов. Необходимо тщательно подойти к выбору технических и программных средств создания заявленной системы, чтобы не допустить несанкционированного проникновения и недобросовестного использования данных ее пользователей.

2.3 Требования к функционалу системы

Составлен список функций, которые должна выполнять система.

Во-первых, хранение данных преподавателей, в частности – их лицензий, а также контроль сроков их действия и своевременное уведомление об их истечении. Фильтрация по любым полям.

Во-вторых, поддержка планирования: создание, хранение и просмотр всех видов планов в рамках системы.

Возможность ведения электронного журнала разными преподавателями, и автоматической обработки его данных администратором (получение средних баллов, процента посещаемости).

Поддержка ведения учёта часов преподавания. Согласованность норм, фактических часов, занятий в расписании. Просмотр сводок по разным периодам и преподавателям.

Автоматические базовые расчёты (стипендий, зарплат), на основании данных электронного журнала и часов преподавания.

Наконец, формирование отчётов на основании данных, хранящихся в системе, в также загрузка документов в систему (лицензий, норм, актов, программ) для обновления переменных, используемых в обработке данных.

Таким образом, был определён перечень модулей будущей системы:

- модуль «БД Преподавателей»,
- модуль «Планирование»,
- модуль «Учёт часов»,
- модуль «Электронный журнал»,
- модуль «Расчёты»,
- модуль «Документы».

2.4 Возможность использования аналогов

Из предыдущих подразделов ясно, что собственная система должна быть разработана из-за требований безопасности Технической школы. Представителями организации были упомянуты нормативные положения, запрещающие использование стороннего ПО. Также может оказаться предпочтительнее использовать иной стек технологий, чем используют многие из существующих аналогов. Из-за совмещения многих функций в одну

информационную систему, подобрать приложение, покрывающее все, затруднительно.

Неполные аналоги всё-таки существуют. Аналогами информационной системы для Технической школы являются LMS (Learning Management System). Наиболее широко-распространённым примером LMS является Moodle (модульная объектно-ориентированная динамическая обучающая среда). Например, moodle.org [12] – самый популярный вариант – и подобные: Google Classroom, Schoology Learning, а PowerSchool Unified Classroom product, Blackboard Learn [13]. Однако не планируется использовать разрабатываемую систему для размещения на ней образовательных курсов. LMS позволяют осуществлять планирование и вести электронный журнал, но предполагают вовлечение студентов в пользование системой, что не требуется в Технической школе. То есть, вероятно, многие функции Moodle в данном случае являются излишними.

Существуют и более комплексные системы управления школой, например, Capita SIMS [14], где SIMS – School Information Management System, то есть информационная система управления школой (подкласс информационных систем управления MIS). Данное приложение широко используется для поддержки учебного процесса школами и СПО по всему миру. Компания, разрабатывающая продукт, занимает достаточно большую часть рынка, так как другие аналоги (в том числе русскоязычные) обладают меньшей гибкостью и удобством использования.

Эффективным аналогом практически любому разрабатываемому с нуля модулю для табличных расчётов является Microsoft Excel. Тем не менее, такой способ требует активного участия пользователя в процессе расчётов. Кроме того, расчёты – лишь один аспект в поддержке вышеописанного учебного процесса.

Также предпринимались попытки найти аналоги среди приложений для управления персоналом и систем для ведения внутреннего электронного

документооборота компании, но поиски в данных направлениях оказались неэффективными.

По итогам обзора аналогов возможно сделать следующие выводы:

1) комплексные и проверенные временем программные продукты чаще всего требуют приобретения лицензии и подписки, реже – бесплатны и имеют открытый исходный код,

2) существующие решения обычно состоят из нескольких различных модулей, декомпозиция функционала является распространённой и эффективной практикой,

3) аналоги покрывают лишь часть требований Технической школы, что делает выбор лишь одного продукта затруднительным, но оставляет возможность для интеграции нескольких продуктов.

Выводы по второй главе

Были выявлены основные требования к системе для образовательного подразделения «Техническая школа» ГУП «Петербургский метрополитен»: общие требования, требования к безопасности, функциональные требования. Были рассмотрены частичные аналоги системы, приведены причины, по которым затруднительно использовать готовые решения. Оптимальный вариант реализации ИИС с такими характеристиками – веб-приложение. Таким образом, следующие разделы выпускной квалификационной работы посвящены проектированию и разработке веб-приложения для Технической школы.

3 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

3.1 Общее видение

В ходе предварительного проектирования информационной системы различные её аспекты были представлены в виде диаграмм и схем.

3.1.1 IDEF0

Основной бизнес-процесс системы визуализирован в нотации IDEF0 (приложение А). IDEF0 (Icam DEFinition for Function Modeling, где ICAM – аббревиатура Integrated Computer Aided Manufacturing) – методология функционального моделирования для описания производственных функций, которая предлагает язык функционального моделирования для анализа, разработки, реинжиниринга и интеграции информационных систем, описания бизнес-процессов или программного инженерного анализа [6].

В таблице 1 представлено описание подпроцессов, составляющих основной бизнес-процесс, отраженных на диаграмме IDEF0, построенной для ИС Технической школы метрополитена.

Таблица 1 – Описание IDEF0

Элемент	Описание
Загрузить и обработать входные данные (1)	<p>Данный элемент подразумевает загрузку исходной информации в систему, она является входными данными процесса</p> <p>Администратор-руководитель загружает информацию о преподавателях из имеющейся аналоговой базы, информацию о студентах нового потока, нормативные документы (присылаемые начальством регулярно: учебные планы, приказы, другие нормативы), график рабочих/выходных дней</p> <p>Данные перед загрузкой находятся в различных форматах (сканы и фото в PDF, JPG), часть обрабатывается автоматически, часть – вносится вручную в БД</p> <p>Управление: информация о форматах документов, информация о местонахождении значимых данных в документе, то есть, какие-либо принципы извлечения и внесения данных в систему</p> <p>Механизмы: программные модули для извлечения текста из PDF и JPG, собственно БД, в которой хранится информация системы (например, PostgreSQL), администратор для ввода информации, которая заполняется вручную</p> <p>Выходные данные, таким образом, – БД с данными за текущий период (обычно год)</p>

Элемент	Описание
Сформировать общие планы обучения (2)	<p>Под общими имеются в виду годовые и месячные планы обучения, содержащие информацию о временных промежутках обучения, группах по различным программам подготовке, необходимом количестве человек и групп; черновики планов создаются программными средствами, админ заполняет их до конца, редактирует и утверждает; при создании используются сторонняя информация и информация из БД</p> <p>На входе – БД из предыдущего элемента</p> <p>Управление: часы и предметы из учебных планов, даты из приказов и графиков</p> <p>Механизмы: языки программирования, реализующие веб-интерфейс планирования, админ</p> <p>На выходе процесса – таблицы общих планов</p>
Актуализировать информацию о преподавателях (3)	<p>Подбирая преподавателей для нового потока обучающихся, руководитель должен учитывать базу ранее преподававших сотрудников; это могут быть как постоянные преподаватели ТШ, так и те, для которых постоянным местом работы является какое-либо другое подразделение метрополитена</p> <p>Данный процесс подразумевает обновление информации о потенциальных преподавателях, здесь система предусматривает именно сохранение этой информации в системе, а не автоматическое ее получение; админ-руководитель самостоятельно производит обзвон кандидатов, в том числе – для уточнения готовности преподавать</p> <p>Вход – таблицы общих планов, из которых ясно, преподаватели каких предметов требуются и в какие даты</p> <p>Управление: нормы о сроках прохождения образования или переподготовки (для преподавания нужны актуальные лицензии и регулярное повышение квалификации в установленные сроки), а также контакты преподавателей</p> <p>Механизмы: «прозвон» админом, веб-интерфейс для редактирования БД</p> <p>Выход: таблицы преподавателей (с актуальными данными о них, прикрепленными документами, подтверждающими квалификацию)</p>
Сформировать новые планы обучения для преподавателей и групп (4)	<p>Данная функция охватывает составление новых более узких планов на основе имеющихся общих. Индивидуальные планы, в отличие от общих, уже включают детали о группах, принятых на обучение студентах, днях и часах работы преподавателей, предметах и т. д. Этот же процесс включает планирование часов работы преподавателей, так как учёт часов регламентируется нормами.</p> <p>Вход – таблицы планов (на год и месяц), таблицы преподавателей.</p> <p>Управление: нормы о количестве часов работы за период, рабочее расписание сотрудника (существуют нормы часов преподавания в день, неделю и месяц, преподавание в личное и рабочее время учитываются и оплачиваются различно)</p> <p>Механизмы: (аналогично составлению общих планов) языки программирования, реализующие веб-интерфейс планирования, админ</p> <p>Выход – таблицы индивидуальных планов</p>

Элемент	Описание
Получить и обработать данные электронного журнала (5)	<p>В информационной системе предполагается наличие такой подсистемы, как электронный журнал (заменяющий существующий аналоговый журнал); он формируется на основании существующих планов групп</p> <p>Заполнением журнала оценок и посещаемости занимаются преподаватели предметов</p> <p>Админ-руководитель получает и обрабатывает эти данные с помощью ИС</p> <p>Вход – таблицы индивидуальных планов, дополнительно – оценки и посещаемость студентов по группам</p> <p>Управление – правила расчета средних баллов и коэффициента посещаемости (они далее понадобятся при расчёте стипендий студентов)</p> <p>Механизмы – автоматические расчеты на JS либо python</p> <p>Выход – таблицы промежуточных результатов обучения</p>
Рассчитать стипендии студентов и зарплаты преподавателей (6)	<p>Для выделения части бюджета организации на зарплаты и стипендии требуется знать точные суммы</p> <p>Заработная плата преподавателя регламентируется положением об оплате преподавателя, зависит от переменных в таблице учета часов преподавания (а также от его квалификации), от контингента студентов группы</p> <p>Стипендия студента ТШ регулируется положением о стипендии ученикам и зависит от промежуточных результатов студента (оценок и посещаемости)</p> <p>Таким образом, входные данные – таблицы промежуточных результатов</p> <p>Управление – положение об оплате преподавателя, положение о стипендии ученикам</p> <p>Механизмы – автоматические расчеты на уровне (возможно реализовать как на backend, так и на frontend)</p> <p>Выход – рассчитанные суммы</p>
Сформировать итоговые таблицы для вывода в нужный формат (7)	<p>Фактически, эта функция работает с любыми из имеющихся таблиц-планов</p> <p>Наличие ИС не отменяет бумажный документооборот организации: админ формирует итоговые таблицы и экспортирует их в необходимый формат, например, XLSX</p> <p>Вход – расчеты с предыдущего элемента диаграммы, таблицы планов</p> <p>Управление – шаблоны отчетов</p> <p>Механизмы – данные из БД + средства реализации frontend и backend, модуль для экспорта отчетов, админ-руководитель (редактирует и утверждает)</p> <p>Выход – отчеты для документооборота</p>

Описанные функции покрывают составные части основного бизнес-процесса разрабатываемой ИС.

3.1.2 IDEF3

IDEF3 (Integrated DEFinition for Process Description Capture Method) – дополнительный к IDEF0 метод моделирования бизнес-процесса, стандарт документирования процессов, происходящих в системе. IDEF3 использует структурный метод выражения знаний о функционировании системы (процесса, предприятия...), чтобы показать причинно-следственные связи между ситуациями или событиями.

В ходе проектирования информационной системы один из элементов диаграммы IDEF0 был декомпозирован и смоделирован с помощью методологии IDEF3. Речь идёт о третьем блоке «Актуализировать информацию о преподавателях». Такой выбор объясняется наличием дополнительных нюансов, которые не были в полной мере раскрыты в рамках IDEF0. Приведённая ниже (рисунок 2) диаграмма демонстрирует альтернативные сценарии развития выбранного бизнес-процесса.

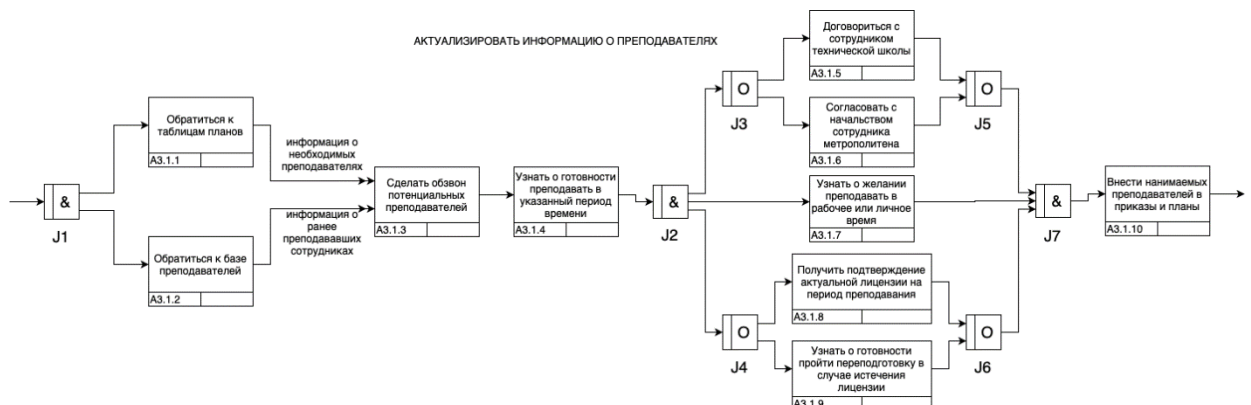


Рисунок 2 – IDEF3

3.1.3 DFD

DFD (Data-flow diagram) – способ представления потока данных, проходящих через процесс или систему (обычно, информационную систему). Поток данных, свойственные данной системе, отражены на представленной далее диаграмме DFD (рисунок 3). Из двух вариантов нотаций используется нотация Гейна-Сарсон.

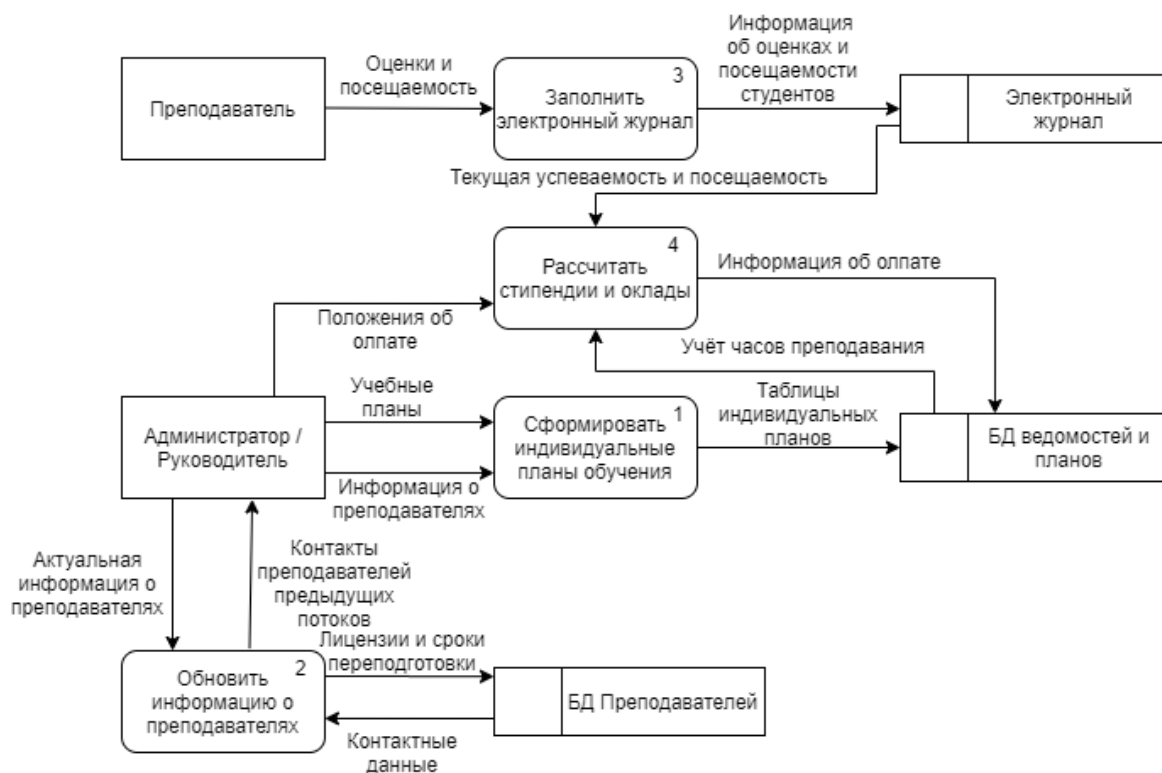


Рисунок 3 – DFD

3.1.4 Диаграммы UML

Также были созданы некоторые виды диаграмм UML. UML (Unified Modeling Language – унифицированный язык моделирования) – система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Иными словами, это язык графического описания, который обеспечивает стандартный способ визуализировать конструкцию системы [4, 5, 18].

3.1.5 Use case

Диаграмма использования (Use case) является представлением взаимодействия пользователя(лей) с системой. Данная диаграмма отражает связи между акторами (пользователями) и прецедентами (случаями взаимодействия). На диаграмме использования для разрабатываемой системы (рисунок 4) изображены два актора – админ (администратор учебного процесса, руководитель) и Преподаватель (это упрощение, фактически, преподавателей много). Преподаватель имеет не так много вариантов взаимодействия с ИС: он может осуществлять вход, заполнять и

просматривать Gradebook (электронный журнал). Это покрывает обязанности преподавателя в контексте системы. Список возможных взаимодействий админа с системой гораздо шире.

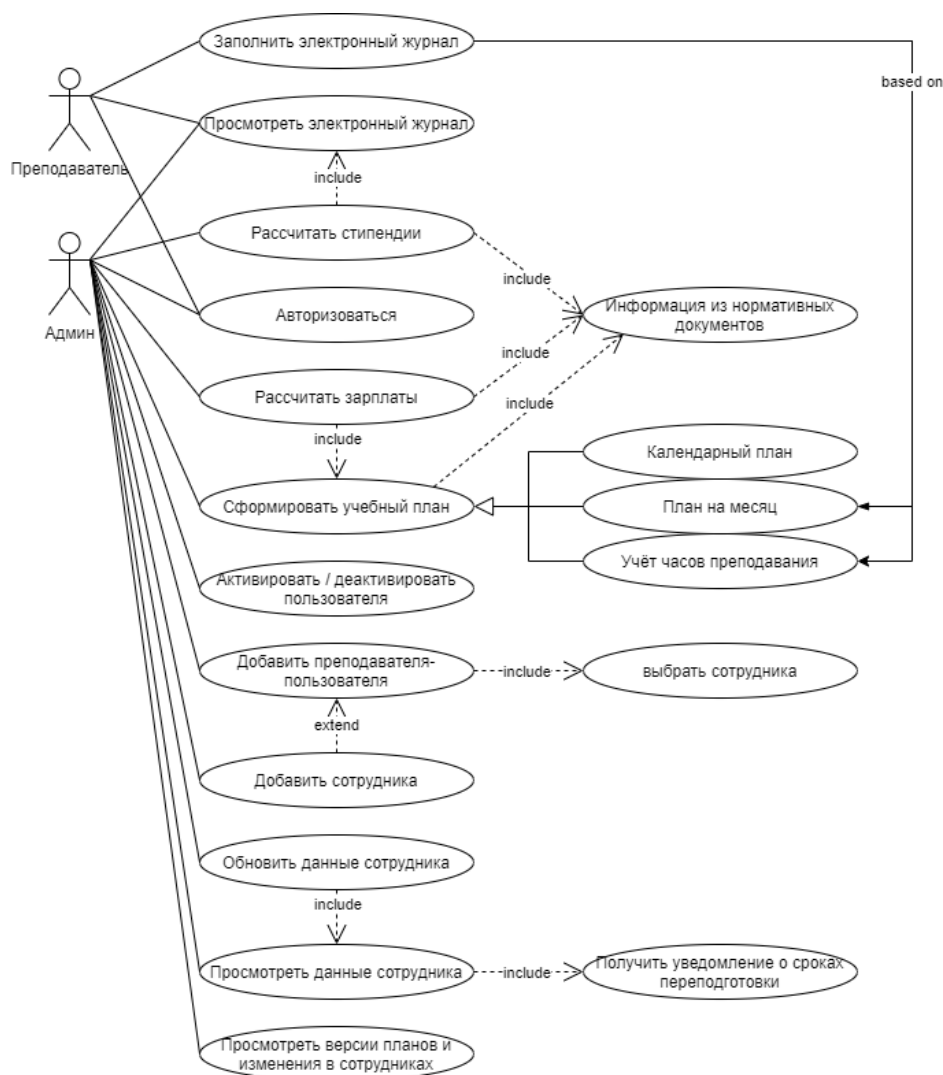


Рисунок 4 – Use case

3.1.6 Диаграмма развёртывания

Диаграмма развёртывания (Deployment Diagram) в рамках UML моделирует физическое развертывание артефактов на узлах, где узел – некоторый физически существующий элемент системы. В контексте веб-сайта обычно показывают какие существуют узлы (аппаратные компоненты: веб-сервер, сервер базы данных, сервер приложения...), какие на каждом узле работают артефакты (программные компоненты: веб-приложение, база данных...), как различные части соединяются друг с другом.

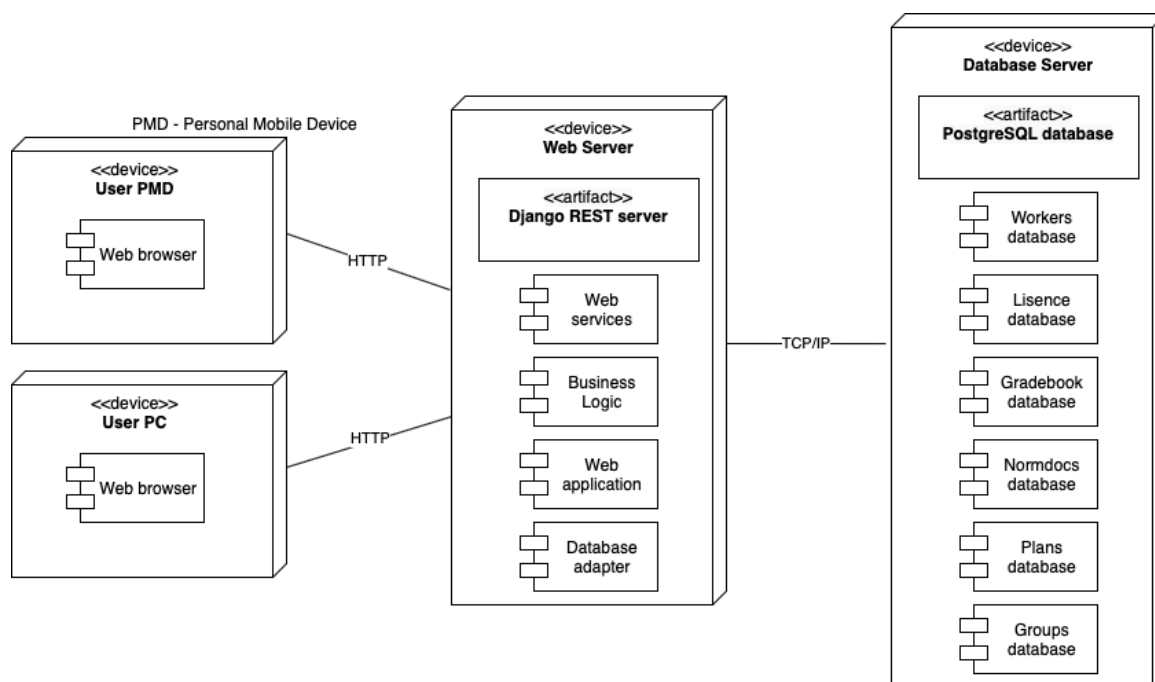


Рисунок 5 – Диаграмма развёртывания

В требованиях к разрабатываемой ИС было заявлено её использование как с ПК, так и с мобильных устройств. Второй вариант касается в основном преподавателей ТШ. Диаграмма представлена на рисунке 5.

Пользовательские устройства отражены в левой части диаграммы: это <<device>> User PMD (PMD - Personal Mobile Device) и <<device>> User PC. Оба включают в себя Web browser. Фактически, это клиентская часть системы. “Общение” с сервером происходит с помощью протокола HTTP.

Узел посередине изображения – <<device>> Web Server. Указанный артефакт объясняется тем, что серверная часть системы будет, вероятно, реализована с помощью Django. Также в данный узел были включены следующие компоненты: Web services, Business Logic, Web application, Database adapter.

Справа расположен узел <<device>> Database Server – сервер базы данных, на котором работает артефакт <<artifact>> PostgreSQL database. Компоненты в данном случае отражают основные элементы БД: Workers database, License database, Gradebook database, Normdocs database, Plans database, Groups database.

Данные между Web Server и Database Server передаются с помощью протокола TCP/IP.

3.1.7 Диаграмма компонентов

Диаграмма компонентов (Component Diagram) в UML – статическая структурная диаграмма, показывающая разбиение программной системы на структурные компоненты (файлы, библиотеки, модули, исполняемые файлы, пакеты и т. д.) и связи (зависимости) между компонентами. Компонента может предоставлять или требовать для своей работы какие-либо интерфейсы.

Построенная для разрабатываемой системы диаграмма компонентов содержит достаточно большое количество компонентов и подсистем компонентов. Итоговый перечень модулей системы, окончательно сформированный уже на этапе разработки системы, несколько отличается от приведенного на данной диаграмме (рисунок 6).

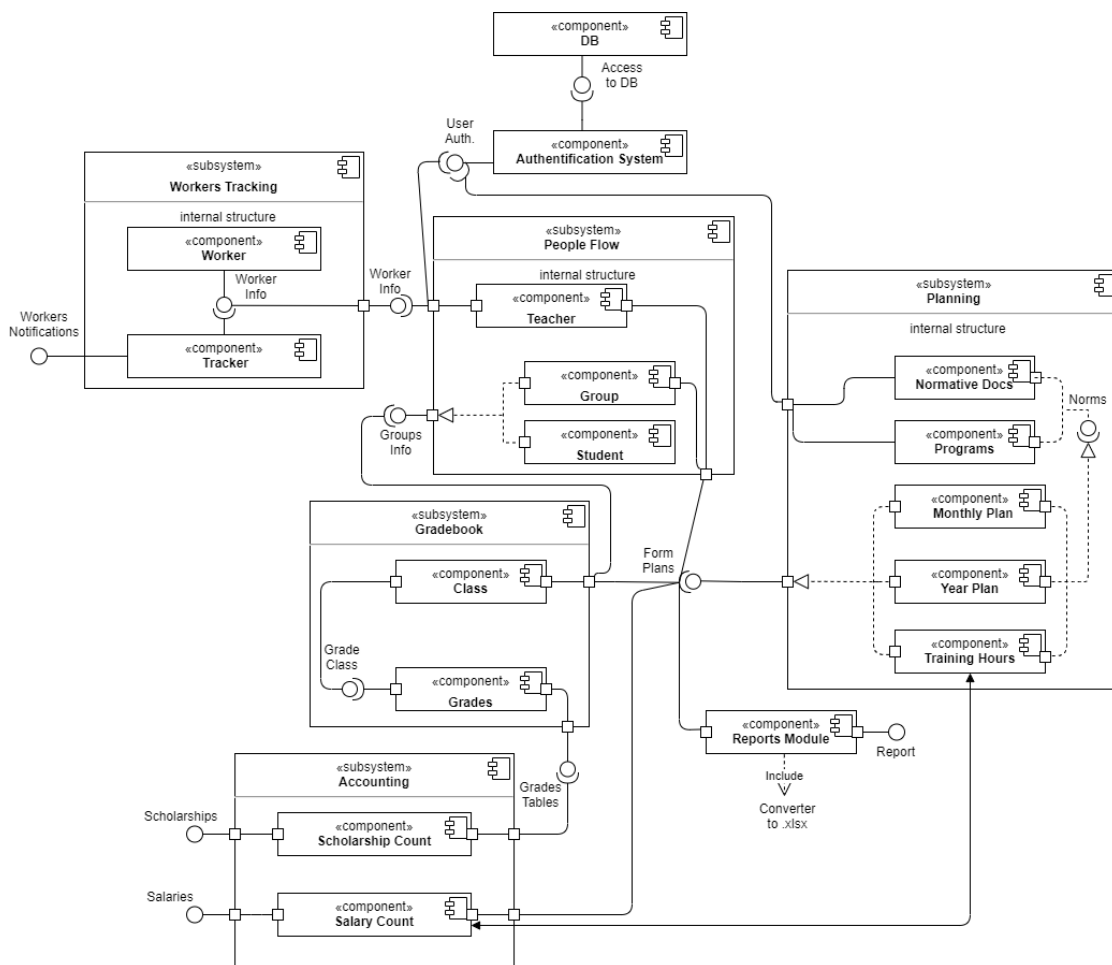


Рисунок 6 – Диаграмма компонентов

<<component>> DB – собственно, основная БД системы, предоставляет интерфейс доступа к БД, через который другие компоненты получают доступ к данным из базы.

<<subsystem>> Workers Tracking – данная подсистема включает компоненты Worker и Tracker, реализует два интерфейса – Workers Notifications (уведомления), Worker info (информация).

<<subsystem>> People Flow – буквально “поток людей”, включает компоненты Teacher, Group, Student. Именно эти компоненты фигурируют в связке при наполнении системы информацией об очередном потоке обучения. Подсистема требует интерфейсы Worker Info, Form Plans, User Auth., а сама предоставляет интерфейс Groups Info.

<<subsystem>> Planning – подсистема, отвечающая за планирование (формирование планов), включает компоненты Normative Docs, Programs (имеются в виду учебные программы), Monthly Plan, Year Plan, Training Hours. Требуется интерфейс User Auth., предоставляет интерфейс Form Plans.

<<subsystem>> Gradebook – электронный журнал, включает два компонента: Class и Gradebook. Данная подсистема требует интерфейс Form Plans (без формирования более общих планов не будет сформирован и электронный журнал), а реализует интерфейс Grades Tables.

<<subsystem>> Accounting – подсистема для учета финансов, а именно стипендий и зарплат на основании различных переменных, включает компоненты Scholarship Count, Salary Count. Accounting реализует интерфейсы Scholarships и Salary, требует наличия интерфейсов Grades Tables, Form Plans.

Смысл компонентов Authentication System и Reports Module очевиден из диаграммы.

3.1.8 Диаграмма классов

Диаграмма классов (Class Diagram) в UML – диаграмма, описывающая статическую структуру иерархии классов системы посредством демонстрации классов, их методов, атрибутов, принципов взаимодействия. Диаграмма, представленная далее (рисунок 7), отражает набор классов и связей между

ними, выявленный на этапе проектирования ИИС Технической школы. Логика, реализованная на этапе разработки системы, частично отличается от визуализированной на схеме.

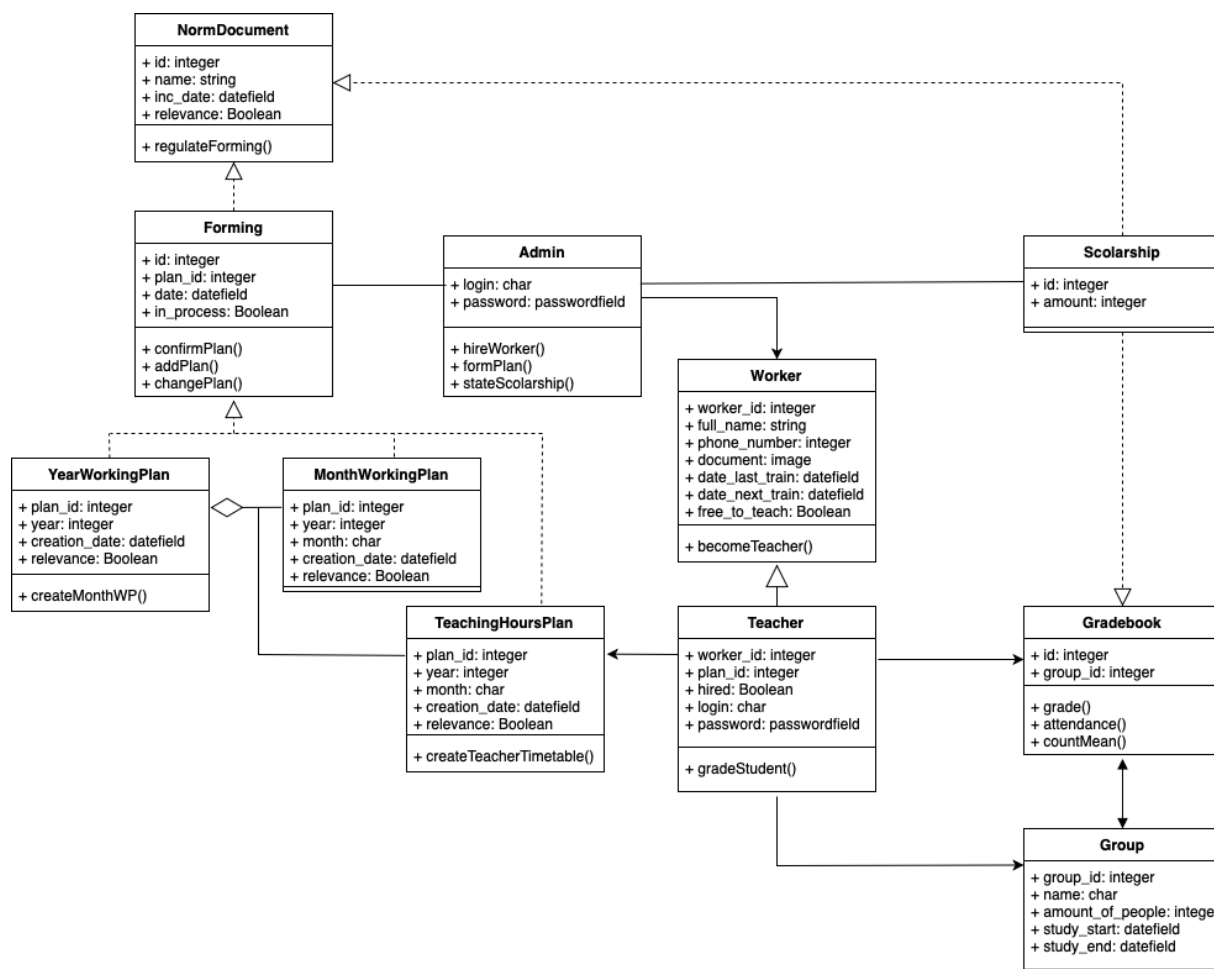


Рисунок 7 – Диаграмма классов

3.2 Проектирование базы данных

База данных (БД) – значимая часть веб-приложения. Структура и форма исходных данных (документов, ведомостей...) были проанализированы для проектирования базы данных, в которой они могли бы эффективно храниться. Ввиду большого размера полное изображение схемы спроектированной БД вынесено в приложение Б. В данном разделе будут приводиться ссылки на определенные места данной схемы. В БД есть отдельный блок таблиц, содержащих различные нормы, которые могут служить параметрами для обработки данных. Также присутствуют таблицы, в названии которых содержится Draft: в них предполагается хранить историю версий планов.

3.3 Модуль «БД Преподавателей»

К данному модулю относятся такие сущности БД как Worker, License, Teacher, Department, WorkerChanges, PersonalInfoChanges, Notification. Рассмотрим, какие роли выполняет каждая таблица.

Worker – непосредственно база сотрудников, имеющая содержащая данные сотрудников, которые ранее преподавали в Технической школе. За сроки прохождения обучения в ней отвечают два поля: last_training (последнее пройденное) и next_training (следующее плановое). Ранее было сказано, что проходить переподготовку преподавателю необходимо через 3 года, но такая структура таблицы позволяет указать и другую плановую дату. Отслеживание того, как скоро необходимо организовать прохождение обучения, происходит относительно даты next_training.

License – сущность для хранения лицензий сотрудников, с указанием принадлежности, даты получения, актуальности документа.

PersonalInfo – отдельная сущность для хранения персональных данных сотрудника. ФИО, телефонный номер и адрес электронной почты целенаправленно хранятся отдельно и соотносятся с Worker по Foreign Key (FK).

Department – информация о подразделении сотрудника также для удобства вынесена в отдельную таблицу.

Особый интерес представляет сущность WorkerChanges: она предназначена для хранения изменений в значениях полей объектов Worker. Эта сущность хранит ссылку на сотрудника, название измененного поля, его новое значение и дату изменения. Так возможно отслеживать изменения отдельных полей в данных сотрудника. Данный способ удобен для ситуаций, когда не планируется часто делать «откат» к предыдущим версиям. Аналогичная таблица с изменениями есть и для PersonalInfo.

Teacher – сущность, хранящая данные активных преподавателей подразделения, использует Worker и User. Не всем сотрудникам из Worker необходимо по умолчанию иметь аккаунт в системе, так как ей будут

пользоваться только те, кто в данный период вовлечён в образовательный процесс.

Notification – таблица уведомлений, которые касаются отслеживания сроков обучения преподавателей. Ссылается на Worker по FK, хранит текст уведомления, время его создания, степень важности.

«БД Преподавателей» предполагает реализацию следующего функционала:

- 1) CRUD модели Worker через интерфейс,
- 2) расширенная фильтрация,
- 3) отслеживание версий полей Worker и PersonalInfo,
- 4) отслеживание приближения сроков next_training с помощью уведомлений.

3.4 Модуль «Планирование»

Модуль предназначен для создания, хранения, просмотра планов. Так как в планировании учитываются многие переменные, здесь используются многие из моделей БД. Те, которые отвечают непосредственно за планы, представлены в таблице 2.

Таблица 2 – Виды планов и модели

Календарные планы	Учебные планы
CalendarPlan, InCP, CPDetails,	TrainingPlan, InTP, TPDetails

Также в рамках планирования часто происходит обращение к Program, CourseType, Subject, Group, Student.

Выстроить процесс обучения невозможно без обращения к учебным программам. В упрощённом виде Program представляет собой сущность учебной программы, которая содержит ряд предметов с указанием часов на их освоение. Предмет (Subject) и программа (Program) имеют связь Many-to-many через ассоциативную сущность InProgram. Program также имеет поле course_type соответствующее какому-либо виду обучения из таблицы CourseType (например, подготовка, переподготовка, инструктаж). Группы

(Group) тоже предполагается создавать на этапе планирования: объект содержит FK на программу обучения, названия, даты начала и окончания теоретической и практической части обучения, контингент студентов, статус (ведется ли сейчас обучение группы). Студенты имеют с группами связь Many-to-many через ассоциативную сущность Membership и должны быть добавлены в систему до начала обучения, чтобы преподаватели могли вести электронный журнал.

Календарный план вносится в систему и при необходимости обновляется. Пример строк плана и его нижней части показан на рисунке 8.

Календарный план проведения обучения в ОП "Техническая школа" на 2009 год

№	Наименование программы	Вид обучения	Кол-во чел.	Кол-во групп	январь	февраль	март	апрель	май
1	Машинист электропоезда	подготовка	70	3	0	0	0	0	1
2	Машинист электропоезда 1 класс	повышение квалификации	24	1	0	0	1	0	0
3	Машинист электропоезда 2 класс	повышение квалификации	48	2	0	0	0	0	0

	23	1	0	0	0	0	0	0	0	0	1	0	0	0
ИТОГО	5417	258	14	28	31	28	18	16	8	13	27	28	29	18
			14	42	73	101	119	135	143	156	183	211	240	258

Рисунок 8 – Пример из календарного плана

Такая структура реализуется с помощью трёх сущностей БД. CalendarPlan хранит базовую информацию: год, релевантность, дату и время создания записи. InCP ссылается на CalendarPlan и Program с указанием необходимого числа людей и групп (в каждой группе обычно не более 25 человек) для обучения по этой программе. На изображении указан и вид обучения, но он содержится в Program. CPDetails привязан к InCP и детализирует число групп по каждому. То есть, на один InCP могут ссылаться до 12 CPDetails.

Учебный план составляется непосредственно в Технической школе и выглядит следующим образом (рисунок 9):

План подготовки и повышения квалификации рабочих кадров на 2022 год в Технической школе														
Наименование и группы	Сроки обучения													Итого
	январь	февраль	март	апрель	май	июнь	июль	август	сентябрь	октябрь	ноябрь	декабрь		
	Основные программы профессионального обучения по ученическим договорам													
Программа 1, АБІ-123	68												68	
...	64							68					132	
	208	256	294				52	126			68		1004	
								248	368	316	116		1048	
											8	328	336	
								128	176	176	84	58	622	
			24				40	28					92	
			80				12						92	
		66	56										122	
		32	96										128	
			24				40	64					128	
Итого по разделу	340	354	574	0	0	0	144	662	544	492	276	386	3772	

Рисунок 9 – Пример из учебного плана

Таблицы БД моделируют эту структуру. Шапка плана с базовой информацией о нем, данными о релевантности документа, отметкой редактирования, хранится в TrainingPlan. Также есть ссылка по FK на календарный план, который послужил основой для учебного. InTP хранит группу (уже содержит в себе программу), FK на TrainingPlan (составной частью которого является), отметку о редактировании. По сути, InTP это элементы плана, но не имеющие пока информации о числе часов по месяцам. Наконец, TPDetails детализирует InTP, ссылаясь на него и храня комбинацию из месяца и часов. Таким образом, каждый TrainingPlan имеет некоторой количество InTP (One-to-many), а каждый InTP – до двенадцати экземпляров TPDetails.

Также требованием к системе является хранение версий изменяемых документов, чтобы иметь возможность возвращаться к предыдущим вариантам. Версии учебных планов должны добавляться при редактировании основных таблиц во вспомогательные DraftTP, DraftInTP, DraftTPDetails.

Основной задачей является заполнить модели учебного плана таким образом, чтобы они соответствовали моделям календарного. В качестве дополнительной была поставлена о реализации полуавтоматического планирования. Интерактивность заполнения планов позволила бы сэкономить время на данном этапе основного бизнес-процесса.

3.5 Модуль «Учёт часов»

В качестве входных данных были предоставлены два вида ведомостей учёта часов: за год по всем преподавателям (рисунок 10) и за месяц по преподавателям конкретной группы (рисунок 11).

	A	B	C	D	E	V	W	X	Y	Z	AA	AB	AC
1	Учет часов работы преподавателей в 2020 году												
2	ФИО преподавателя	Январь		Февраль		Ноябрь		Декабрь		Итого часов за год	В личное время	Часы, входящие в лимит	Лимит часов
3	Александр Александрович	8	8	16	8					80	40	40	180
4	Александр Александрович					55				55	0	55	180
157	Михаил Михайлович	47		8						165	0	165	280
158	Ирина Ивановна									22	0	22	180
159													
160	*жирный шрифт - в личное время	523	187	989	292	1101	199	0	0	7217	1848	5369	

Рисунок 10 – Ведомость учёта часов в год

ВЕДОМОСТЬ																
учета часов работы преподавателей за ноябрь 2020 года																
Группа Машинисты электропоезда № 11-217																
№ п/п	Предметы	ФИО преподавателя	Числа месяца												Дано часов за месяц	Часовая тариф, ставка, руб
			1	2	3	4	5	6	7	8	9	10	31			
1	Электрическое оборудование (модуль 2)	Мухомов Евгений Иванович									1			1	435*	
2	Электрическое оборудование (модуль 5)	Мухомов Евгений Иванович					3				1			4	348	
3	Пневматическое оборудование (модуль 5)	Мухомов Евгений Иванович			1		5				1			7	435*	
17	Устранение неисправностей	Мухомов Евгений Иванович			8									8	435*	
18	Устранение неисправностей	Мухомов Евгений Иванович			8									8	348	
19	Устранение неисправностей	Мухомов Евгений Иванович			8									8	435*	
Итого по предметам:														80		
20	Экзамен	Мухомов Евгений Иванович										5		5	348	
21	Экзамен	Мухомов Евгений Иванович											4	4	348	
26	Экзамен	Мухомов Евгений Иванович										4		4	348	
27	Экзамен	Мухомов Евгений Иванович											4	4	348	
Итого за экзамены:														36		
Всего:														116		
* Занятия проводятся в свободное от основной работы время.																

Рисунок 11 – Ведомость учёта часов за месяц

Было принято решение обобщить оба вида в одной таблице TrainingHours. Сущность TrainingHours ссылается по FK на группу, преподавателя, предмет. К собственным полям таблицы относятся time_time (тип времени, личное или рабочее), дата, часы (число часов преподавания в этот день), отметку о редактировании записи (автор и время). Часы, входящие только в рабочее время, имеют лимит, указанный в самом крайнем столбце, и не должны его превышать. В основном лимит равен 180 часам, но в некоторых случаях расширяется до 280. От того преподавал сотрудник в рабочее время или в свое личное, зависит его ставка, что учитывается в модуле «Расчёты». Регулирующие учёт часов нормы находятся в таблицах HoursNorm (регламентирует лимиты преподавания в день, месяц, год) и WorkingDates

(должна заполняться датами с указанием, какой это день, и эквивалентом в часах).

При такой конструкции таблицы, из нее может быть легко выведена как годовая ведомость, так и ведомость по группе за месяц. Учёт часов близок по смыслу к этапу формирования планов, но всё же был вынесен в отдельный модуль. Разрабатываемое веб-приложение использует TrainingHours не только для получения итоговой сводки по часам преподавания, но и для привязки преподавателя к группе, что является элементом планирования.

Таким образом, основные элементы, требующие реализации в данном модуле:

- 1) CRUD TrainingHours,
- 2) отслеживание лимитов (норм),
- 3) получение итоговых ведомостей с фильтрацией и базовыми расчётами.

3.6 Модуль «Электронный журнал»

Электронный журнал – классический элемент информационной системы образовательного учреждения. Если студенты добавлены в группы на этапе планирования, а преподаватели имеют записи в «учёте часов», можно создать объекты сущности Class (Занятие) с соответствующими параметрами. Говоря иначе, сформировать расписание занятий. Таблица БД Class ссылается по FK на Group, Teacher, Subject. У каждого занятия есть дата и время начала, длительность в часах (целые значения), тип занятия class_type (хранятся в таблице ClassType. Большая часть занятий является обычными занятиями (уроками) либо экзаменами, но список типов может быть кастомизирован. Для Class, который является экзаменом, можно создать относящиеся к нему по FK записи в таблице Comission (Комиссия). Comission сосылется на Class через поле exam и на сотрудника (Worker, не Teacher) через поле worker.

Рассмотрим, как организован контроль за посещаемостью и оценками. В БД предусмотрена сущность Grade, выполняющая функции как оценки, так и отметки о посещаемости. Grade имеет поля class_id (FK на Class) и student (FK

Student). Поля created_at и user (FK User) позволяют отслеживать, кто и когда добавил запись. Поле grade_type принимает два значения: «grade» и «attendance». В зависимости от указанного типа система будет воспринимать объект Grade как оценку либо отметку посещаемости. Далее следуют численное поле grade и бинарное attendance. В расчётах будет использоваться то поле, которое соответствует типу. Таким образом, если преподаватели будут своевременно проставлять баллы за свои занятия студентам в системе, администратор будет получать доступ к полным данным намного быстрее, чем это происходит в привычном аналоговом формате. В рамках модуля «Электронный журнал» должно быть возможно в любой момент учебного процесса отобразить текущие посещаемость и средние баллы студентов, информация о которых уже заполнена.

Таким образом, основные задачи при разработке данного модуля:

- 1) CRUD Class и фильтрация,
- 2) объекты Class согласуются с данными TrainingHours,
- 3) просмотр расписания занятий (своих или всех),
- 4) ведение оценок с указанием авторства,
- 5) базовые расчёты (для каждого студента – средние баллы по предметам, процент посещений от общего числа часов плановых занятий).

3.7 Модуль «Расчёты»

Смысл данного модуля следует из названия. Имеются в виду расчёты стипендий студентам и зарплат преподавателям. Переменные, влияющие на осуществление этих операций, должны извлекаться из «Положения о стипендии студентам» и «Положения об оплате преподавателей» и храниться в таблицах БД ScholarshipNorm и SalaryNorm соответственно. Примерный вид нормы зарплаты преподавателя представлен в таблице 3.

Помимо квалификации преподавателя (education_level) и контингента учащихся группы, на ставку влияют параметры из «учёта часов»:

- 1) непосредственно число данных часов занятий,

2) тип времени, в которое преподаватель вёл занятия (личное время рассчитывается по ставке на уровень выше).

Итоговая зарплата высчитывается путем сложения произведений часов на ставку.

Таблица 3 – Примерный вид нормы зарплаты

Квалификация преподавателя	Контингент учащихся	Часовая тарифная ставка, у. е.
СПО	Рабочие и учащиеся Технической школы	1
Высшее проф. образование	Рабочие и учащиеся Технической школы	2
Высшее проф. образование	Руководители среднего звена, специалисты, служащие	3
Высшее проф. образование	Руководители подразделений	4

Стипендия студентов зависит от программы обучения, взвешенных средних баллов по предметам и процента посещаемости.

Алгоритм расчёта:

1) рассчитывается взвешенный средний балл студента – среднее арифметическое средних баллов по всем предметам,

2) рассчитывается % посещаемости = посетил часов / всего часов в расчетном месяце,

3) в зависимости от программы и среднего балла выбирается один вариант тарифа,

4) наконец, стипендия = тариф * % посещаемости.

Из данного ранее в тексте работы описания модуля «Электронный журнал» очевидно, что данные о среднем балле и посещаемости для реализации функционала модуля «Расчёты» необходимо будет получать из него.

3.8 Модуль «Документы»

Модуль «Документы» должен служить входной точкой для всех поступающих извне документов (планов, норм, программ, лицензий), давать возможность упорядоченно хранить, просматривать, фильтровать (например, по релевантности, типу или дате загрузки) загруженные файлы. Ранее уже было сказано, что параметры для обработки данных системы должны храниться в таблицах БД системы. В любой момент их можно обновить вручную. В данном же модуле может быть использованы какие-либо библиотеки для парсинга данных из форматов DOCX, XLSX, PDF или распознавания текста со сканов в PDF и JPG. Всё, что было автоматически извлечено, может быть проверено вручную администратором перед добавлением в таблицы.

Вторая причина для существования такого модуля – добавление в систему функции автоматического формирования отчётов. Сделав запросы к API, возможно получить от модулей объекты с данными в формате JSON, а затем определённым образом внести их в нужные шаблоны. В задаче могут использоваться библиотеки python для работы с файлами DOCX и XLSX.

Выводы по третьей главе

Были разработаны диаграммы и схемы, позволяющие с разных сторон взглянуть на устройство ИИС. Была детально рассмотрена логика каждого из модулей. Описания и изображения, приведенные в разделе «проектирование системы», послужили руководством для программной реализации ИИС и её модулей.

4 РАЗРАБОТКА МОДУЛЕЙ СИСТЕМЫ

4.1 Статус разработки модулей системы

Разработка комплексной информационной системы – длительный ресурсоёмкий процесс. Поэтому в рамках выпускной квалификационной работы часть приведённых ранее модулей была разработана, а часть – только спроектирована и описана. Статус разработки каждого из модулей указан в таблице 4.

Таблица 4 – Статус разработки модулей

Модуль	Статус
БД Преподавателей	Разработан
Планирование	MVP, предложены варианты будущей интеллектуализации
Учёт часов	MVP
Электронный журнал	Разработан
Расчёты	MVP
Документы	Описан и спроектирован

MVP (minimum viable product) – это минимально жизнеспособный продукт.

4.2 Используемые в проекте технологии и принятые решения

Бэкенд веб-приложения написан на языке программирования Python (v3.8.5) [15] с использованием популярного фреймворка Django (v3.2) [1]. Для создания API использовался Django REST framework (v3.12.4) [2]. В качестве СУБД выбрана PostgreSQL (psql 12.6, server 10.14) [3].

Модуль Django-jazzmin [17] подключен для более свободной кастомизации админ-панели. К примеру, чтобы так называемые Custom Views – собственные представления моделей – поддерживали Bootstrap 4 [16]. Для того, чтобы получить приемлемые результаты в ограниченные сроки разработки, было принято решение уделить больше внимание внутренней логике приложения и его серверной части. Следовательно, основные

интерфейсы, представления данных таблиц созданы с помощью класса `generics.TemplateView` с применением `html`-шаблона и встроены в админ-панель. Предполагается, что разработанный бэкенд рассматриваемой системы будет в будущем расширен и использован для написания отдельного фронтенда с применением какого-либо JS фреймворка, например, `Vue.js`.

Все модели БД описаны в `models.py` основного приложения `tech_school_app`. Однако, остальная логика разделена по отдельным приложениям: основной Django project “`tech_school_backend`” содержит в себе ряд Django apps. В ходе разработки рассматривался и вариант разделения классов моделей по приложениям, но не было принято однозначного решения о принадлежности моделей к модулям. В нём же реализована авторизация пользователей системы по токену с помощью `Djoser`. По задумке, новые пользователи-преподаватели (модель `User`) добавляются администратором через админ-панель (рисунок 12), так как они привязаны к `Worker`, но есть адреса API для авторизации.

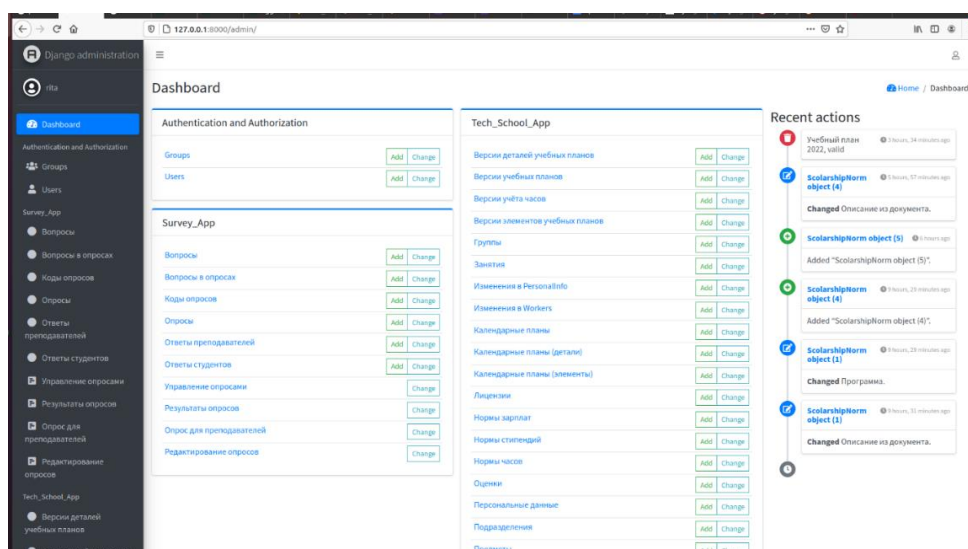


Рисунок 12 – Админ-панель приложения

Во многих моделях используются Django Lifecycle Hooks, или «хуки» жизненного цикла (ЖЦ) [7]. Эта библиотека предоставляет удобную альтернативу встроенным в Django сигналам. Смысл в том, что описанный «хук» реагирует на определенное событие (например, обновление объекта) в ЖЦ модели и запускает выполнение указанного действия. Для применения

lifecycle hooks в модели, она должна наследоваться от класса LifecycleModel. Примеры использования будут описаны далее в тексте работы.

4.3 Модуль «БД Преподавателей»

Модулю соответствует приложение workersdb_app и служит прохождению такого этапа бизнес-процесса, как актуализация данных преподавателей. Взаимодействие с моделями модуля в полной мере реализуется через админ-панель Django (рисунок 13): в admin.py кастомизированы классы admin.ModelAdmin (например, выбраны поля для отображения, фильтрации, поля readonly).

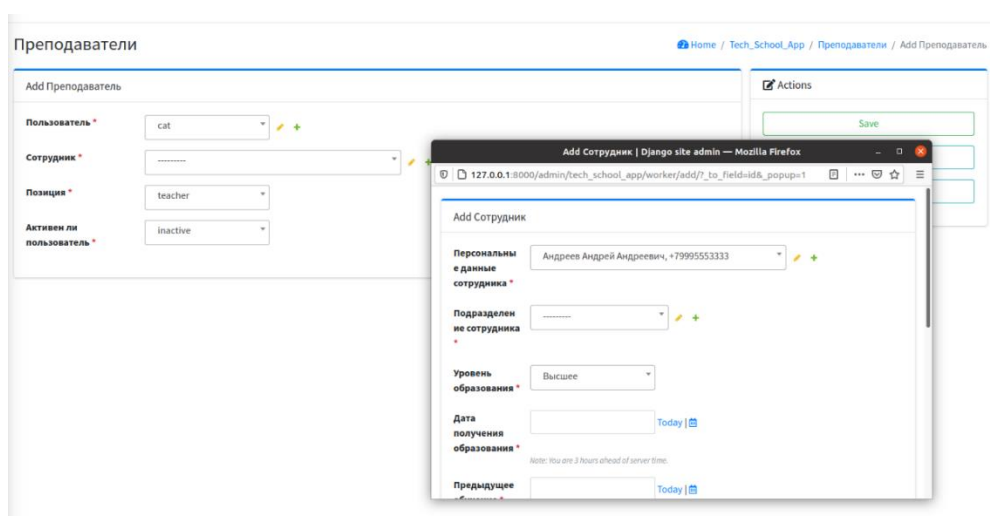


Рисунок 13 – Добавление преподавателя

Однако, это не единственный способ взаимодействия. Django REST framework был использован для создания API. В serializers.py добавлены сериализаторы (рисунок 14), как базовые, так и расширенные, где поля с Foreign Key также сериализованы.

```
class WorkerSerializer(serializers.ModelSerializer):
    class Meta:
        model = Worker
        fields = '__all__'

class DetailWorkerSerializer(WorkerSerializer): # расширенный Worker
    personal_info = PersonalInfoSerializer(read_only=True)
    department = DepartmentSerializer(read_only=True)
    education_level = serializers.CharField(source="get_education_level_display")
    available = serializers.CharField(source="get_available_display")
```

Рисунок 14 – Пример сериализаторов Worker

Во views.py приведены необходимые отображения для всех операций с моделями, в том числе для фильтрации Worker и License (рисунок 15).

```
class FilterLicensesView(generics.ListAPIView):
    serializer_class = DetailLicenseSerializer

    def get_queryset(self):
        queryset = License.objects.all()

        params = self.request.query_params

        name = params.get('name', None)
        upload1 = params.get('upload1', None)
        upload2 = params.get('upload2', None)
        relevance = params.get('relevance', None)
        worker = params.get('worker', None)

        if name:
            queryset = queryset.filter(name__contains=name)

        if upload1: # upload date после этой даты
            queryset = queryset.filter(last_training__gte=upload1)

        if upload2: # upload date до этой даты
            queryset = queryset.filter(last_training__lte=upload2)

        if relevance:
            queryset = queryset.filter(relevance=relevance)

        if worker:
            queryset = queryset.filter(worker__id=worker)

        return queryset
```

Рисунок 15 – ListAPIView для фильтрации License.

ViewSet Django REST framework совмещают в себе набор из нескольких view для одного класса (рисунок 16).

```
class WorkerViewSet(viewsets.ModelViewSet):
    queryset = Worker.objects.all()

    def get_serializer_class(self):

        if self.action in ["list", "retrieve"]:

            return DetailWorkerSerializer

        return WorkerSerializer
```

Рисунок 16 – ViewSet для Worker

Адреса API workersdb_app/ указаны в urls.py (рисунок 17).

```
urlpatterns = [
    path('workers/all', WorkerViewSet.as_view({"get": "list"})),
    path('workers/<int:pk>', WorkerViewSet.as_view({"get": "retrieve"})),
    path('workers/update/<int:pk>', WorkerViewSet.as_view({"patch": "partial_update"})),
    path('workers/create', WorkerViewSet.as_view({"post": "create"})),
    path('teachers/all', TeacherViewSet.as_view({"get": "list"})),
    path('teachers/<int:pk>', TeacherViewSet.as_view({"get": "retrieve"})),
    path('teachers/update/<int:pk>', TeacherViewSet.as_view({"patch": "partial_update"})),
    path('teachers/create', TeacherViewSet.as_view({"post": "create"})),
    path('wchanges/<int:worker>', WorkerChangesView.as_view()),
    path('pchanges/<int:worker>', PersonalInfoChangesView.as_view()),
    path('personalinfo/update/<int:pk>', PersonalInfoViewSet.as_view({"patch": "partial_update"})),
    path('personalinfo/create', PersonalInfoViewSet.as_view({"post": "create"})),
    path('departments/update/<int:pk>', DepartmentViewSet.as_view({"patch": "partial_update"})),
    path('departments/create', DepartmentViewSet.as_view({"post": "create"})),
    path('workers/check/<int:delta>', CheckWorkersDataView.as_view()),
    path('workers/filter', FilterWorkersView.as_view()),
    path('workers/notifications/create', CreateNotificationView.as_view()),
    path('workers/license/all', FilterLicensesView.as_view()),
    path('workers/license/update/<int:pk>', LicenseViewSet.as_view({"patch": "partial_update"})),
    path('workers/license/create', LicenseViewSet.as_view({"post": "create"})),
]
```

Рисунок 17 – Адреса API workersdb/

На скриншотах ниже (рисунок 18, 19) отображены два возможных способа фильтрации сотрудников.

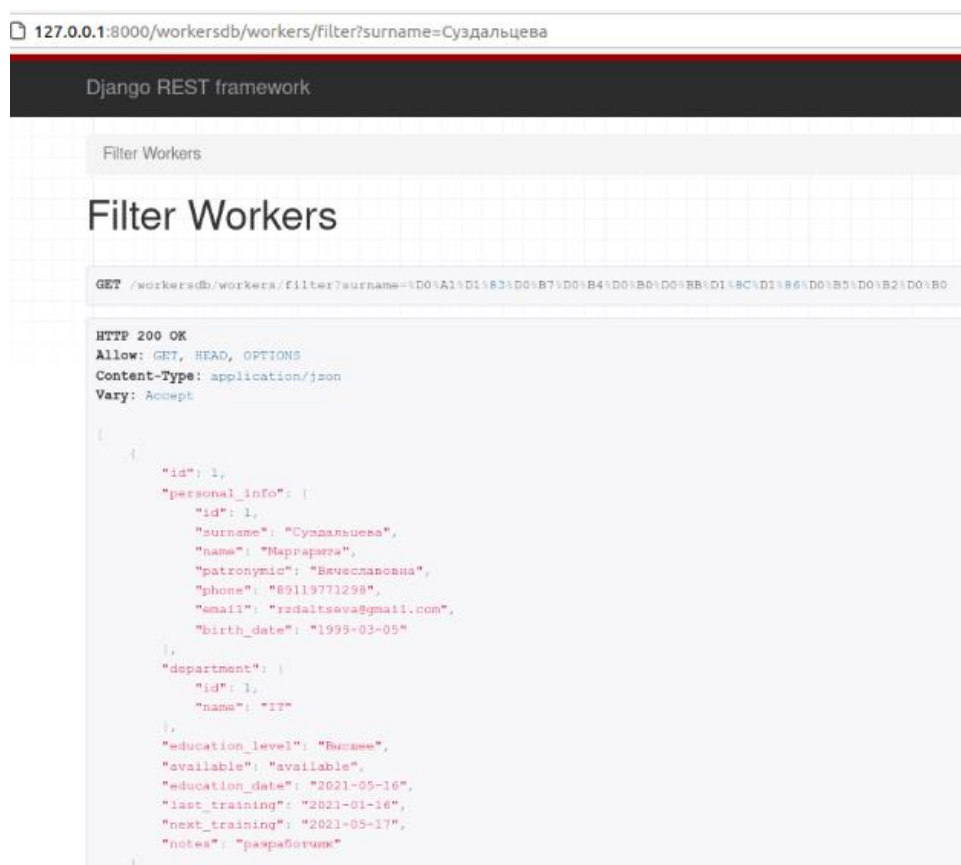


Рисунок 18 – Фильтрация сотрудников по фамилии с помощью API

Сотрудники

Home / Tech_School_App / Сотрудники

Select Сотрудник to change

Суздальцева | Название подразделения | Уровень образования

Дата получения образования | Предыдущее обучение | Следующее обучение

Заметки | Доступен для преподавания | Search 1 result (3 total)

Go 0 of 1 selected Add Сотрудник

<input type="checkbox"/>	Персональные данные сотрудника	Подразделение сотрудника	Уровень образования	Дата получения образования	Предыдущее обучение	Следующее обучение	Заметки	Доступен для преподавания
<input type="checkbox"/>	Суздальцева Маргарита Вячеславовна, 89119771298	Подразделение "ИТ"	Высшее	May 16, 2021	Jan. 16, 2021	May 17, 2021	разработчик	available

1 Сотрудник

Рисунок 19 – Также фильтрация сотрудников, но через админ-панель

Каждому преподавателю можно добавить лицензии в таблице License (рисунок 20). Они доступны для просмотра, редактирования и фильтрации. По ссылке открывается загруженное изображение, хранящееся в поле типа ImageField.

Лицензии

Home / Tech_School_App

Select Лицензия to change

Название лицензии | Дата загрузки | Релевантность

Сотрудник | Search

Go 0 of 2 selected Add Ли

<input type="checkbox"/>	Название лицензии	Дата загрузки	Изображение	Релевантность	Сотрудник
<input type="checkbox"/>	Новая лицензия	May 29, 2021, 12:11 p.m.	media/licenses/itmo_logo_transp_en.png	relevant	Суздальцева Маргарита Вячеславовна, 89119771298
<input type="checkbox"/>	Логотип ИТМО	May 28, 2021, 12:06 p.m.	media/licenses/itmo_logo_blue_en.jpg	relevant	Иванов Иван Иванович, +79191919191

Рисунок 20 – Лицензии

Некоторые из URL-адресов этого модуля являются частью схемы получения уведомлений об актуальности данных сотрудников. Был написан скрипт `python notification_script.py` (приложение В), выполняющий необходимую последовательность действий, схема его работы – на рисунке 21.

Введён параметр `delta` – число дней, которое используется для проверки сроков. По умолчанию предложены такие варианты значений: 30 дней (месяц), 90 дней (3 месяца), 180 дней (около полугода). В файле даны шаблоны текстов уведомлений, словарь соответствия значения `delta` и важности уведомления, а также функции, отправляющие необходимые запросы к API. `Delta` принимается как аргумент командной строки при запуске скрипта. Скрипт

нужно запускать последовательно со всеми тремя значениями по убыванию важности (значение наибольшей важности в приоритете).

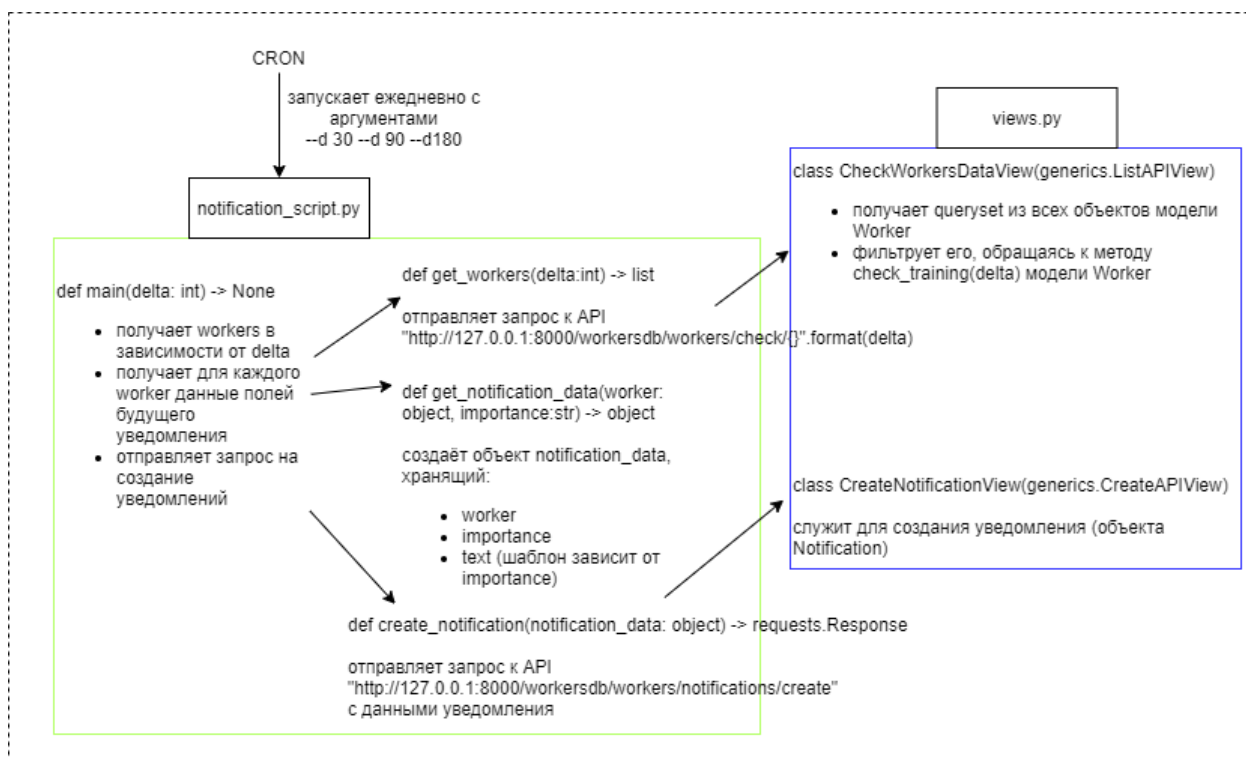


Рисунок 21 – Схема работы notification_script.py

Автоматизировать регулярное (ежедневное) выполнение скрипта при развертывании системы планируется с помощью демона cron. Чтобы ограничить число одинаковых уведомлений с маленькой разницей в дате добавления, у модели Notification был переопределен метод save() и добавлены два метода same_recent и same_today. Объект не сохраняется, если в последнюю неделю/день уже создавалось подобное уведомление.

В def check_training(self) класса модели Worker (models.py) рассчитывается разница (current_delta) между текущей датой и той датой, в которую преподавателю необходимо пройти следующее обучение (рисунок 22). Если разница меньше или равно delta, до даты обучения осталось меньше delta дней, а значит нужно организовать прохождение переобучения преподавателем. Когда даты последнего актуального и ближайшего следующего обучения будут изменены, скрипт не будет реагировать на преподавателя, пока не подойдут новые сроки.

```

def check_training(delta): # получение queryset из преподавателей
    current_date = datetime.datetime.now().date()

    queryset = []

    for row in Worker.objects.all():
        current_delta = row.next_training - current_date

        if current_delta.days <= delta:
            queryset.append(row)

    return queryset

```

Рисунок 22 – Метод check_training(delta)

Уведомления добавляются с помощью API и рассылаются на почту с помощью «хука» AFTER_CREATE модели Notification. Рассылка настроена на базе почтового сервиса mail.ru с помощью django.core.mail.

Для реализации хранения версий полей Worker также были использованы «хуки» AFTER_CREATE и AFTER_UPDATE, создающие запись в WorkerChanges при создании или обновлении объекта Worker (рисунок 23). Аналогичный алгоритм применяется для хранения изменений таблицы PersonalInfo.

```

@hook(AFTER_CREATE)
@hook(AFTER_UPDATE)
def update_worker_changes(self):
    worker = Worker.objects.filter(pk=self.pk).values()[0]

    for field in worker:
        if self.has_changed(field):
            changes = WorkerChanges(
                worker=self,
                field_name=field,
                new_value=worker[field]
            )

            changes.save()

```

Рисунок 23 – «хуки» для сохранения версий полей Worker

Записи в таблице изменений доступны только для просмотра, как через админ-панель (рисунок 24), так и через API.

Изменения в Workers		Home / Tech_School_App / Изменения в V
Change Изменение в Workers		
Изменение в Workers	Андреев Андрей Андреевич, +79995553333, изменено 2021-05-28 12:25:04.095404+00:00	
Worker	Андреев Андрей Андреевич, +79995553333	
Field name	available	
New value	1	
Changed at	May 28, 2021, 12:25 p.m.	

Рисунок 24 – Запись в таблице WorkerChanges

4.5 Модуль «Планирование»

4.5.1 MVP

Модулю соответствует приложение `planning_app`. В MVP предусмотрены все необходимые структуры данных и создано базовое API, но редактирование планов производится с помощью интерфейса расширенной админ-панели Django, а не с помощью API либо интерфейсов Custom View.

На данном этапе администратор должен заполнить таблицы календарных планов `CalendarPlan`, `InCP`, `CPDetails`. Затем могут быть добавлены программы обучения с предметами (если появились новые), созданы группы этих программ. Наконец, должны быть заполнены таблицы учебных планов `TrainingPlan`, `InTP`.

Соответствие данных составляемого учебного плана значениям актуального календарного плана проверяется с помощью валидации в Django.

Приведём примеры. Одним из написанных методов модели `InTP` является `def program_n_times(self)`, сравнивающих число групп программы, добавленных в учебный план, с нормативным числом групп этой программы из календарного плана (рисунок 25). Другой метод `def program_in_CP(self)` проверяет, есть ли вообще указываемая в учебном плане программа в исходном календарном.

```

def program_n_times(self):
    program = self.group.program.id
    cp = self.trainingplan.calendarplan
    incps = InCP.objects.filter(calendarplan=cp)
    intps = InTP.objects.filter(trainingplan=self.trainingplan)
    n = getattr(incps.filter(program=program)[0], 'groups_number') # нормативное число групп для программы
    m = len(intps.filter(group__program_id=program)) # сколько уже групп этой программы
    if not self.id:
        if n + 1 <= m:
            return True
        else:
            return False
    else:
        if n <= m:
            return True
        else:
            return False

```

Рисунок 25 – Валидатор для строк учебного плана.

Метод `clean()` `InTP` переопределен для проверки полей формы (рисунок 26). Если для объекта будут получены значения `False`, будут вызваны ошибки валидации, предотвращающие отправку формы с неверными значениями (рисунок 27).

```

def clean(self, *args, **kwargs):
    if not self.program_in_CP():
        raise ValidationError('Вы пытаетесь добавить в учебный план группу с программой, кото
    if not self.program_n_times():
        raise ValidationError('Нельзя добавить в учебный план больше групп этой программы, че

```



Рисунок 26 – Переопределенный метод `clean()`



Учебные планы (элементы)

Please correct the error below.

❗ Вы пытаетесь добавить в учебный план группу с программой, которая не предусмотрена в календарном плане.

Add Учебный план (элемент)

Группа *  

Учебный план *  

Последний редактировал -

Дата и время создания -

Рисунок 27 – Запись не прошла валидацию.

Для наглядного просмотра планов были созданы несколько собственных представлений моделей, расширяющих админ-панель (рисунок 28, 29).

Календарный план

Отобразить план:

Фильтровать Сброс

Календарный план

Год 2021
Релевантность relevant
Создан May 20, 2021, 1:26 p.m.

тип обучения (программы)

Программа	Вид обучения	Кол-во чел.	Кол-во групп	январь	февраль	март	апрель	май	июнь	июль	август	сентябрь	октябрь	ноябрь	декабрь
Программа "машинист электропоезда"	подготовка	70	3	0	0	0	0	1	0	0	0	1	0	0	1
Программа "машинист электропоезда 1 класс"	повышение квалификации	24	1	0	0	1	0	0	0	0	0	0	0	0	0
Программа "машинист электропоезда 2 класс"	повышение квалификации	48	2	0	0	0	0	0	0	0	0	1	0	1	0
ИТОГО		142	6	0	0	1	0	1	0	0	0	2	0	1	1

Рисунок 28 – Интерфейс «Календарный план (таблица)»

Учебный план

Отобразить план:

Фильтровать

Сброс

Учебный план

Год

2021

Статус

active

Релевантность

valid

Создан

May 23, 2021, 6:04 p.m.

Соответствующий календарный план

Календарный план 2021, relevant

тип обучения (программы)

Программа	Группа	январь	февраль	март	апрель	май	июнь	июль	август	сентябрь	октябрь	ноябрь	декабрь	ИТОГО
Программа "машинист электропоезда"	Группа М-231	0	0	0	0	0	0	0	0	0	0	0	68	68
Программа "машинист электропоезда 1 класс"	Группа МЭВПК-11	0	0	64	0	0	0	0	0	0	0	0	0	64
Программа "машинист электропоезда 2 класс"	Группа МЭВПК-13	0	0	0	0	0	0	0	0	0	0	30	30	60
Программа "машинист электропоезда 1 класс"	Группа МЭВПК-12	0	0	0	0	0	0	0	60	0	0	0	0	60
Программа "машинист электропоезда"	Группа М-230	0	0	0	0	0	0	0	68	0	0	0	0	68
Программа "машинист электропоезда"	Группа М-229	0	0	0	0	68	0	0	0	0	0	0	0	68
ИТОГО		0	0	64	0	68	0	0	0	128	0	30	98	388

Рисунок 29 – Интерфейс «Учебный план (таблица)»

Создание Custom View для админ-панели требует выполнения следующих шагов:

- 1) добавление дополнительных адресов админ-панели в настройках модуля jazzmin в settings.py проекта, а также адресов директорий с шаблонами html-страниц (templates),
- 2) написание необходимых представлений, наследующихся от класса generics.TemplateView во views.py приложения,
- 3) создание соответствующего template, в который бы передавался объект, сформированный в TemplateView, при подключенном jazzmin на данном этапе можно использовать компоненты и стили bootstrap,
- 4) опционально – использование в template JavaScript,
- 5) указание адресов созданных views при регистрации модели в админ-панели.

Рассмотрим view для отображения учебного плана. Первая часть кода представлена на рисунке 30.

```
class TrainingPlanView(TemplateView):
    template_name = "trainingplan.html"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context.update(site.each_context(self.request))

        context["headings"] = [
            'Программа',
            'Группа',
            'январь',
            'февраль',
            'март',
            'апрель',
            'май',
            'июнь',
            'июль',
            'август',
            'сентябрь',
            'октябрь',
            'ноябрь',
            'декабрь',
            'ИТОГО'
        ]

        plans = TrainingPlan.objects.all()
        context["plans"] = plans

        ctypes = CourseType.objects.all()
        context["ctypes"] = ctypes

        tplan = self.request.GET.get("tplan")
        ctype = self.request.GET.get("ctype")

        trainingplans = TrainingPlan.objects.all()
        trainingplan = trainingplans.filter(relevance='1')[0]
```

Рисунок 30 – TrainingPlanView (1 часть)

В `get_context_data` формируется объект, который будет передан в `html-шаблон`. `context['headings']` содержит заголовки таблицы плана. Для возможности отображения конкретного плана и фильтрации строк по виду обучения с помощью JS, `context` содержит все объекты таблиц `TrainingPlan` (`plans`) и `CouseType` (`ctypes`). В свою очередь, переменные `tplan` и `ctype` хранят параметры, полученные от пользователя, который отправляет GET-запрос. Переменная `trainingplan` по умолчанию хранит последний релевантный план.

```

if tplan:
    trainingplan = trainingplans.filter(id=tplan)[0]

intps = InTP.objects.filter(trainingplan=trainingplan)

if ctype:
    intps = intps.filter(group__program__course_type=ctype)

hours_sum = intps.values('group').annotate(x_sum=Sum('tpdetails__hours'))
months_sum = [{ 'month': f'{k}', 'sum': 0 } for k in range(1, 13)]
total_sum = intps.aggregate(total=Sum('tpdetails__hours'))['total']

rows = []

for intp in intps:
    _row = {}
    _row["intp"] = intp
    _row["x_sum"] = hours_sum.filter(group=intp.group)[0]['x_sum']

    tpds = TPDetails.objects.filter(intp=intp).order_by('month')
    _row["tpds"] = []

    for m in range(1, 13):
        tpd = tpds.filter(month=str(m))
        num = getattr(tpd[0], 'hours') if tpd else 0
        months_sum[m-1]["sum"] += num
        _row["tpds"].append({'month': str(m), 'hours': num})

    rows.append(_row)

plan = {'tp': trainingplan, 'months_sum': months_sum, 'total_sum': total_sum, 'rows': rows}
context['plan'] = plan

return context

```

Рисунок 31 – TrainingPlanView (2 часть)

Перейдем ко второй части кода (рисунок 31). Если получено значение `tplan`, `trainingplan` принимает объект того плана, который был выбран: происходит фильтрация `queryset` из всех учебных планов по `id` выбранного плана. Стоит упомянуть, что Django содержит множество различных инструментов для фильтрации и выполнения запросов к моделям БД. В переменную `intps` сохраняется результат запроса к `InTP` с фильтрацией по полю `trainingplan`. Иначе говоря, она хранит все детали учебных планов, который ссылаются на нужный учебный план по FK. Если задан `ctype`, среди

intps фильтруются те, которые содержат группу, обучающуюся по программе выбранного типа.

Далее используются такие возможности Django как агрегация и аннотирование. Из `django.db.models` импортированы операторы Avg, Count, Q, Sum. Выражение, присвоенное `hours_sum`, с помощью `annotate` группирует `intps` по значению `group`, при этом рассчитывая сумму часов (`hours`), указанных в смежной таблице `CPDetails`. Полная сумма без группировки по группам рассчитана с помощью `aggregate` и `Sum` того же поля `hours`. Переменная `months_sum` задаёт список, хранящий словари с ключами «month» и «sum». Логично предположить, что данный список будет хранить суммы часов по месяцам плана. Приведенные на скриншоте циклы служат формированию массива строк `rows` будущего объекта целого плана. Извлекаются и сопоставляются каждому объекту `InCP` поля объектов `CPDetails`, рассчитываются все суммы, необходимые для отображения на плане. `Context['plan']` содержит в себе итоговый объект `plan`, являющийся словарём python с ключами и значениями `tp`, `months_sum`, `total_sum` и `rows`.

Был свёрстан шаблон веб-страницы `trainingplan.html`, содержащий таблицу значений `plan`, в шапке у которой базовая информация о плане. Интерфейс содержит два поля ввода – выпадающие списки для фильтрации по плану и виду обучения, а также – кнопки «фильтрация» и «сброс». Их работа обеспечивается благодаря небольшому коду JavaScript (рисунок 32). Определенные действия с полями ввода приводят к отправке GET-запросов. Представление для календарного плана построено аналогичным образом, за исключением отличия в некоторых полях моделей.

```

<script>
  const filteringPlan = document.querySelector("#filteringPlan")

  filterSearch.addEventListener("click", () => {
    window.location.search = `?tplan=${filteringPlan.value}`
  })

  filteringType.addEventListener("change", () => {
    window.location.search = `?ctype=${filteringType.value}`
  })

  document.querySelector("#resetAll").addEventListener("click", () => {
    window.location.search = ""
  })
</script>

```

Рисунок 32 – JS в шаблоне для плана

4.5.2 Потенциал интеллектуализации

Хотя была создана лишь минимальная версия модуля, уже возможно предложить решения по интеллектуализации планирования. Во-первых, самостоятельный интерфейс для динамического редактирования таблиц планов. Во-вторых, валидация введённых полей не только с помощью валидации моделей на backend, но и с помощью визуализации на frontend. Наконец, поддержка полуавтоматического планирования. Наиболее простой вариант – создание всех шаблонов, предварительно частично заполненных данными, с опорой на даты из регулирующих документов, а также реализация системы подсказок, соотносящих текущий календарный план с разрабатываемым учебным. Более сложное решение в данном направлении целесообразно лишь тогда, когда в системе накопится достаточное количество экземпляров планов. При соблюдении этого условия возможно встроить в приложение подобие рекомендательной системы, основанной на анализе и усреднении значений из старых планов для выдачи подсказок в заполнении нового. MVP модуля сможет ускорить процесс накопления документов в системе.

4.6 Модуль «Учёт часов»

Модулю соответствует приложение hours_app и разработан до MVP. В данной части системы осуществляется планирование и учёт часов преподавания сотрудников. Таблица TrainingHours позволяет спланировать, в какие дни и время преподаватели будут вести занятия (рисунок 33). Эта

информация сходит из планов, регламентируется нормами и в дальнейшем используется в модуле «Расчёты». Также учёт часов условно закрепляет преподавателей за группами.

The screenshot shows a web form titled 'Учёт часов' (Hour Accounting) with a sub-header 'Add Учёт часов'. The form contains several fields with red asterisks indicating required fields:

- Группа ***: A dropdown menu with 'Группа АЫ-666' selected.
- Преподаватель ***: A dropdown menu with 'Преподаватель sat, Андреев Андрей Андреевич, +79995553333' selected.
- Предмет ***: A dropdown menu with 'Предмет "Программирова...' selected.
- Тип времени ***: A dropdown menu with 'personal' selected.
- Дата ***: A date input field with '2021-05-31' and a 'Today' button with a calendar icon.
- Часы ***: A numeric input field with '1'.

Below the date field, there is a note: 'Note: You are 3 hours ahead of server time.' At the bottom of the form, there are two fields: 'Последний редактировал' (Last edited by) and 'Дата и время создания' (Creation date and time), both showing a dash '-'.

Рисунок 33 – Добавление записи в учёт часов

Добавление записи в TrainingHours: запись устанавливает соответствие между преподавателем, группой, предметом, типом времени (личное или рабочее), датой занятия, часами преподавания.

The screenshot shows the 'Учёт часов' (Hour Accounting) admin panel. At the top, there is a navigation bar with 'Home / Tech_School_App / Учёт часов'. Below it, there is a search bar with the text 'Select Учёт часов to change'. The search bar contains several dropdown menus for 'Группа', 'Преподаватель', 'Предмет', 'Тип времени', and 'Дата', followed by a 'Search' button and a result count '3 results (7 total)'. Below the search bar, there is a 'Go' button and a text '0 of 3 selected'. To the right of the search bar, there is a green button with a plus icon and the text 'Add Учёт часов'.

<input type="checkbox"/>	Группа	Преподаватель	Предмет	Тип времени	Дата	Часы	Дата и время создания	Последний редактировал
<input type="checkbox"/>	Группа НУ-111	Преподаватель rabbit, Иванов Иван Иванович, +79191919191	Предмет "Механика"	working	Nov. 10, 2020	6	May 17, 2021, 9:24 a.m.	rita
<input type="checkbox"/>	Группа АЫ-666	Преподаватель rabbit, Иванов Иван Иванович, +79191919191	Предмет "Механика"	working	May 17, 2021	8	May 17, 2021, 9:21 a.m.	-
<input type="checkbox"/>	Группа ХЫ-123	Преподаватель rabbit, Иванов Иван Иванович, +79191919191	Предмет "Механика"	personal	May 14, 2021	2	May 17, 2021, 9:21 a.m.	-

At the bottom left of the table, there is a text '3 Учёт часов'.

Рисунок 34 – Просмотр и фильтрация записей в админ-панели

Редактирование данной таблицы подчиняется нормам, хранящимся в таблицах WorkingDates (календарь рабочих дат), HoursNorm (нормы часов). Также учёт часов соотносится с планированием, но данный функционал пока не был реализован. В таблице HoursNorm указаны лимиты преподавания в

личное и рабочее время, в разные периоды (день, месяц, год). WorkingDates хранит эквиваленты рабочих и выходных дней в часах.

На рисунке 35 изображён метод модели TrainingHours, используемый для валидации на соответствие WorkingDates. Производится сравнение суммы добавляемых часов в день (найдена с помощью aggregate) с число часов для этого дня в календаре дат.

```
def check_working_dates(self):
    _date = self.date
    _teacher = self.teacher
    norm_hours = getattr(WorkingDates.objects.filter(date=_date)[0], 'in_hours')
    th = TrainingHours.objects.filter(teacher=_teacher, date=_date)
    hours_th = th.aggregate(Sum('hours'))["hours__sum"]
    plus_hours = self.hours
    if not self.id:
        if hours_th + plus_hours <= norm_hours:
            return True
        else:
            return False
    if self.id:
        minus_hours = getattr(TrainingHours.objects.get(pk=self.id), 'hours')
        if hours_th + plus_hours - minus_hours <= norm_hours:
            return True
        else:
            return False
```

Рисунок 35 – Валидатор для учёта часов

Для получения сводных таблиц, выступающих аналогами оригинальных документов, которые были проанализированы перед началом разработки, как и в предыдущих разделах, использовались Django views (рисунок 36, 37).

Учёт часов (год)

Дата (будет использован год):

ДД . ММ . ГГГГ

Фильтровать Сброс

Учёт часов преподавания

преподаватель	итого за год	в личное время	в рабочее время
Преподаватель rita, Суздальцева Маргарита Вячеславовна, 89119411200	23	3	20
Преподаватель rabbit, Иванов Иван Иванович, +79191919191	16	2	14

Рисунок 36 – Интерфейс для просмотра учёта часов за год

Учёт часов (группа, месяц)

Отобразить группу:

Дата (месяц и год):

ДД . ММ . ГГГГ

[Фильтровать](#) [Сброс](#)

Учёт часов преподавания

преподаватель	детали	личные часы	рабочие часы	всего
Преподаватель rita, Суздальцева Маргарита Вячеславовна, 801103730006	<ul style="list-style-type: none"> Предмет "Программирование" Группа АЫ-666 May 16, 2021 8 ч. (working) Предмет "Математика" Группа АЫ-666 May 17, 2021 3 ч. (personal) Предмет "Математика" Группа АЫ-666 May 26, 2021 7 ч. (working) 	3	15	18
Преподаватель rabbit, Иванов Иван Иванович, +79191919191	<ul style="list-style-type: none"> Предмет "Механика" Группа АЫ-666 May 17, 2021 8 ч. (working) 	0	8	8
ИТОГО		3	23	26

Рисунок 37 – Интерфейс для просмотра учёта часов по группе за месяц

Рассмотрим представление данных из таблицы TrainingHours с фильтрацией по группе и месяцу (рисунок 38).

```

teachers_group = {}

for row in queryset:
    if not teachers_group.get(row.teacher.pk):
        teachers_group[row.teacher.pk] = {}
        teachers_group[row.teacher.pk]["data"] = []
        teachers_group[row.teacher.pk]["teacher"] = row.teacher

    teachers_group[row.teacher.pk]["data"].append(row)

for key in teachers_group.keys():
    row = teachers_group[key]

    teachers_group[key]["sum_working"] = 0
    teachers_group[key]["sum_personal"] = 0
    teachers_group[key]["sum_total"] = 0

    for _row in row["data"]:
        teachers_group[key]["sum_total"] += _row.hours
        if _row.time_type == 'w':
            teachers_group[key]["sum_working"] += _row.hours
        elif _row.time_type == 'p':
            teachers_group[key]["sum_personal"] += _row.hours

teachers_groups = []

for key in teachers_group.keys():
    teachers_groups.append(teachers_group[key])

total_p = queryset.filter(time_type="p").aggregate(Sum('hours'))["hours__sum"]
total_w = queryset.filter(time_type="w").aggregate(Sum('hours'))["hours__sum"]
total = queryset.aggregate(Sum('hours'))["hours__sum"]
totals = {'total_p': total_p, "total_w": total_w, "total": total}

context["training_hours"] = teachers_groups
context["totals"] = totals

```

Рисунок 38 – Часть кода описываемого view

На основе параметров GET-запросов (data, group) был отфильтрован queryset из всех объектов TrainingHours, то есть были получены только записи выбранного месяца и группы. На скриншоте отображено следующее действие – получение объекта для вывода в html-шаблоне. Сначала создаётся словарь,

где каждый преподаватель (точнее, id Teacher) используется как ключ. В качестве значений хранятся словари с ключами «data» (объект из queryset) и «teacher». Затем в цикле из всех строк для каждого преподавателя рассчитываются суммы личных и рабочих часов, а также общая сумма. Дополнительно с помощью фильтрации и aggregate найдены суммы для всей выборки.

4.7 Версионность планирования и учёта часов

Планирование и учёт часов – модули, требующие хранения версий. Для решения задачи были использованы сериализаторы соответствующих моделей в JSON, дополнительные модели для хранения версий и Lifecycle hooks. Отслеживанию подлежат модели TP, InTP, TPDDetails, то есть все составные части учебного плана. Для учёта часов отслеживается непосредственно таблица TrainingHours. Рассмотрим создание версий на примере модели учёта часов.

На рисунке 39 видно, что к модели добавлены «хуки» @hook(AFTER_CREATE, where='created_at', has_changed=True) и @hook(AFTER_UPDATE, when_any=['group', 'teacher', 'subject', 'time_type', 'date', 'hours'], has_changed=True).

```
@hook(AFTER_CREATE, when='created_at', has_changed=True)
@hook(AFTER_UPDATE, when_any=['group', 'teacher', 'subject', 'time_type', 'date', 'hours'], has_changed=True)
def traininghours_versions(self):
    th = TrainingHours.objects.get(pk=self.pk)
    serializer = TrainingHoursSerializer(th)
    changes = DraftTH(
        content=serializer.data,
        who_changed=self.who_changed_last,
    )
    changes.save()
```

Рисунок 39 – «хуки» для создания версий

Первый срабатывает после создания объекта модели, когда было изменено поле «created_at». Второй срабатывает после обновления одного из значащих полей модели. Действия, которые должны быть совершены при срабатывании одного из них, описаны в def traininghours_versions(self). В первых, функция получает объект плана, который был только что добавлен

либо отредактирован. Во-вторых, инициирует соответствующий сериализатор, здесь – TrainingHoursSerializer. Это сериализатор класса serializers.ModelSerializer, он сериализует все поля модели, то есть представляет экземпляры TrainingHours в формате JSON. Такая версия сериализатора (рисунок 40) сохраняет только id, если поля имеют значения FK.

```
class TrainingHoursSerializer(serializers.ModelSerializer):
    class Meta:
        model = TrainingHours
        fields = '__all__'
```

Рисунок 40 – Используемый сериализатор

Затем, происходит создание новой записи в таблице DraftTP. В поле content заносится измененный объект в JSON, в who_changed – автор изменения, а поле created_at заполняется автоматически, так как в классе модели для него указано auto_now_add=True. Записи таблицы с версиями доступны для просмотра в админ-панели (рисунок 41).

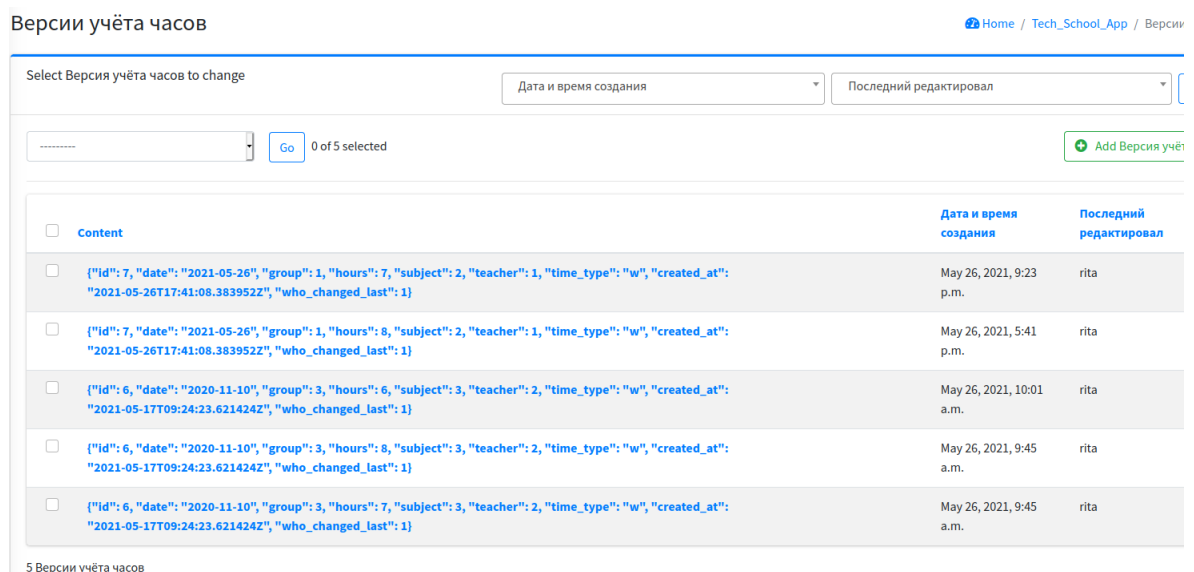


Рисунок 41 – Хранение версий

Версии учебных планов создаются по абсолютно тому же принципу. При такой системе хранения версий в будущем могут быть реализованы откаты между версиями. Для совершения перехода к более старой версии плана необходимо будет предусмотреть код, создающий новую версию объекта, заполненную значениями старой, взятыми из таблицы Draft.

4.8 Модуль «Электронный журнал»

Модулю соответствует приложение `ejournal_app`. Разработанные в ходе написания выпускной квалификационной работы функции покрывают необходимый функционал, определенный ранее. Модуль электронный журнал предназначен для использования как администратором учебного процесса, так и преподавателями. Для них существуют разные сценарии использования: планирование и анализ либо ведение журнала.

4.8.1 Формирование расписания

Администратор учебного процесса формирует расписание занятий, исходя из данных таблицы «учёт часов». Занятия добавляются через админ-панель (рисунок 42).

Рисунок 42 – Добавление занятия

Для модели `Class` предусмотрен валидатор (рисунок 43), который позволяет добавлять занятия только с теми характеристиками, которые были закреплены в `TrainingHours` (дата, группа, преподаватель и предмет).

```
def class_planned(self):
    _date = self.when.date()
    _group = self.group
    _teacher = self.teacher
    _subject = self.subject
    th = TrainingHours.objects.filter(group=_group, teacher=_teacher, subject=_subject, date=_date)
    if th: return True
    else: return False
```

Рисунок 43 – Валидатор для `Class`

Функция, осуществляющая поиск той записи в `TrainingHours`, на основании которой добавляется занятие в `Class`. Также часы занятий,

добавленных в Class, должны соответствовать плановым. На рисунке 44 представлена функция, которая используется для вызова ValidationError: нельзя добавить те часы, которые превышают лимит.

```
def hours_match_limit(self):
    _date = self.when.date()
    _group = self.group
    _teacher = self.teacher
    _subject = self.subject
    th = TrainingHours.objects.filter(group=_group, teacher=_teacher, subject=_subject, date=_date)
    classes = Class.objects.filter(when=self.when)
    hours_th = th.aggregate(Sum('hours'))["hours__sum"]
    hours_classes = classes.aggregate(Sum('hours'))["hours__sum"]
    plus_hours = self.hours
    if not self.id:
        if hours_classes + plus_hours <= hours_th:
            return True
        else:
            return False
    if self.id:
        minus_hours = getattr(Class.objects.get(pk=self.id), 'hours')
        if hours_classes + plus_hours - minus_hours <= hours_th:
            return True
        else:
            return False
```

Рисунок 44 – Валидатор, обеспечивающий соответствие лимиту часов

4.8.2 Добавление оценок и просмотр информации занятий

Для того, чтобы собрать данные о посещаемости и оценках учащихся, преподавателям необходимо вносить отметки в систему. Запись в таблицу Grade может быть добавлена через интерфейс админ-панели (рисунок 45), разделённый на несколько вкладок (с помощью fieldsets при настройке класса ModelAdmin). Такое решение визуально отделяет два возможных варианта отметки, так что пользователь может заполнить основную информацию и затем – только одну из вкладок. К любой оценке автоматически добавляется отметка об авторстве. Также у модели существуют два метода, используемые для валидации: def student_class_match(self) и def attendance_exist(self). Первый отвечает за возможность поставить оценку за занятие только студенту той группы, для которой оно проводилось. Второй не допускает добавления дублирующих отметок о посещении одного занятия одним студентом.

Оценки

Add Оценка

main grade attendance

Занятие *
Группа АЫ-666 | Преподаватель gabbit, Иванов Иван Иванович, +79191919191 | Предмет "Механика" | 2021-05-17 15:00:00+00:00

Студент *
Студент hhhhhh, Олегов Олег Олегович, +7666666666666

Оценка / Посещение *
grade

Автор оценки
-

Рисунок 45 – Добавление оценки

Аналогом расписания занятий является расширяющий админ-панель `ClassesViews(TemplateView)` для модели `Class` (рисунок 46). Во view сначала фильтруется `queryset` из всех занятий, а затем в цикле из них формируются словари значений для заполнения карточек. Данное представление содержит карточки всех занятий с возможностью фильтрации по группе, преподавателю, типу занятия и дате. Каждая карточка имеет кнопку, раскрывающую список студентов, участвующих в занятии, с информацией о том, были ли им проставлены отметки.

Преподаватель:

Тип занятия:

Дата:

Фигурный сброс

Экзамен

Группа: Группа НУ-111

Преподаватель: Преподаватель gabbit, Иванов Иван Иванович, +79191919191

Предмет: Предмет "Механика"

Когда: 2021-05-22 10:57:36

Длительность: 1 ч.

Студенты

Студент	Посещаемость	Оценки
Студент hhhhhh, Олегов Олег Олегович, +7666666666666	present	[5]

Урок

Группа: Группа АЫ-666

Преподаватель: Преподаватель rita, Суздальцева Маргарита Вячеславовна, 89119771298

Предмет: Предмет "Механика"

Когда: 2021-05-19 10:44:10

Длительность: 1 ч.

Студенты

Студент	Посещаемость	Оценки
Студент 191891, Студентов Студент Студентович, 000000000000000	present	[4, 4]
Студент hfkgf, Егоров Егор Егорович, +701001101010	absent	Нет оценок.

Урок

Рисунок 46 – Интерфейс «Карточки занятий»

Одно из требований к системе – возможность использовать электронный журнал с мобильных устройств. Интерфейсы админ-панели при подключенной библиотеке jazzmin сами по себе являются адаптивными (способны подстраиваться под различные технические условия, размеры экранов). Однако, интерфейс с карточками было незначительно доработан в самом template для достижения адаптивности (рисунок 47).

Группа:

Преподаватель:

Дата (месяц и год):

ДД . ММ . ГГГГ

Фильтровать Сброс

Экзамен

Группа
Группа НУ-111

Преподаватель
Преподаватель rabbit, Иванов Иван Иванович, +79191919191

Предмет
Предмет "Механика"

Когда
May 22, 2021, 10:57 a.m.

Длительность

Рисунок 47 – Мобильный вариант view с карточками занятий

4.8.3 Получение средних показателей студентов

Каждый месяц администратору требуется получать доступ к статистике электронного журнала: средним баллам студентов по предметам и проценту посещаемости каждого из них. Как уже было сказано в предыдущих разделах, эти параметры применяются в расчёте стипендий. Этой цели служит представление EjournalView(TemplateView) (рисунок 48). Некоторые из операций, выполняемых в рамках данного view: записи таблицы Grade были различными способами отфильтрованы, агрегированы, аннотированы. Далее

значения были сгруппированы по студентам. Сам шаблон имеет внутри три карточки:

- 1) оценки сгруппированы по студентам и предметам,
- 2) посещения сгруппированы по студентам и предметам,
- 3) итоговые средние значения сгруппированы по студентам.

```
# информация отображается только если выбрана группа и дата
if group and date:
    year, month, _ = date.split('-') # год и месяц
    queryset = queryset.filter(class_id__when__year=year, class_id__when__month=month, class_id__group_id=group)

    classes = Class.objects.filter(group_id=group, when__year=year, when__month=month) # занятия группы
    total_hours = classes.aggregate(total=Sum('hours'))["total"] # сколько всего часов занятий у группы в указанный период

    grades_queryset = queryset.filter(grade_type="g") # только оценки
    attendance_queryset = queryset.filter(grade_type="a") # только посещения

    # средние баллы по каждому предмету для каждого студента (складываются из всех оценок за период)
    avg_grades = grades_queryset.values('class_id__subject', 'student').order_by().annotate(Avg('grade'))

    # % посещаемости по каждому предмету для каждого студента без учёта длительности занятий, только соотношение отметок "present" и "absent"
    present_count = Count('grade', filter=Q(attendance='1'))
    absent_count = Count('grade', filter=Q(attendance='0'))
    avg_attendance = attendance_queryset.values('class_id__subject', 'student').order_by().annotate(present=present_count).annotate(absent=absent_count)
```

Рисунок 48 – Часть кода view

Внешний вид интерфейса изображен на рисунке 49.

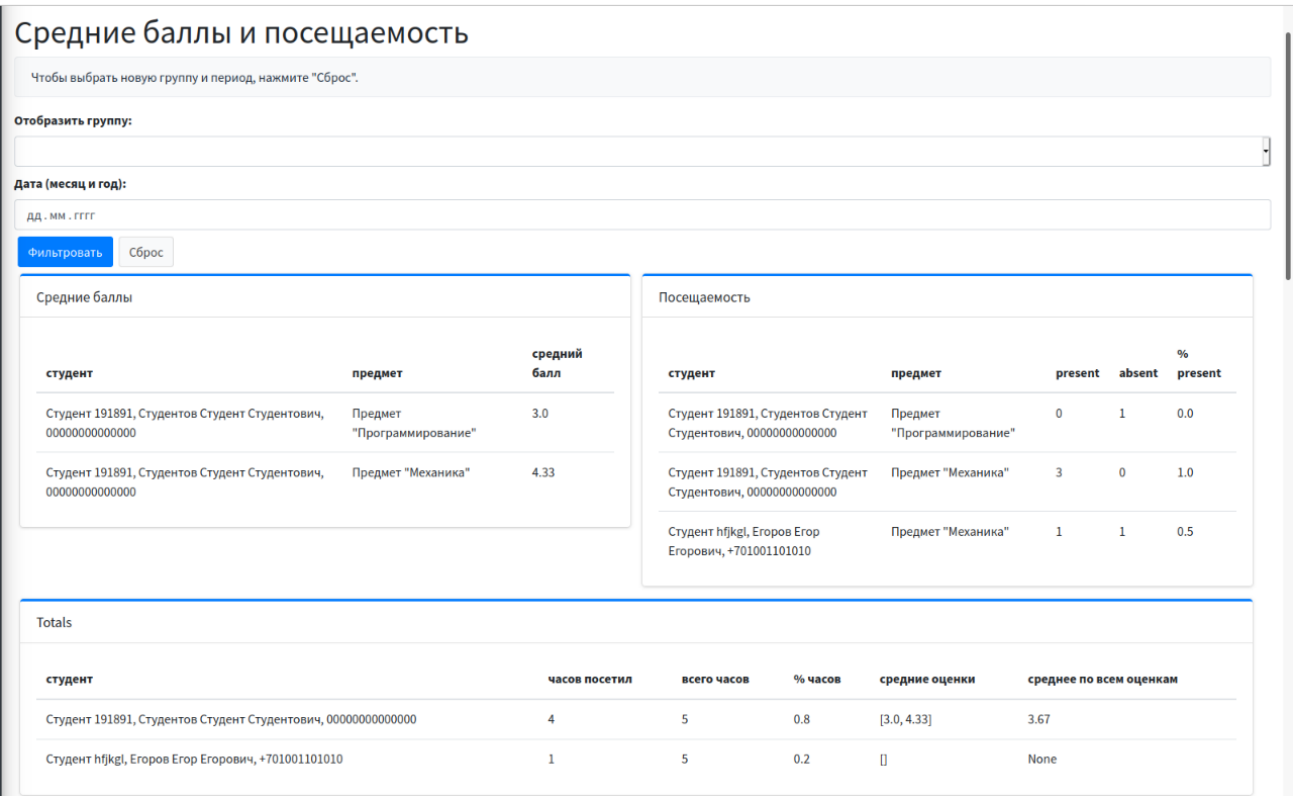
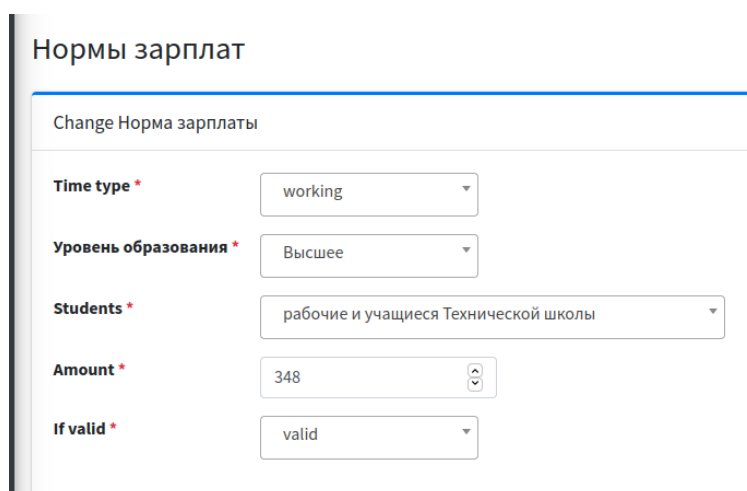


Рисунок 49 – Интерфейс «Средние баллы и посещаемость»

4.9 Модуль «Расчёты»

Модулю соответствует приложение `accounting_app`. Эта часть системы также существует в виде MVP, так как еще не все возможные операции были автоматизированы. Данный модуль обрабатывает значения таблиц, которые уже должны быть заполнены данными к соответствующему этапу. Для проведения расчётов в таблицах `SalaryNorm` и `ScholarshipNorm` должны содержаться валидные нормы (рисунок 50), основанные на документах (положениях об оплате и стипендии). Интерфейс для расчёта зарплат представлен далее (рисунок 51, 52).



Нормы зарплат

Change Норма зарплат

Time type * working

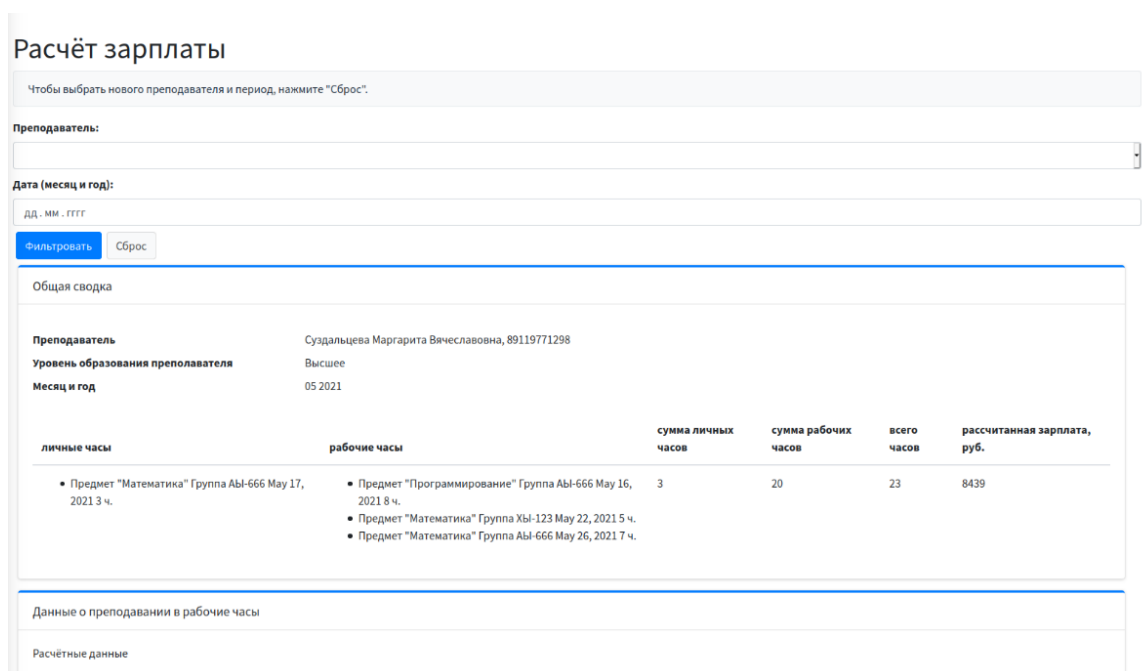
Уровень образования * Высшее

Students * рабочие и учащиеся Технической школы

Amount * 348

If valid * valid

Рисунок 50 – Заполнение нормы зарплаты через форму админ-панели



Расчёт зарплаты

Чтобы выбрать нового преподавателя и период, нажмите "Сброс".

Преподаватель:

Дата (месяц и год):

ДД - мм - гggg

Фильтровать Сброс

Общая сводка

Преподаватель	Суздальцева Маргарита Вячеславовна, 89119771298
Уровень образования преподавателя	Высшее
Месяц и год	05 2021

личные часы	рабочие часы	сумма личных часов	сумма рабочих часов	всего часов	рассчитанная зарплата, руб.
• Предмет "Математика" Группа АЫ-666 May 17, 2021 3 ч.	• Предмет "Программирование" Группа АЫ-666 May 16, 2021 8 ч. • Предмет "Математика" Группа ХЫ-123 May 22, 2021 5 ч. • Предмет "Математика" Группа АЫ-666 May 26, 2021 7 ч.	3	20	23	8439

Данные о преподавании в рабочие часы

Расчётные данные

Рисунок 51 – Интерфейс «Расчёт зарплаты» часть 1

Для получения расчёта необходимо выбрать преподавателя и месяц. Страница совмещает в себе три карточки: «общая сводка», «данные о преподавании в рабочие часы», «данные о преподавании в личные часы». Для каждой строки в двух последних карточках рассчитывается свой подытог, зависящий от остальных параметров, отражённых в таблице (уровень студентов группы, тип часов преподавания, число часов). Учитывается также уровень образования преподавателя. Для каждого набора параметров подбирается и отображается в отдельном столбце подходящая действующая норма из таблицы SalaryNorm. Полная рассчитанная зарплата (сумма подытогов) приведена в первой карточке.

группа	уровень студентов	часов	подходящая норма		руб.
Группа АЫ-666	рабочие и учащиеся Технической школы	15	SalaryNorm object (4)		5220
			Тип часов	working	
			Уровень образования	Высшее	
			Контингент	рабочие и учащиеся Технической школы	
			Тариф	348	
Группа ХЫ-123	руководители среднего звена, специалисты, служащие	5	SalaryNorm object (6)		2175
			Тип часов	working	
			Уровень образования	Высшее	
			Контингент	руководители среднего звена, специалисты, служащие	
			Тариф	435	
Итого за рабочие часы, руб.					7395

Данные о преподавании в личные часы					
Расчётные данные					
группа	уровень студентов	сумма часов	подходящая норма		Итого по строке, руб.
Группа АЫ-666	рабочие и учащиеся Технической школы	3	SalaryNorm object (4)		1044
			Тип часов	working	
			Уровень образования	Высшее	
			Контингент	рабочие и учащиеся Технической школы	
			Тариф	348	
Итого за рабочие часы, руб.					1044

Рисунок 52 – Интерфейс «Расчёт зарплаты» часть 2

Внутри класса SalaryCountView(TemplateView) выполняются следующие действия:

- 1) фильтрация queryset из объектов TrainingHours с использованием параметров GET-запроса,
- 2) фильтрация по двум типам часов (деление на две выборки th_working и th_personal) и суммирование hours с помощью aggregate,
- 3) аннотирование th_working, th_hours (также сумма часов, но с группировкой по значениям группы и уровня образования группы),

4) два цикла с получением объектов из строк для карточек, подбор нормы для каждой строки с помощью filter по значимым полям), расчёт подытогов,

5) расчёт полного итога.

Условия для получения разных уровней стипендии не ограничиваются средним баллом (рисунок 53). Например, вариант может зависеть от получения неудовлетворительной оценки за конкретный экзамен, а не напрямую от среднего балла, который рассчитывается в модуле «Электронный журнал». По этой причине интерфейс для расчёта стипендии должен работать в полуавтоматическом режиме.

Нормы стипендий

Change Norma stipendii

Нижняя граница балла 3.0

Верхняя граница балла 4.0

Количество руб./мес. * 13000

Описание из документа от 3 до 4

If valid * valid

Программа * Программа "Программа"

Период обучения * theory

Рисунок 53 – Заполнение нормы стипендии через админ-панель

Предусмотрено поле description для хранения описания из исходного документа. Расчёт стипендии в данном модуле производится с помощью таблицы (рисунок 54, 55). При выборе студента и месяца таблица заполняется той информацией о нем, которая необходима для расчётов: оценки за экзамены, средние баллы по предметам, общий средний балл, % посещаемости. Важную роль играет программа той группы, в которой состоит студент. В столбце таблицы «нормы программы» находится выпадающий список описаний действующих норм стипендии для этой программы из таблицы ScholarshipNorm. Пока проверка всех условий для каждого тарифа не

реализована, администратор процесса может самостоятельно оценить средние баллы студента и вручную выбрать подходящую норму по описанию. Когда норма выбрана, в столбце «Стипендия, руб.» отображается рассчитанная сумма.

Принцип расчёта средних показателей описан в разделе, посвященном модулю «Электронный журнал». Принципиально новыми нюансами в ScholarshipCountView являются только получение группы и программы студента, фильтрация норм (SalaryNorm) по программе, использование формулы расчёта стипендии.

Расчёт стипендии

Студент:

Дата (месяц и год):

ДД . ММ . ГГГГ

Фильтровать

Сброс

Стипендия студента

Студент

Группа

Программа

Студент 191891, Студентов Студент Студентович, 0000000000000000
Группа АЫ-666
Программа "Программа"

Оценки за экзамены	Ср.баллы	Общий средний балл	% посещаемости	Нормы программы	Стипендия, руб.
<ul style="list-style-type: none">Предмет "Программирование": 3.0Предмет "Механика": 4.33		3.6666666666666665	0.8 (4 / 5 ч.)	<div>▼</div> <div>есть неуд по экзамену от 4 и выше до 3.0 и когда есть оценка ниже 3.0 от 3 до 4</div>	10400.0

Рисунок 54 – Интерфейс «Расчёт стипендии» часть 1

Студент:

Дата (месяц и год):

ДД . ММ . ГГГГ

Фильтровать

Сброс

Стипендия студента

Студент

Группа

Программа

Студент hhhhhh, Олег Олг Олегович, +7666666666666666
Группа НУ-111
Программа "Программа 2"

Оценки за экзамены	Ср.баллы	Общий средний балл	% посещаемости	Нормы программы	Стипендия, руб.
<ul style="list-style-type: none">Предмет "Механика": 5	<ul style="list-style-type: none">Предмет "Механика": 5.0	5.0	1.0 (1 / 1 ч.)	<div>▼</div> <div>норма для всех</div>	20000.0

Рисунок 55 – Интерфейс «Расчёт стипендии» часть 2

Выводы по четвертой главе

В главе приведены основные технологии, выбранные для разработки ИИС, обоснованы принятые решения. Далее была рассмотрена разработка модулей веб-приложения: представлены изображения интерфейсов, кратко описана реализованная логика. Программная логика основывалась на информации раздела «проектирование системы». Также была предусмотрена версия «планирования», «учёта часов» и «БД преподавателей». Обеспечена минимальная необходимая адаптивность интерфейсов.

Результатом выполнения практической части выпускной квалификационной работы является пилотная версия ИИС, покрывающая базовый функционал, заявленный в требованиях к системе. Иначе говоря, веб-приложение готово к развёртыванию в Технической школе в тестовом режиме. Также проделанная работа может рассматриваться как бэкграунд для дальнейшего развития системы.

5 БЕЗОПАСНОСТЬ И МАСШТАБИРОВАНИЕ СИСТЕМЫ

5.1 Обеспечение безопасности системы

5.1.1 Информационные меры

Для всех сотрудников, которые имеют доступ в систему, необходимо провести краткий инструктаж о том, как корректно её использовать. Немаловажными являются регулярные смены паролей пользователей и проведение аудита безопасности системы специалистами. В организации должен присутствовать системный администратор, регулярно выполняющий бэкапы на уровне СУБД.

5.1.2 Разграничение доступа

Админ-панель Django позволяет гибко настраивать права пользователей. Когда веб-приложение будет развёрнуто, планируется создать в БД группу пользователей «Преподаватели» и назначить ей соответствующий перечень прав. Преподаватели должны иметь доступ к редактированию и просмотру тех таблиц и представлений, который позволят им вносить оценки и посещаемость обучающихся в Технической школе студентов. При этом администратор (или группа лиц, выполняющая его функции) должен обладать всеми предусмотренными правами. Также доступ в систему может быть ограничен путём ограничения списка IP-адресов, с которых можно подключиться к серверу, адресами подключения из сети организации.

5.1.3 Хранение данных

В качестве хранилища данных для системы, в которой хранятся в том числе персональные данные граждан РФ, необходимо выбирать СУБД, сертифицированную ФСТЭК (Федеральная служба по техническому и экспортному контролю). Сертификация гарантирует соответствие требованиям по защите персональных данных. Также рационально ограничить подключение к БД одним хостом (тем же, где запущен бэкенд).

5.1.4 Выбор технологий разработки

Сам по себе выбор стека технологий может выступать в качестве преимущества или уязвимости в вопросе информационной безопасности веб-

приложений. Django (основное средство разработки рассматриваемого проекта) считается достаточно безопасным фреймворком. В документации проекта есть обзор функций, позволяющих обеспечить оптимальный уровень безопасности и советы по их применению [10].

5.1.5 Условия развёртывания системы

На безопасность информационной системы также влияют характеристики сервера. Оптимальным решением является приобретать уже защищённые сервера. Операционные системы, как на виртуальных, так и на физических серверах, тоже могут быть сертифицированы соответствующими организациями либо нет. В качестве примера сертифицированного ФСТЭК российского дистрибутива Linux можно привести ALT Linux [11]. Многие версии операционной системы Windows сертифицированы ФСТЭК и также могут быть использованы для заявленных целей. При условии, что развёртывание выполняется не на локальном сервере и в России, оптимально приобрести домен .RU/.РФ и арендовать VDS у юридически проверенного хостинг-провайдера, находящегося на территории России, а также имеющего дата-центры в РФ.

5.2 Возможность дальнейшего масштабирования системы

5.2.1 Интеграция модулей системы

Хотя все модули информационной системы описаны как тесно связанные между собой, из раздела, посвященного разработке, очевидно, что может быть проделано еще много работы по их интеграции. Диаграмма компонентов UML наглядно показывает, как выглядит их взаимодействие в идеальном случае. Следующим шагом в разработке ИИС Технической школы должна стать реализация логики «общения» модулей друг с другом с использованием API. Должны быть созданы такие эндпоинты, чтобы один модуль мог получать от другого данные (параметры) сразу в том виде, в котором они необходимы для обработки данных (расчётов, планирования). Выполнение расчётов «с нуля» хотя и даёт корректные результаты, не является оптимальным.

5.2.2 Повторное использование данных для анализа

Потенциально полезным вариантом расширения функционала системы является создание инструментов для создания датасетов на основе таблиц БД приложения. На ранних этапах существования системы, когда она содержит все еще не так много информации, сложно оценить пользу от такого программного решения. Однако, при длительной эксплуатации возможность выгрузить накопленные в системе структурированные данные позволит быстро перейти к их анализу. Как уже говорилось в тексте работы ранее, такой анализ позволит оптимизировать бизнес-процессы организации. Например, выявление закономерностей в планах позволит системе давать пользователю рекомендации по их составлению.

Выводы по пятой главе

В главе были описаны базовые правила и условия для обеспечения адекватного уровня безопасности веб-приложения на данном этапе: информационные меры, выбор средств для разработки и развертывания, важность наличия сертификатов у ПО. Упомянута необходимость привлечения специалистов по информационной безопасности к работе над системой в будущем. Дополнительно были приведены некоторые перспективные варианты масштабирования ИИС Технической школы.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было проведено исследование предметной области: проанализирована специфика образовательного подразделения, выявлены и визуализированы информационные потоки Технической школы, описан и декомпозирован основной бизнес-процесс – администрирование учебного процесса.

В ходе работы над заданием выявлены и описаны требования к будущей системе, определён список необходимых модулей, а также обоснована необходимость создания собственной информационной системы. Были спроектированы структуры для хранения данных, выбраны подходящие актуальные технологии для реализации ИИС в качестве веб-приложения. С помощью фреймворка Django языка программирования Python и других вспомогательных библиотек разработана пилотная версия многомодульного веб-приложения с элементами интеллектуализации.

Также приведены базовые требования к безопасности системы, сформулированы варианты её масштабирования в будущем. Реализованная ИИС создаёт в образовательном подразделении «Техническая школа» ГУП «Петербургский метрополитен» условия для накопления данных, которые позволят в будущем отслеживать динамику информационных потоков с целью оптимизировать все процессы, касающиеся работы с информацией.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Django documentation. [Электронный ресурс]. – 2021. – URL: <https://docs.djangoproject.com/en/3.1/> (дата обращения: 20.05.2021).
2. Django REST framework. [Электронный ресурс]. – 2021. – URL: <https://www.django-rest-framework.org/> (дата обращения: 20.05.2021).
3. PostgreSQL: Documentation. [Электронный ресурс]. – 2021. – URL: <https://www.postgresql.org/docs/> (дата обращения: 20.05.2021).
4. Сайт ресурсов UML. [Электронный ресурс]. – 2021. – URL: <https://www.uml.org/> (дата обращения: 20.05.2021).
5. Моделирование на UML. Интернет-книга "Моделирование на UML" [Электронный ресурс]. – 2013. – URL: <http://book.uml3.ru/> (дата обращения: 02.06.2021).
6. Методология функционального моделирования IDEF0: руководящий документ. М.: ИПК Изд-во стандартов, 2000. URL: <https://nsu.ru/smk/files/idef.pdf> (дата обращения: 27.05.2021).
7. Django Lifecycle Hooks. [Электронный ресурс]. – 2021. – URL: <https://rsinger86.github.io/django-lifecycle/> (дата обращения: 20.05.2021).
8. Visual Paradigm Online. What is Information Flow Diagram? [Электронный ресурс]. – 2021. – URL: <https://online.visual-paradigm.com/knowledge/business-design/what-is-information-flow-diagram/> (дата обращения: 02.06.2021).
9. Техническая школа. Официальный сайт ГУП «Петербургский метрополитен». [Электронный ресурс]. – 2021. – URL: <http://www.metro.spb.ru/techscool.html> (дата обращения: 31.05.2021).
10. Security in Django. Django Documentation. [Электронный ресурс]. – 2021. – URL: <https://docs.djangoproject.com/en/3.2/topics/security/> (дата обращения: 20.05.2021).
11. ALT Linux Wiki. [Электронный ресурс]. – 2021. – URL: <https://www.altlinux.org/%D0%93%D0%BB%D0%B0%D0%B2%D0%BD%D0>

[%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0](#) (дата обращения: 31.05.2021).

12. Moodle – Open-source learning platform. Moodle.org. [Электронный ресурс]. – 2021. – URL: <https://moodle.org/> (дата обращения: 28.05.2021).

13. 17 Best Learning Management Systems (LMS) of 2021 Ranked. Adam Enfroy. [Электронный ресурс]. – 2021. – URL: <https://www.adamenfroy.com/learning-management-system> (дата обращения: 28.05.2021).

14. ESS SIMS. Education Software Solutions. [Электронный ресурс]. – 2021. – URL: <https://www.ess-sims.co.uk/> (дата обращения: 28.05.2021).

15. Язык программирования Python. Python.org. [Электронный ресурс]. – 2021. – URL: <https://www.python.org/> (дата обращения: 20.05.2021).

16. Introduction – Bootstrap v4.6. [Электронный ресурс]. – 2020. – URL: <https://getbootstrap.com/docs/4.6/getting-started/introduction/> (дата обращения: 20.05.2021).

17. Django-jazzmin (Jazzy Admin) – PyPI. [Электронный ресурс]. – 2021. – URL: <https://pypi.org/project/django-jazzmin/> (дата обращения 20.05.2021).

18. Visual Paradigm Tutorials. [Электронный ресурс]. – 2020. – URL: <https://www.visual-paradigm.com/tutorials/> (дата обращения: 27.05.2021).

ПРИЛОЖЕНИЕ А

Диаграмма IDEF0

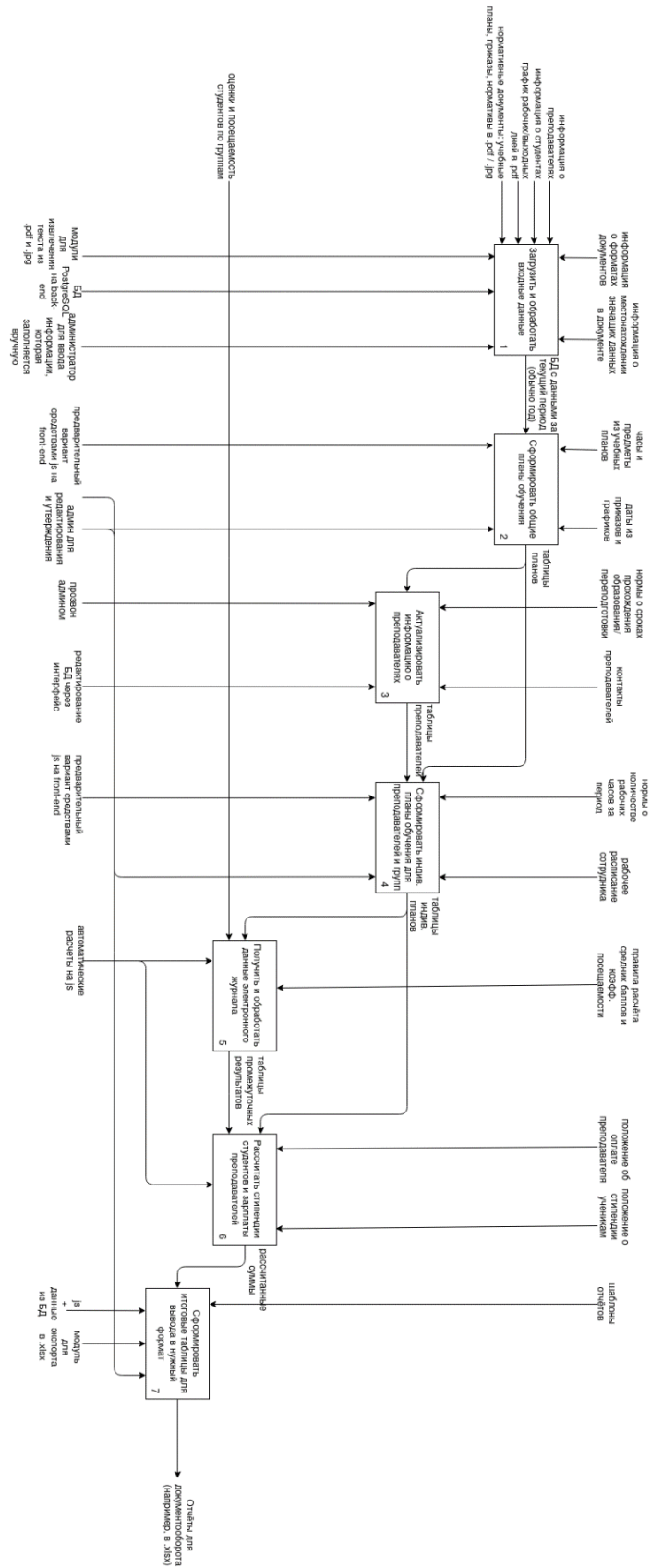


Рисунок 56 IDEF0

ПРИЛОЖЕНИЕ Б

Схема базы данных приложения

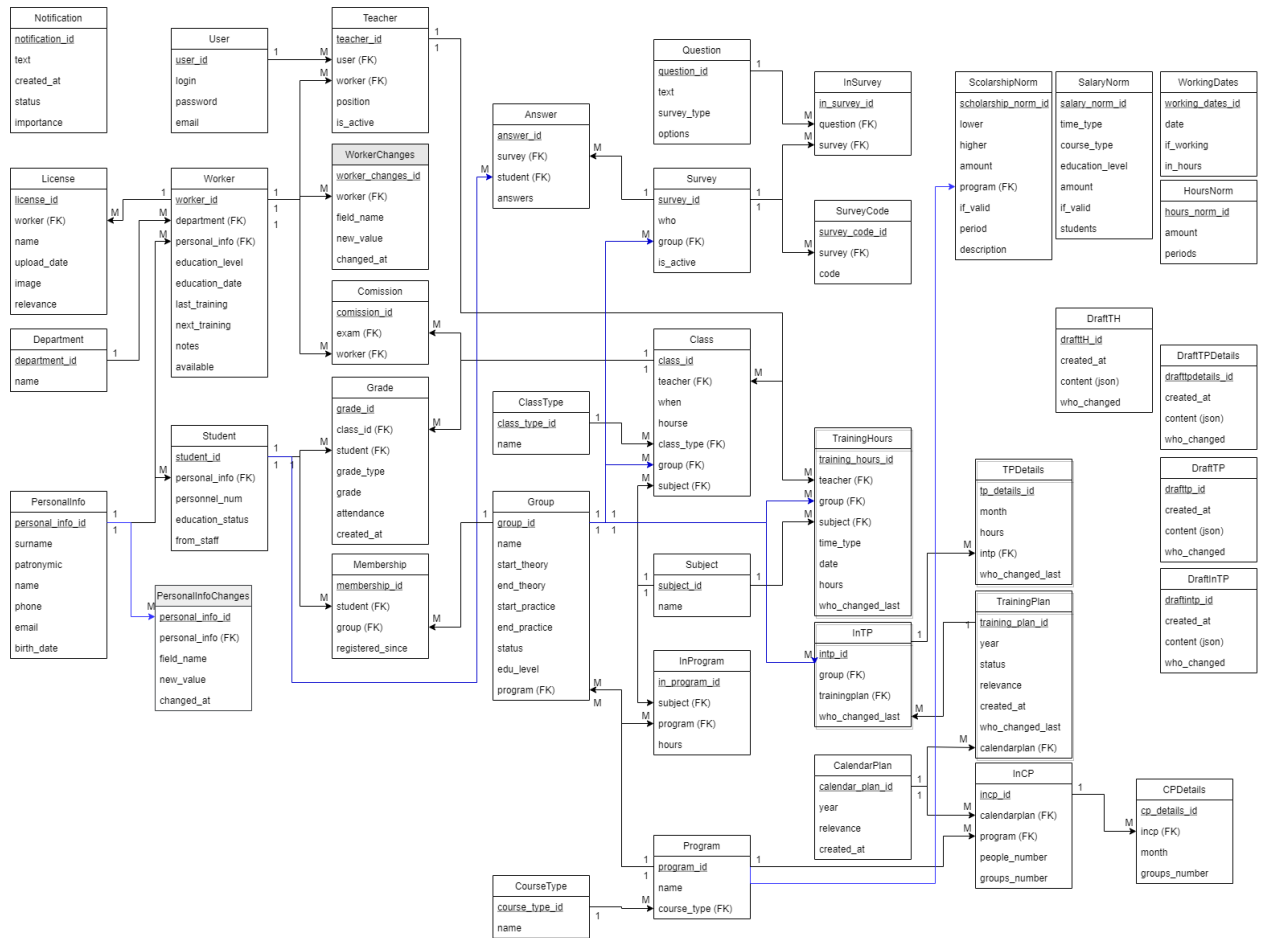


Рисунок 57 Схема БД

ПРИЛОЖЕНИЕ В

Код notification_script.py

```
import requests
import json
import argparse

NOTIFICATION_TEXT = [
    "Осталось менее полугода до указанной даты следующего обучения преподавателя {}.",
    "Меньше, чем через три месяца наступит указанная дата следующего обучения преподавателя {}. Необходимо организовать обучение и поменять даты а базе.",
    "Меньше, чем через 30 дней наступит (или уже наступила) указанная дата переобучения преподавателя {}! Необходимо организовать обучение и поменять даты а базе."
]

IMPORTANCE = {
    180: "0", # важность низкая
    90: "1", # важность нейтральная
    30: "2" # важность высокая
}

"""
delta -- число дней, которое используется для проверки сроков. В def
check_training(self) модели Worker (models.py) рассчитывается
разница (current_delta) между текущей датой и той датой, в которую
преподавателю необходимо пройти следующее обучение.
Если разница меньше или равно delta, до даты обучения осталось меньше delta
дней, а значит нужно организовать прохождение переобучения преподавателем.
Когда даты последнего актуального и ближайшего следующего обучения будут
изменены, скрипт не будет реагировать на преподавателя,
пока не подойдут новые сроки. Уведомления добавляются с помощью API и
рассылаются на почту с помощью lifecycle hook AFTER_CREATE модели
Notification.
"""

def get_workers(delta: int) -> list:
    """
    Принимает число delta,
    отправляет запрос к адресу API для проверки актуальности данных о сроках
    образования,
    возвращает список преподавателей.

    delta -- число дней для проверки сроков
    """
    response: requests.Response = requests.get(
        "http://127.0.0.1:8000/workersdb/workers/check/{}".format(delta)
    )

    json_body = response.content

    body = json.loads(json_body)

    print(body)

    return body

def create_notification(notification_data: object) -> requests.Response:
    """
    Принимает объект notification_data,
```

```

отправляет запрос к адресу API для создания уведомлений,
возвращает HTTP-Response.
"""
response: requests.Response = requests.post(
    "http://127.0.0.1:8000/workersdb/workers/notifications/create",
    data=json.dumps(notification_data),
    headers={"Content-type": "application/json"})
)
return response

def get_notification_data(worker: object, importance: str) -> object:
    """
    Принимает объект worker и значение importance,
    возвращает объект notification_data с данными полей уведомления.
    """
    notification_data = {
        "worker": worker['id'],
        "importance": importance,
        "text":
NOTIFICATION_TEXT[int(importance)].format(worker['personal_info']['surname'])
    }

    return notification_data

def main(delta: int) -> None:
    """
    Принимает число delta,
    получается список преподавателей workers с помощью get_workers,
    создаёт уведомления для всех преподавателей.

    delta -- число дней для проверки сроков
    """
    workers = get_workers(delta)
    for worker in workers:
        notification_data = get_notification_data(worker, IMPORTANCE[delta])
        response = create_notification(notification_data)

parser = argparse.ArgumentParser(description='Число дней для проверки сроков
переподготовки')
parser.add_argument("--d", default=0, type=int, help="delta в днях.
Регулярный запуск выполняется с параметрами 30, 90, 180 (в порядке убывания
важности).")

args = parser.parse_args()

d = args.d

main(d)

```