|  |  |  |
| --- | --- | --- |
| Институт (факультет) | | Институт информационных технологий |
| Кафедра | Математического и программного обеспечения ЭВМ | |

КУРСОВАЯ РАБОТА

|  |  |
| --- | --- |
| по дисциплине | Спецификация, архитектура и проектирование ПО |

|  |  |
| --- | --- |
| на тему | Разработка спецификации и архитектуры ПО |

|  |
| --- |
| Выполнил студент группы |
| 1ПИб-01-2оп-21 |
| направление подготовки (специальности) |
| 09.03.04., Программная инженерия |
| шифр, наименование |
| Ульянов Александр Сергеевич |
| фамилия, имя, отчество |

|  |
| --- |
| Руководитель |
| Варфоломеев Игорь Андреевич |
| фамилия, имя, отчество  Кандидат технических наук, доцент |
|  |
| должность |

|  |
| --- |
| Дата представления работы |
| « » \_\_\_\_\_\_ 2025 г. |
| Заключение о допуске к защите |
|  |
|  |
|  |
| Оценка \_\_\_\_\_, \_\_\_\_\_\_ количество баллов |
|  |
| Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Оглавление

[Введение 2](#_Toc190514583)

[1. Основная часть 5](#_Toc190514584)

[1.1. Сравнительный анализ аналогов проектируемой системы 5](#_Toc190514585)

[1.2. Выбор технологии, среды и языка программирования 6](#_Toc190514586)

[1.3. Анализ процесса обработки информации, выбор структур данных для её хранения, выбор методов и алгоритмов решения задачи. 11](#_Toc190514587)

[1.4. Разработка спецификаций проектируемой системы 14](#_Toc190514588)

[1.4.1. Построение диаграммы вариантов использования 15](#_Toc190514589)

[1.4.2. Построение контекстной диаграммы классов 29](#_Toc190514590)

[1.4.3. Построение диаграмм последовательностей системы 31](#_Toc190514591)

[1.4.4. Построение диаграмм деятельностей вариантов использования 41](#_Toc190514592)

[1.4.5. Построение диаграммы переходов состояний 42](#_Toc190514593)

[1.5. Проектирование системы 45](#_Toc190514594)

[1.5.1. Проектирование структуры системы и построение диаграмм пакетов 45](#_Toc190514595)

[1.5.2. Проектирование классов в пакетах 46](#_Toc190514596)

[1.5.2.1. Проектирование классов пакета «Models» 46](#_Toc190514597)

[1.5.2.1.1. Построение исходной диаграммы классов пакета «Models» 46](#_Toc190514598)

[1.5.2.1.2. Построение диаграммы последовательности взаимодействия объектов классов пакета «Model» 48](#_Toc190514599)

[1.5.2.1.3. Построение диаграммы кооперации пакета «Models» 50](#_Toc190514600)

[1.5.2.1.4. Построение уточнённой диаграммы классов пакета «Models» 50](#_Toc190514601)

[1.5.2.1.5. Построение детальной диаграммы классов пакета «Models» 51](#_Toc190514602)

[1.5.3. Построение диаграмм компонентов 56](#_Toc190514603)

[1.5.4. Построение диаграмм размещения 65](#_Toc190514604)

[1.6. Результаты проведения испытаний 66](#_Toc190514605)

[Заключение 68](#_Toc190514606)

[Список литературы 69](#_Toc190514607)

[Приложение 1. Техническое задание 70](#_Toc190514608)

[Приложение 2. Текст программы 81](#_Toc190514609)

# Введение

Современные образовательные учреждения, такие как Череповецкий государственный университет, активно внедряют цифровые технологии для автоматизации административных процессов. Одной из важных задач является управление стипендиями, которое включает расчет, распределение и учет выплат для студентов. Эффективное выполнение этой задачи требует создания специализированных программных решений, которые позволят упорядочить и упростить сложные процессы работы с данными.

Целью данного проекта является разработка программного обеспечения для управления стипендиями в университете. Проект включает серверную и клиентскую части, предназначенные для использования в Управлении информационных технологий (УИТ) города Череповца. Реализация данного программного обеспечения позволит оптимизировать работу сотрудников, сократить время на обработку данных и минимизировать риск ошибок при ведении списка студентов.

Основные задачи разработки включают:

* Создание серверной части для безопасного хранения данных и обеспечения их обработки.
* Разработку клиентской части для удобного взаимодействия сотрудников с системой.
* Интеграцию программного обеспечения с существующими базами данных и учетными системами университета.
* Обеспечение автоматизации расчета и формирования отчетов по стипендиям.
* Настройку прав доступа в зависимости от ролей пользователей (администраторы, жюри, студенты, проверяющие).

Для реализации проекта используются современные технологии: язык программирования Python с использованием Django для серверной части, а для клиентской части используется Views они состоят их HTML, CSS, JS. В качестве системы управления базами данных планируется использовать SQLite, что обеспечит надежное и быстрое хранение данных. Важной частью разработки станет внедрение пользовательского интерфейса, который будет простым, интуитивно понятным и адаптированным для работы на различных устройствах.

Результатом работы станет веб-приложение, которое позволит сотрудникам УИТ эффективно управлять процессом начисления и выплаты стипендий, обеспечивая высокий уровень надежности, безопасности и производительности системы.

## 1. Основная часть

## 1.1. Сравнительный анализ аналогов проектируемой системы

На сегодняшний день существует большое множество программных продуктов по работе с заявлениями на повышенную стипендию и все они имеют в себе различный функционал. Для выявления сильных и слабых мест сторонних решений был проведен технический сравнительный аудит следующих информационно-системных решений: веб-приложение Санкт-Петербургский Политехнический Университет Петра Великого

«Повышенная государственная академическая стипендия», инструкция Тюменского государственного университета «Повышенная государственная академическая стипендия», веб-приложение Санкт-Петербургского государтсвенного университета «Прием заявлений на академическую стипендию».

Анализ каждой системы представлен в таблице 1.

Таблица 1

Анализ аналогов

|  |  |  |
| --- | --- | --- |
| Наименование | Преимущества | Недостатки |
| Повышенная государственная академическая стипендия.  Санкт-Петербургский Политехнический Университет Петра Великого | + Показана дата приёма заявлений  + Есть инструкция  + Заполнение в электронном виде |  |
| Повышенная государственная академическая стипендия.  Тюменский государственный университет | + Показана дата приёма заявлений  + Есть инструкция | - Нет шаблона  - Отправка на почту  - Заполнение через Word |
| Прием заявлений на академическую стипендию.  Санкт-Петербургский государтсвенный университет | + Показана дата приёма заявлений  + Есть инструкция  + Заполнение в электронном виде |  |

1.2. Выбор технологии, среды и языка программирования

Существует множество моделей жизненного цикла. Все эти модели представляют собой цепочку шагов в последовательном построении, которая уникальна для своего типа и обеспечивает успех на этапе разработки. Выбор оптимальной модели способствует качественному проектированию и разработке, что влияет на конечное качество продукта.

Для разработки программного обеспечения «Стипендии ЧГУ» была выбрана каскадная модель жизненного цикла с возможностью возвратов (рис. 1). Этот подход обеспечивает поэтапное выполнение процесса разработки — от анализа требований до ввода системы в эксплуатацию. Каждый этап завершается перед началом следующего, но при необходимости возможен возврат на предыдущие стадии для внесения корректировок. Такая структура позволяет тщательно прорабатывать каждый шаг, снижая вероятность ошибок и повышая качество конечного продукта.

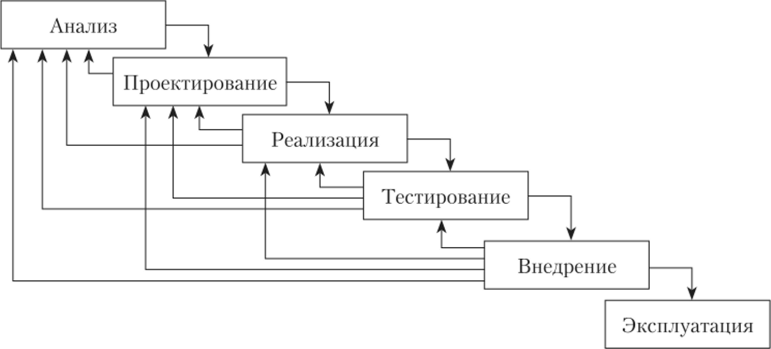


Рис.1. Каскадная модель жизненного цикла с возвратами

Основные этапы каскадной модели с возвратами:

Анализ требований – определение целей, задач и ограничений проекта, формализация всех требований к системе.

Проектирование – разработка архитектуры системы с учетом функциональных и нефункциональных требований.

Реализация – написание кода на основе проектной документации.

Тестирование – проверка системы на соответствие заданным требованиям, выявление и исправление ошибок.

Внедрение – установка программного обеспечения и обучение пользователей.

Эксплуатация и поддержка – обеспечение стабильной работы системы, устранение возможных проблем и доработка функционала при необходимости.

Преимущества каскадной модели с возвратами:

Структурированность – чёткое разделение этапов с возможностью возврата для исправления недочетов.

Прогнозируемость – заранее определенные сроки и бюджет разработки.

Стабильность – фиксация результатов каждого этапа снижает риск критических ошибок.

Объектно-ориентированный подход основан на постоянном использовании моделей для разработки программной среды, на основе из ее целей, таких как обслуживание крупных корпоративных клиентов или же работа с клиентами предприятия, обработка информации для правильной работы сложных систем. При использовании объектно-ориентированного подхода создаются модели, для воссоздания определенных формальных конструкций. Модель содержит в себе не все свойства и качества основного объекта, а лишь те, которые важны для разрабатываемой информационной системы. Тем самым модель меньше объекта, а значит и проще для восприятия и воссоздания, но при этом в модели сохраняются зависимости и понятия необходимые для разработки. Это упрощает разработку, тестирование и реализацию продукта.

Преимущества объектно-ориентированного подхода:

* объектная декомпозиция сокращает размер программных систем, используя общие механизмы и повышает уровень унификации и повторного использования. Объектно-ориентированный подход приводит к более компактным системам и уменьшает объем кода, что снижает стоимость проекта;
* объектная декомпозиция уменьшает риск создания сложных систем программного обеспечения, так как она предполагает эволюционный путь развития системы на базе относительно небольших подсистем. Процесс интеграции системы растягивается на все время разработки, а не превращается в единовременное событие;
* объектная модель вполне естественна, поскольку в первую очередь ориентирована на человеческое восприятие мира, а не на компьютерную реализацию;
* объектная модель позволяет в полной мере использовать выразительные возможности объектных и объектно-ориентированных языков программирования.

Минусы объектно-ориентированного подхода:

* разработка больших систем и их использование требует больших трудозатрат;
* сложность ведения документации, из-за повышенных трудозатрат;
* сложность иерархии приводит к сложности определения какие поля и классы фактически относятся к данному объекту;
* код обращения очень велик, в виду чего необходимо использовать несколько методов для его прочтения.

Для разработки архитектуры сложного приложения требуется создать визуальное описание предметной области, что обеспечит моделирование системного проектирования и отображение организационных структур. Для графического представления был выбран язык UML.

Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем. UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных веб-приложений и даже встроенных систем реального времени. UML – это язык для визуализации, специфицирования, конструирования и документирования артефактов программных систем [13].

UML позволяет решить проблему документирования системной архитектуры и всех ее деталей, предлагает язык для формулирования требований к системе и определения тестов и, наконец, предоставляет средства для моделирования работ на этапе планирования проекта и управления версиями.

Для реализации проекта были выбраны Django в качестве веб-фреймворка и SQLite как система управления базами данных. Эти технологии идеально подходят для задач, требующих быстрого развертывания и надежного хранения данных на начальных этапах разработки.

Django — это высокоуровневый веб-фреймворк на языке Python, который предоставляет удобные инструменты для быстрого создания мощных веб-приложений. Основные преимущества Django включают:

1. Быстрое прототипирование: Django позволяет ускорить процесс разработки благодаря встроенной ORM, автоматической генерации интерфейсов администратора и обширной экосистеме модулей.
2. Безопасность: Встроенные механизмы защиты от SQL-инъекций, межсайтового скриптинга (XSS) и других уязвимостей обеспечивают надежность создаваемого приложения.
3. Масштабируемость: несмотря на простоту начального использования, Django легко масштабируется для работы с большими проектами и высокими нагрузками.
4. Соответствие стандартам: Фреймворк поддерживает MVC-архитектуру и предоставляет интуитивно понятную структуру проектов.

SQLite была выбрана в качестве базы данных из-за ее легкости, простоты в настройке и отсутствия необходимости в дополнительной установке серверного ПО. Преимущества SQLite:

1. Легкость в использовании: SQLite интегрирована в Django как база данных по умолчанию, что упрощает процесс настройки.
2. Компактность: Она использует файл для хранения всех данных, что особенно удобно на ранних этапах разработки.
3. Поддержка стандартов SQL: SQLite полностью поддерживает SQL-92, обеспечивая совместимость с большинством функциональных требований.
4. Удобство разработки: Простота миграции данных и интеграции с Django ORM делает SQLite оптимальным выбором для небольших проектов и прототипов.

В качестве среды разработки был выбран PyCharm, популярный редактор кода, который идеально подходит для работы с Python и Django благодаря своей расширяемости и удобству.

Среди ключевых преимуществ PyCharm:

* Поддержка Django: PyCharm предоставляет встроенные инструменты для работы с Django, включая поддержку маршрутизации, управление моделями, работу с шаблонами и админ-интерфейсами. Автозаполнение кода и подсветка синтаксиса помогают быстрее разрабатывать приложения.
* Интеграция с SQLite: В PyCharm предусмотрены встроенные средства работы с базами данных, включая SQLite. Это позволяет выполнять SQL-запросы, просматривать данные и управлять структурой базы непосредственно из среды разработки.
* Мощные инструменты отладки: PyCharm предоставляет визуальный отладчик, который упрощает поиск ошибок, анализ выполнения кода и тестирование приложений.
* Расширяемость и удобство: Поддержка плагинов и интеграция с системами контроля версий (например, Git) делают PyCharm универсальной средой для проектов любой сложности.

Таким образом, использование PyCharm совместно с Django и SQLite обеспечивает удобную, мощную и профессиональную платформу для разработки веб-приложений, что позволяет значительно упростить создание, управление и тестирование кода.

### 1.3. Анализ процесса обработки информации, выбор структур данных для её хранения, выбор методов и алгоритмов решения задачи.

В настоящее время процесс подачи заявлений на повышенную стипендию в Череповецком государственном университете осуществляется вручную через использование таблиц Excel. Студенты заполняют шаблон Excel-файла и отправляют его на указанную электронную почту. После этого сотрудники проверяют каждую заявку вручную, что требует значительных временных и трудовых ресурсов. Такой подход затрудняет оперативность работы и увеличивает вероятность ошибок.

Для оптимизации процесса было предложено разработать специализированное программное обеспечение «Система управления стипендиями Череповецкого государственного университета», которое исключит использование файлов Excel и автоматизирует значительную часть проверок заявлений. Система будет функционировать через единое веб-приложение, где все участники процесса смогут выполнять свои задачи без необходимости скачивания или отправки файлов Excel.

Основой системы станет распределение ролей, каждая из которых будет иметь чёткие функции:

* Студент: подаёт заявление через личный кабинет на сайте, указывая все необходимые данные.
* Проверяющий: оценивает заявление, выставляет баллы и фиксирует замечания, если они имеются.
* Жюри: принимает решения в спорных случаях, может одобрить или отклонить заявление.
* Администратор: задаёт ключевые параметры системы, такие как сроки подачи заявлений, а также контролирует общий процесс.

Вся работа сосредоточена в одном интерфейсе, что обеспечивает прозрачность и удобство. Процесс подачи заявлений будет более структурированным: система автоматически проверяет корректность заполнения данных, снижая нагрузку на проверяющих.

В связи с поставленной задачей приложение будет разработано на Python с использованием Django и базы данных SQLite. Оно предназначено для работы с информацией о пользователях, ролях, заявлениях, их полях, типах данных и правах доступа.

Основные этапы обработки данных включают сбор данных через веб-интерфейс, их валидацию, управление правами доступа, сохранение в базе данных и извлечение для отображения. Проверка прав доступа пользователей играет ключевую роль в обеспечении безопасности и корректности выполнения операций.

Для реализации хранения данных была выбрана реляционная база данных SQLite. Это легковесное решение идеально подходит для начальных этапов разработки и тестирования приложений. Структура базы данных разработана с учетом задач приложения и включает таблицы для работы с пользователями, ролями, заявлениями и их содержимым. Основные таблицы базы данных:

1)STATMENT

Хранит данные о заявлениях, включая:

* ID: String – Уникальный идентификатор
* USER: Int – Фамилия, имя, отчество пользователя
* STATUS: Int – Статус заявления
* DATE: Date – Дата подачи заявления

2) Log

Хранит историю действий на сайте, включая:

* ID: Int – Уникальный идентификатор
* EVENT: String – Название действия
* STATMENT\_COPY: JSON – Копия данных всех действий
* DATA: String – Дата действия
* USER: Int – ID пользователя
* DATE: Date – Комментарии к действию

3) PERIOD

Хранит даты о начале и конце сбора заявлений, включая:

* ID: Int – Уникальный идентификатор
* DATE: String – Дата
* DESCRIPTION: String – Описание

4) ROLE

Хранит роли пользователей, включая:

* ID: Int – Уникальный идентификатор
* ROLE: String – Роль пользователя

5) STATUS

Хранит все статусы заявлений, включая:

* ID: Int – Уникальный идентификатор
* STATUS: String – Статус заявлений

6) USER

Хранит все сведения о пользователях, включая:

* ID: String – Уникальный идентификатор
* SESSION: String – Уникальный идентификатор сессии
* LOGIN: String – Логин пользователя
* ROLE: Int – Роль пользователя
* SURNAME: String – Фамилия пользователя
* NAME: String – Имя пользователя
* MIDDLENAME: String – Отчество пользователя
* EMAIL: String – Почта пользователя
* PHONE: String – Телефон пользователя
* GROUP: String – Группа пользователя

С помощью Django данные обрабатываются через уровни модели (Models), представлений (Views) и шаблонов (Templates), которые связаны с SQLite через встроенный ORM Django.

Основные задачи обработки:

* Создание, чтение, обновление и удаление данных (CRUD). Эти операции выполняются с использованием моделей Django и встроенных функций ORM. Для выполнения запросов используются методы objects.create(), objects.get(), objects.filter(), objects.update(), и objects.delete().
* Проверки выполняются на основе данных таблиц, связанных с группами, разрешениями и пользователями через User и Role.

Для повышения производительности системы SQLite поддерживает индексы на ключевых полях, таких как user\_id в таблицах, используемых для прав доступа и связей. Индексы создаются автоматически для первичных и внешних ключей, а дополнительные можно добавить через параметр index=True в модели.

Использование SQLite и Django предоставляет простую и удобную архитектуру, позволяющую эффективно обрабатывать данные, управлять правами доступа и поддерживать высокую производительность на начальных этапах разработки или для небольших проектов.

### 1.4. Разработка спецификаций проектируемой системы

Спецификация — это декларативное описание того, как нечто устроено или работает.

Объектный подход к разработке программного обеспечения основан на объектной декомпозиции. Это означает представление программного обеспечения как набора объектов для эффективного взаимодействия и выполнения нужных функций.

### 1.4.1. Построение диаграммы вариантов использования

Разработку спецификаций программного обеспечения начинают с анализа требований к функциональности, указанных в техническом задании. В процессе анализа выявляют внешних пользователей разрабатываемого программного обеспечения и перечень отдельных аспектов его поведения в процессе взаимодействия с конкретными пользователями.

Вариант использования представляет собой характерную процедуру применения разрабатываемой системы конкретным действующим лицом, в качестве которого могут выступать не только люди, но и другие системы или устройства.

Для информационной системы была построена диаграмма вариантов использования, в основе которой выступает студент, проверяющий, система, жюри, администратор (рис.3-4).

В приложении возможны варианты взаимодействия для пяти ключевых лиц — студент, проверяющий, система, жюри, администратор.

Функционал ролей:

1. Студент

Студенты взаимодействуют с системой через функции подачи и редактирования заявлений:

* Отправить заявление на проверку: студенты заполняют форму заявления и направляют её для рассмотрения.
* Просмотреть свои заявления: пользователи могут отслеживать статус своих заявлений.
* Заполнить/редактировать заявление: до отправки на проверку студент может внести изменения или дополнения в заявление.

2. Проверяющий

Роль проверяющего предназначена для оценки и управления поданными заявлениями в рамках его компетенции:

* Просмотреть часть заявления согласно своему направлению: доступ к тем частям заявки, которые входят в компетенцию проверяющего.
* Просмотреть список заявлений: работа с заявлениями со статусами "Проверенное", "На проверке", "Ошибочное".
* Оценить часть заявления: проверка и оценка содержимого заявления в соответствии с направлением деятельности проверяющего.

3. Жюри

Жюри участвует на этапе финальной оценки заявлений:

* Просмотреть заявление: возможность изучения заявлений целиком.
* Просмотреть список заявлений: доступ к заявлениям со статусами "Отклонённое", "Конфликтное", "Одобренное".
* Одобрить/отклонить заявление: принятие финального решения по заявке.

4. Система

Система выполняет автоматические операции, минимизируя человеческое вмешательство:

* Формирование рейтинга: автоматически генерирует рейтинг студентов по завершении срока подачи заявлений. Эта функция помогает систематизировать данные для последующего анализа.

5. Администратор

Администратор обладает наибольшими возможностями для управления системой и её пользователями:

* Просмотреть и очистить историю действий: мониторинг и управление журналами событий системы.
* Установить сроки подачи заявлений: настройка дат начала и окончания периода подачи.
* Кэшировать заявления: сохранение заявлений для ускорения доступа к ним.
* Просмотреть заявление и список заявлений: доступ к заявкам на всех этапах их обработки.
* Изменить статус заявления: возможность ручной корректировки статуса заявки.
* Управление пользователями: просмотр данных пользователей, смена их ролей, управление списком пользователей.
* Работа с устаревшими заявлениями: доступ к сохранённым копиям заявлений для анализа или восстановления данных.

Для того, чтобы понять как организован и работает каждый вариант использования были построены таблицы с кратким описанием и ходом событий (табл.2-43).

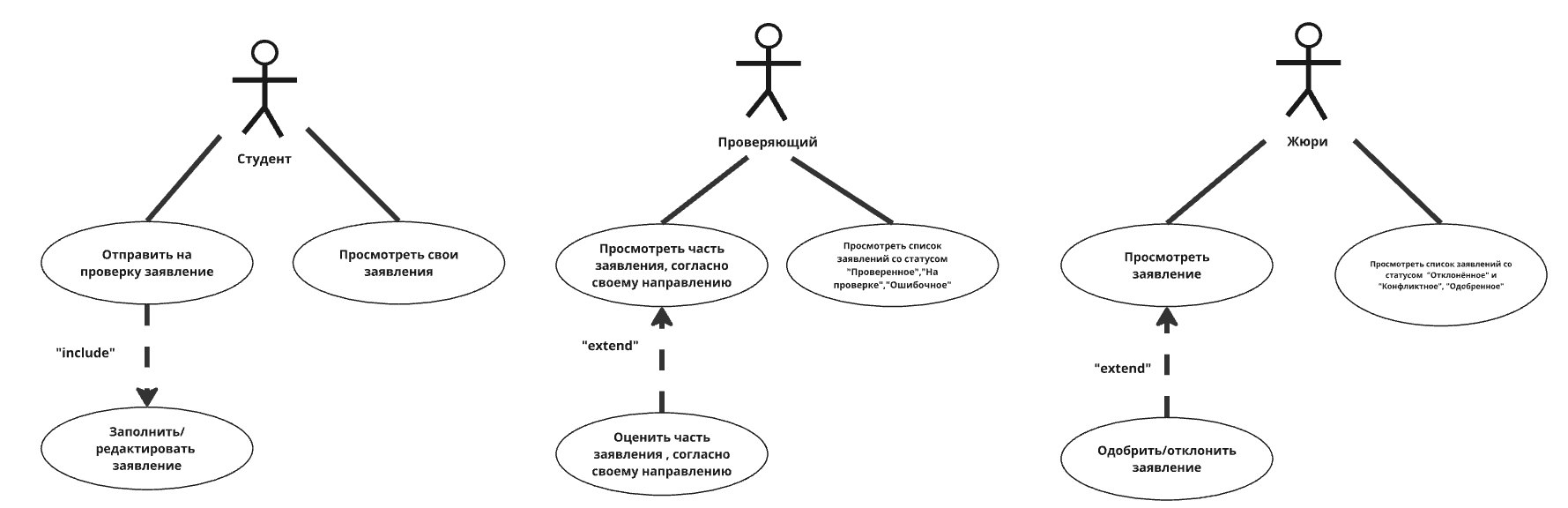


Рис. 2. Диаграмма вариантов использования

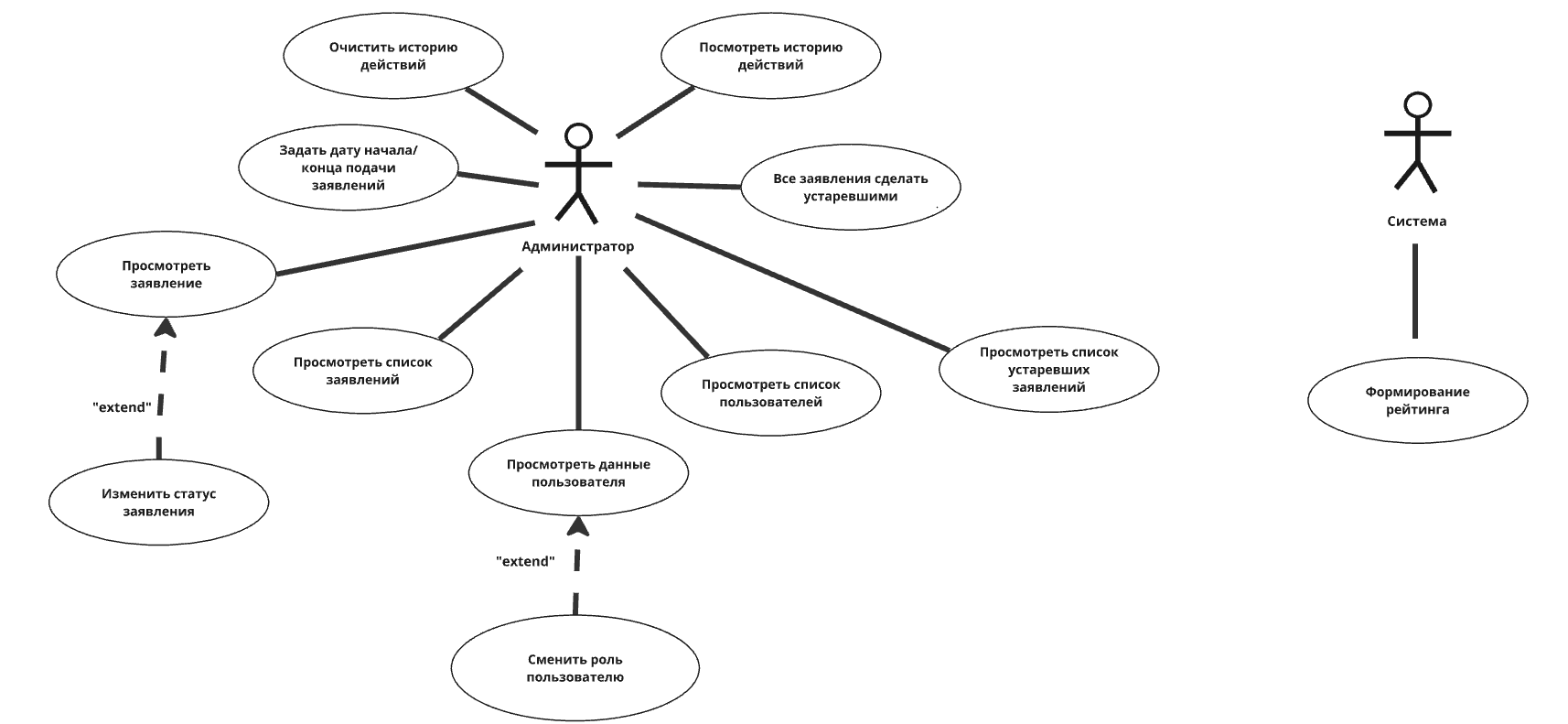


Рис. 3. Диаграмма вариантов использования

На диаграмме изображен один основной пользователь. В его распоряжении возможность производить настройку системы и получать информацию о её работе, формировать отчёты на основе данных распознавания. Для каждого варианта использования были составлены описания, которые представлены в табл. 2–43.

Студент:

Описание блока «Отправить на проверку»

Краткое описание варианта использования «Отправить на проверку заявление на повышенную стипендию» представлено в табл. 2.

Таблица 2

Краткое описание варианта «Отправить на проверку заявление»

|  |  |
| --- | --- |
| Название | Отправить на проверку заявление |
| Назначение | Отправка системой заявления на обработку проверяющему |
| Действующие лица | Студент |
| Краткое описание | Пользователем заполняются необходимые поля: вид деятельности, заполнение форм именно этой деятельности для отправки заявления. Далее заявление система уже отправляет проверяющему |

Типичный ход событий для данного варианта представлен в табл. 3.

Таблица 3

Типичный ход событий варианта «Отправить на проверку заявление»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Нажатие на кнопку отправить | 2)Передача заявления проверяющему |

Описание блока «Отправить на проверку заявление на повышенную стипендию».

Краткое описание варианта использования «Просмотреть свои заявления» представлено в табл. 4.

Таблица 4

Краткое описание варианта «Просмотреть свои заявления»

|  |  |
| --- | --- |
| Название | Просмотреть свои заявления |
| Назначение | Просмотр заявлений на повышенную стипендию, их статус и данные |
| Действующие лица | Студент |
| Краткое описание | Пользователь заходит в личный кабинет, где у него находятся все поданные заявления и статус рассмотрения заявки |

Типичный ход событий для данного варианта представлен в табл. 5.

Таблица 5

Типичный ход событий варианта использования «Просмотреть свои заявления»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Открытие личного кабинета | 2)Показ данных о студенте |

Описание блока «Заполнить/редактировать заявление».

Краткое описание варианта использования «Заполнить/редактировать заявление на повышенную сти﻿пендию» представлено в табл. 6.

Таблица 6

Краткое описание варианта «Заполнить/редактировать заявление»

|  |  |
| --- | --- |
| Название | Заполнить/редактировать заявление |
| Назначение | Редактирование уже отправленного заявления или заполнение нового |
| Действующие лица | Студент |
| Краткое описание | Пользователь заходит в личный кабинет, где у него находятся все поданные заявления и статус рассмотрения заявки, можно нажать на заявку и поменять что-то или добавить, если упущена информация или неверна. |

Типичный ход событий для данного варианта представлен в табл. 7.

Таблица 7

Типичный ход событий варианта «Заполнить/редактировать заявление»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Открытие файла для редактирования | 2)Предоставление формы для добавления или редактирования документа |

Проверяющий:

Описание блока «Оценить часть заявления, согласно своему направлению» Краткое описание варианта использования «Оценить часть заявления, согласно своему направлению» представлено в табл. 8.

Таблица 8

Краткое описание варианта «Оценить часть заявления, согласно своему направлению»

|  |  |
| --- | --- |
| Название | Оценить часть заявления, согласно своему направлению |
| Назначение | Оценка корректности поданного заявления |
| Действующие лица | Проверяющий |
| Краткое описание | В личном кабинете открывается список заявлений на проверку, выбираем желаемое заявление и проверяем корректность данных. |

Типичный ход событий для данного варианта представлен в табл. 9.

Таблица 9

Типичный ход событий варианта «Оценить часть заявления, согласно своему направлению»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Открытие файла для проверки корректности данных | 2) Вывод содержимого файла для дальнейшей проверки |

Описание блока «Просмотреть список заявлен﻿ий со статусом "Проверенное","Н﻿а проверке","Ошибочное"».

Краткое описание варианта использования «Просмотреть список заявлений на повышенную стипендию со статусом "От﻿крыто"» представлено в табл. 10.

Таблица 10

Краткое описание варианта «Просмотреть список заявлен﻿ий со статусом "Проверенное","Н﻿а проверке","Ошибочное"»

|  |  |
| --- | --- |
| Название | Просмотреть список заявлен﻿ий со статусом "Проверенное","Н﻿а проверке","Ошибочное" |
| Назначение | Просмотр заявлений c разными статусами для дальнейшей работы |
| Действующие лица | Проверяющий |
| Краткое описание | Вывод списка заявлений разбитый на категории: "Проверенное","Н﻿а проверке","Ошибочное" |

Типичный ход событий для данного варианта представлен в табл. 11.

Таблица 11

Типичный ход событий варианта «Просмотреть список заявлен﻿ий со статусом "Проверенное","Н﻿а проверке","Ошибочное"»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Открывает вкладку для просмотра заявлений | 2) Вывод списка заявлений, разбитый на категории |

Описание блока «Просмотреть часть заявления, согласно своему направлению».

Краткое описание варианта использования «Просмотреть ч﻿асть выбранного из списка заявления, согласно своему направлению» представлено в табл. 12.

Таблица 12

Краткое описание варианта «Просмотреть ч﻿асть заявления, согласно своему направлению»

|  |  |
| --- | --- |
| Название | Просмотреть ч﻿асть заявления, согласно своему направлению |
| Назначение | Просмотр заявления на проверку |
| Действующие лица | Проверяющий |
| Краткое описание | Просмотр заявления с целью удостовериться, что оно заполнено верно, чтобы в дальнейшем его оценить |

Типичный ход событий для данного варианта представлен в табл. 13.

Таблица 13

Типичный ход событий варианта «Просмотреть ч﻿асть заявления, согласно своему направлению»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Выбор заявления на проверку | 2) Предоставление содержимого заявления |

Жюри:

Описание блока «Одобрить/отклонить заявление».

Краткое описание варианта использования «Одобрить/отклонить заявление» представлено в табл. 14.

Таблица 14

Краткое описание варианта «Одобрить/отклонить заявление»

|  |  |
| --- | --- |
| Название | Одобрить/отклонить заявление |
| Назначение | Одобрение или отклонение заявления на повышенную стипендию |
| Действующие лица | Жюри |
| Краткое описание | В зависимости от одобрения или отклонения заявления оно либо будет в итоговом списке, либо его не будет |

Типичный ход событий для данного варианта представлен в табл. 15.

Таблица 15

Типичный ход событий варианта «Одобрить/отклонить заявление»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Выбирает заявление и либо одобряет, либо отклоняет | 2) Предоставляет данные заявления и в зависимости от решения жюри, либо добавляет в итоговый список заявление, либо не добавляет |

Описание блока «Просмотреть список заявлений со статусом "Проверенное", "Отклонённое" и "Конфликтное", "Одобренное", "Проверенное﻿"».

Краткое описание варианта использования «Просмотреть список заявлений со статусом "Проверенное", "Отклонённое" и "Конфликтное", "Одобренное", "Проверенное﻿"»» представлено в табл. 16.

Таблица 16

Краткое описание варианта «Просмотреть список заявлений со статусом «Отклонённое" и "Конфликтное", "Одобренное", "Проверенное﻿"»

|  |  |
| --- | --- |
| Название | Просмотреть список заявлений со статусом «Отклонённое" и "Конфликтное", "Одобренное", "Проверенное﻿"» |
| Назначение | Просмотр списка заявлений |
| Действующие лица | Жюри |
| Краткое описание | Просмотр списка заявлений, разбитый на категории: "Отклонённое" и "Конфликтное", "Одобренное", "Проверенное﻿" |

Типичный ход событий для данного варианта представлен в табл. 17

Таблица 17

Типичный ход событий варианта «Просмотреть список заявлений со статусом "Отклонённое" и "Конфликтное", "Одобренное", "Проверенное﻿"»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Открывает вкладку для просмотра заявлений | 2) Вывод списка заявлений, разбитый на категории: "Отклонённое" и "Конфликтное", "Одобренное", "Проверенное﻿" |

Описание блока «Просмотреть зая﻿вление».

Краткое описание варианта использования «Просмотреть зая﻿вление» представлено в табл. 18.

Таблица 18

Краткое описание варианта «Просмотреть зая﻿вление»

|  |  |
| --- | --- |
| Название | Просмотреть зая﻿вление |
| Назначение | Нужно для того, чтобы просмотреть заявление |
| Действующие лица | Жюри |
| Краткое описание | Просмотр заявления, чтобы в дальнейшем вынести вердикт |

Типичный ход событий для данного варианта представлен в табл. 19.

Таблица 19

Типичный ход событий варианта «Просмотреть зая﻿вление»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Открывает заявление | 2) Показ данных, заполненных пользователем |

Администратор:

Описание блока «Просмот﻿реть список заявлений на повышенную стипендию».

Краткое описание варианта использования «Просмот﻿реть список заявлений» представлено в табл. 20.

Таблица 20

Краткое описание варианта «Просмот﻿реть список заявлений»

|  |  |
| --- | --- |
| Название | Просмот﻿реть список заявлений |
| Назначение | Нужно, на тот случай если будет ошибка в вердикте |
| Действующие лица | Администратор |
| Краткое описание | Если вдруг ошибется в вердикте, либо проверяющий, либо жюри, можно было убрать заявление из списка |

Типичный ход событий для данного варианта представлен в табл. 21.

Таблица 21

Типичный ход событий варианта «Просмот﻿реть список заявлений»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Просмотр списка заявлений | 2) Вывод списка всех заявлений |

Описание блока «Просмотреть данные пользователя».

Краткое описание варианта использования «Просмотреть данные пользователя» представлено в табл. 22.

Таблица 22

Краткое описание варианта «Просмотреть данные пользователя»

|  |  |
| --- | --- |
| Название | Просмотреть данные пользователя |
| Назначение | Просмотр данных пользователя для изменения роли |
| Действующие лица | Администратор |
| Краткое описание | Нужно, чтобы добавить кому-то роль, либо убрать ее |

Типичный ход событий для данного варианта представлен в табл. 23.

Таблица 23

Типичный ход событий варианта «Просмотреть данные пользователя»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Нажатие на вкладку для пользователей | 2) Вывод списка пользователей |

Описание блока «Сменить роль пользователю».

Краткое описание варианта использования «Сменить роль пользователю» представлено в табл. 24.

Таблица 24

Краткое описание варианта «Сменить роль пользователю»

|  |  |
| --- | --- |
| Название | Сменить роль пользователю |
| Назначение | Смена роли пользователям |
| Действующие лица | Администратор |
| Краткое описание | Смена роли пользователям в зависимости от надобности, кому-то дать, у кого-то забрать |

Типичный ход событий для данного варианта представлен в табл. 25.

Таблица 25

Типичный ход событий варианта «Сменить роль пользователю»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Выбор пользователя и нажатие на кнопку дать кому-то роль или отобрать | 2)Вывод данных пользователя и в зависимости от выбора, либо дать кому-то права, либо забрать |

Описание блока «Просмотреть список пользователей».

Краткое описание варианта использования «Просмотреть список пользователей» представлено в табл. 26.

Таблица 26

Краткое описание варианта «Просмотреть список пользователей»

|  |  |
| --- | --- |
| Название | Просмотреть список пользователей |
| Назначение | Просмотр списка пользователей для дальнейших действий |
| Действующие лица | Администратор |
| Краткое описание | Просмотр списка, чтобы в дальнейшем уже дать кому-то роль или отобрать или просто просмотреть его данные |

Типичный ход событий для данного варианта представлен в табл. 27.

Таблица 27

Типичный ход событий варианта «Просмотреть список пользователей»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Вход на вкладку для просмотра списка пользователей | 2) Вывод списка пользователей |

Описание блока «Просмотреть список пользователей».

Краткое описание варианта использования «Просмотреть список пользователей» представлено в табл. 28.

Таблица 28

Краткое описание варианта «Задать дату начала/конца подачи заявлен﻿ий»

|  |  |
| --- | --- |
| Название | Задать дату начала/конца подачи заявлен﻿ий |
| Назначение | Задача даты конца и начала подачи |
| Действующие лица | Администратор |
| Краткое описание | Требуется для того, чтобы ограничить поток заявок в срок и далее передать после срока заявки жюри |

Типичный ход событий для данного варианта представлен в табл. 29.

Таблица 29

Типичный ход событий варианта «Задать дату начала/конца подачи заявлен﻿ий»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Вход на вкладку для задачи даты и вводит дату далее сохраняет ее | 2) Сохраняет даты и выводит далее у всех пользователей |

Описание блока «Изменить статус заявления».

Краткое описание варианта использования «Изменить статус заявления» представлено в табл. 30.

Таблица 30

Краткое описание варианта «Изменить статус заявлени﻿я»

|  |  |
| --- | --- |
| Название | Изменить статус заявлени﻿я |
| Назначение | Изменение списка заявлений |
| Действующие лица | Администратор |
| Краткое описание | Необходимо для того, чтобы контролировать процесс проверки, если кто-то ошибется и заявление пройдет куда не должно |

Типичный ход событий для данного варианта представлен в табл. 31.

Таблица 31

Типичный ход событий варианта «Изменить статус заявлени﻿я»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Открывает заявление из списка | 2) Вывод данных о заявлении и если требуется поменять статус его |

Описание блока «Просмотреть з﻿аявлени﻿е».

Краткое описание варианта использования «Просмотреть з﻿аявлени﻿е» представлено в табл. 32.

Таблица 32

Краткое описание варианта «Просмотреть з﻿аявлени﻿е»

|  |  |
| --- | --- |
| Название | Просмотреть з﻿аявлени﻿е |
| Назначение | Просмотреть одно заявление |
| Действующие лица | Администратор |
| Краткое описание | Необходимо для того, чтобы контролировать процесс проверки в дальнейшем, если кто-то ошибется и заявление пройдет куда не должно |

Типичный ход событий для данного варианта представлен в табл. 33.

Таблица 33

Типичный ход событий варианта «Просмотреть з﻿аявлени﻿е»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Открывает заявление из списка | 2) Вывод данных о заявлении и если требуется поменять статус его |

Описание блока «Очистить историю действий».

Краткое описание варианта использования «Очистить историю действий» представлено в табл. 34.

Таблица 34

Краткое описание варианта «Очистить историю действий»

|  |  |
| --- | --- |
| Название | Очистить историю действий |
| Назначение | Очистка всех действий |
| Действующие лица | Администратор |
| Краткое описание | Необходимо для того, чтобы было удобнее отслеживать работу программы |

Типичный ход событий для данного варианта представлен в табл. 35.

Таблица 35

Типичный ход событий варианта «Очистить историю действий»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Нажатие на кнопку очистка истории действий | 2) Очищение списка истории действий |

Описание блока «Посмотреть список истории действий».

Краткое описание варианта использования «Посмотреть список истории действий» представлено в табл. 36.

Таблица 36

Краткое описание варианта «Посмотреть список истории действий»

|  |  |
| --- | --- |
| Название | Посмотреть список истории действий |
| Назначение | Просмотр всех действий |
| Действующие лица | Администратор |
| Краткое описание | Необходимо для того, чтобы отслеживать работу программы, если вдруг что-то работает не так, чтобы понимать, где ошибка |

Типичный ход событий для данного варианта представлен в табл. 37.

Таблица 37

Типичный ход событий варианта «Посмотреть список истории действий»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Вход на страницу списка действий | 2) Вывод списка всех действий |

Описание блока «Все заявления﻿ сделать устаревшими».

Краткое описание варианта использования «Все заявления сделать устаревшими» представлено в табл. 38.

Таблица 38

Краткое описание варианта «Все заявления сделать устаревшими»

|  |  |
| --- | --- |
| Название | Все заявления сделать устаревшими |
| Назначение | Перевод всех заявлений |
| Действующие лица | Администратор |
| Краткое описание | Необходимо для того, чтобы выкачать все заявления, набравшиеся допустим за год, чтобы не засорялся список всех заявлений |

Типичный ход событий для данного варианта представлен в табл. 39.

Таблица 39

Типичный ход событий варианта «Все заявления сделать устаревшими»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Нажатие на кнопку «Сделать все заявления устаревшими» | 2) Перевод всех заявлений в статус «Устаревшие» всех заявлений |

Описание блока «Про﻿смотреть список устаревших заявлений» Краткое описание варианта использования «Про﻿смотреть список устаревших заявлений» представлено в табл. 40.

Таблица 40

Краткое описание варианта «Про﻿смотреть список устаревших заявлений»

|  |  |
| --- | --- |
| Название | Про﻿смотреть список устаревших заявлений |
| Назначение | Просмотреть список всех заявлений |
| Действующие лица | Администратор |
| Краткое описание | Необходимо для того, чтобы просмотреть корректность все ли заявления успешно записаны в статус «Устаревшие» |

Типичный ход событий для данного варианта представлен в табл. 41.

Таблица 41

Типичный ход событий варианта «Про﻿смотреть список устаревших заявлений»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Вход на страницу для просмотра списка «Уставревших» заявлений | 2) Вывод списка устаревших заявлений |

Система:

Описание блока «Формирование рейтинга».

Краткое описание варианта использования «Формирование рейтинга» представлено в табл. 42.

Таблица 42

Краткое описание варианта «Формирование рейтинга»

|  |  |
| --- | --- |
| Название | Формирование рейтинга |
| Назначение | Формирование рейтинга всех заявок |
| Действующие лица | Система |
| Краткое описание | Когда выходит срок на подачу заявления все заявления переходят в руки жюри |

Типичный ход событий для данного варианта представлен в табл. 43.

Таблица 43

Типичный ход событий варианта «Формирование рейтинга»

|  |  |
| --- | --- |
| Действие пользователя | Отклик системы |
| 1) Ожидание выхода срока заявок | 2) Передача заявок Жюри |

1.4.2. Построение контекстной диаграммы классов

Диаграмма классов – диаграмма, предназначенная для представления модели статической структуры программной системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы.

Для отображения структуры программного обеспечения была построена контекстная диаграмма классов (рис.5).

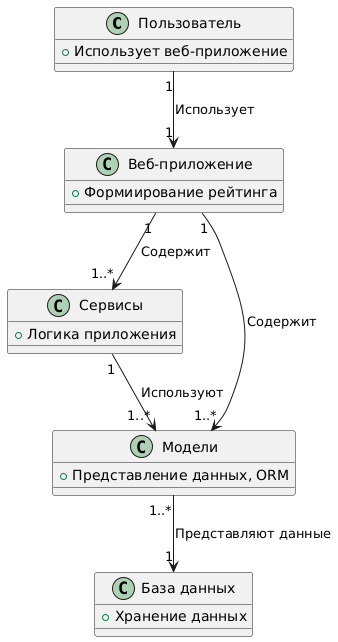


Рис.4. Контекстная диаграмма классов

Пользователи могут взаимодействовать с веб-приложением, выполняя различные действия. Эти действия связаны с хранимой в базе данных информацией, которая необходима для полноценного функционирования приложения и предоставления пользователю нужных данных.

Для краткого описания назначения классов была построена табл. 44.

Таблица 44

Описание классов

|  |  |
| --- | --- |
| Наименование сущности | Описание сущности |
| Пользователь | Пользователь взаимодействует с веб-приложением для управления заявлениями. |
| Веб-приложение | Центральный компонент, обеспечивающий обработку запросов пользователя, управление данными и создание, проверку, оценку заявлений. |
| Сервисы | Компонент, выполняющий бизнес-логику приложения, включая обработку данных для заявлений. |
| Модели | Компонент, представляющий структуру данных в приложении и определяющий, как данные сохраняются и используются. |
| База данных | Хранилище данных, содержащее информацию о заявлениях и других данных, необходимых для работы веб-приложения. |

Разработанная контекстная диаграмма классов подтверждает высокую степень обособленности сущностей, их единообязанность применительно к своим задачам, а также высокий уровень взаимодействия между объектами классов предметной области. Исходя из этого, выбор объектно-ориентированного подхода при разработке программного обеспечения полностью оправдан.

### 1.4.3. Построение диаграмм последовательностей системы

Диаграмма последовательности - диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определенного объекта и взаимодействие действующих лиц в рамках какого-либо определенного прецедента.

Для всех вариантов использования были построены диаграммы последовательности системы (рис.5-19).

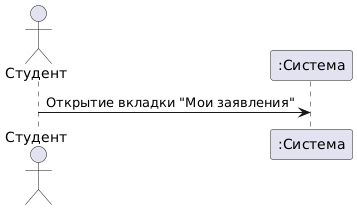


Рис.5. Диаграмма последовательности системы. Просмотреть с﻿вои заявления

Таблица 45

Описание операции «Просмотреть с﻿вои заявления»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Просмотреть свои заявления |
| Тип | Системная |
| Ссылки | Вариант использования «Просмотреть свои заявления» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Список заявлений студента |

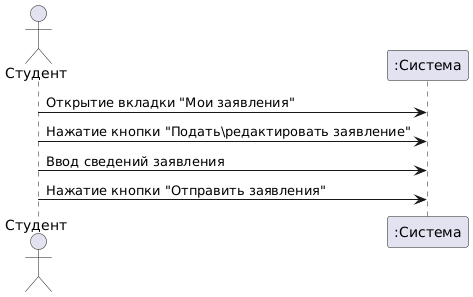


Рис.6. Диаграмма последовательности системы. Заполнить/редактировать заявлен﻿ие

Таблица 46

Описание операции «Заполнить/редактировать заявлен﻿ие»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Заполнить/редактировать заявлен﻿ие |
| Тип | Системная |
| Ссылки | Вариант использования «Заполнить/редактировать заявлен﻿ие» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Отправленное заявление |

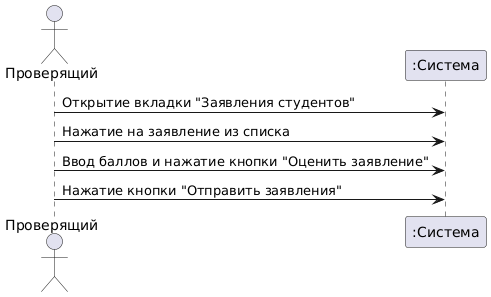


Рис.7. Диаграмма последовательности системы. Оценить часть заявления, согласно своему направ﻿л﻿ению

Таблица 47

Описание операции «Оценить часть заявления, согласно своему направ﻿л﻿ению»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Оценить часть заявления, согласно своему направ﻿л﻿ению |
| Тип | Системная |
| Ссылки | Вариант использования «Оценить часть заявления, согласно своему направ﻿л﻿ению» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Оцененное заявление |

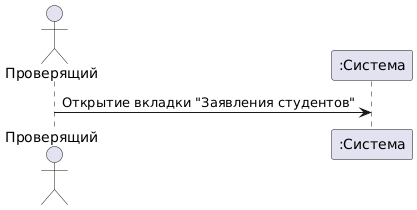


Рис.8. Диаграмма последовательности системы. Просмотреть список заявлен﻿ий со статусом "Проверенное","На проверке","Ошиб﻿очное"

Таблица 48

Описание операции «Оценить часть заявления, согласно своему направ﻿л﻿ению»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Просмотреть список заявлен﻿ий со статусом "Проверенное","На проверке","Ошиб﻿очное" |
| Тип | Системная |
| Ссылки | Вариант использования «Одобрить/отклонить за﻿явлени﻿е» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Список заявлений студентов |

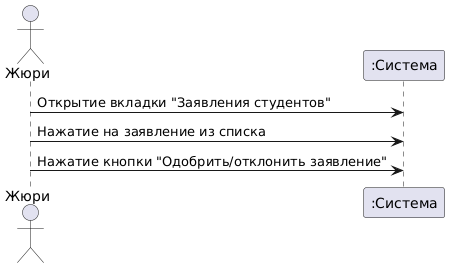


Рис.9. Диаграмма последовательности системы. Одобрить/отклонить за﻿явлени﻿е

Таблица 49

Описание операции «Одобрить/отклонить за﻿явлени﻿е»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Одобрить/отклонить за﻿явлени﻿е |
| Тип | Системная |
| Ссылки | Вариант использования «Одобрить/отклонить за﻿явлени﻿е» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Одобренное/Отклоненное заявление |

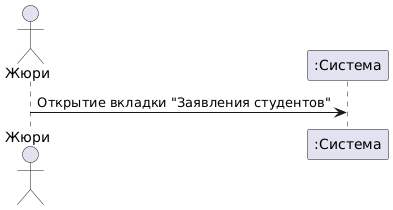


Рис.10. Диаграмма последовательности системы. Просмотреть список заявлений со статусом "Отклонённое" и "Конфликтное", "Одобренное"

Таблица 50

Описание операции «Просмотреть список заявлений со статусом "Отклонённое" и "Конфликтное", "Одобренное"»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Просмотреть список заявлений со статусом "Отклонённое" и "Конфликтное", "Одобренное" |
| Тип | Системная |
| Ссылки | Вариант использования «Просмотреть список заявлений со статусом "Отклонённое" и "Конфликтное", "Одобренное"» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Список заявлений студентов |



Рис.11. Диаграмма последовательности системы. Просмотреть список заявл﻿ений

Таблица 51

Описание операции «Просмотреть список заявл﻿ений»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Просмотреть список заявл﻿ений |
| Тип | Системная |
| Ссылки | Вариант использования «Просмотреть список заявл﻿ений» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Список заявлений студентов |

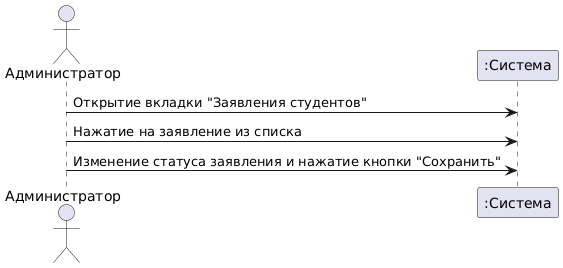


Рис.12. Диаграмма последовательности системы. Изменить статус заявлени﻿я

Таблица 52

Описание операции «Изменить статус заявлени﻿я»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Изменить статус заявлени﻿я |
| Тип | Системная |
| Ссылки | Вариант использования «Изменить статус заявлени﻿я» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Измененный статус у заявления |

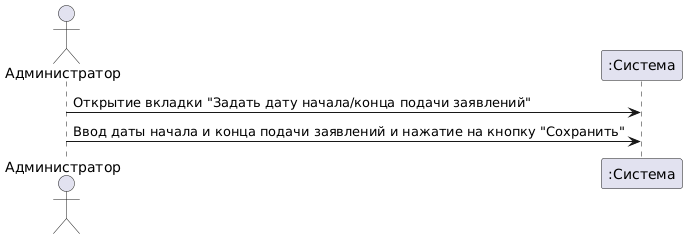


Рис.13. Диаграмма последовательности системы. Задать дату начала/конца подачи заявлений

Таблица 53

Описание операции «Задать дату начала/конца подачи заявлений»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Задать дату начала/конца подачи заявлений |
| Тип | Системная |
| Ссылки | Вариант использования «Задать дату начала/конца подачи заявлений» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Заданная дата начала и конца подачи заявлений |

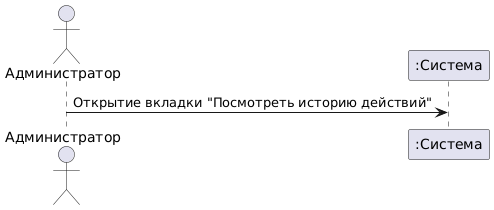


Рис.14. Диаграмма последовательности системы. Посмотреть историю дейс﻿твий

Таблица 54

Описание операции «Посмотреть историю дейс﻿твий»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Посмотреть историю дейс﻿твий |
| Тип | Системная |
| Ссылки | Вариант использования «Посмотреть историю дейс﻿твий» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Заданная дата начала и конца подачи заявлений |

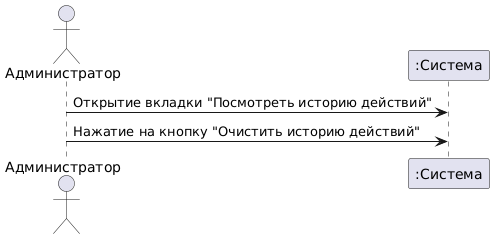


Рис.15. Диаграмма последовательности системы. Очистить историю дейст﻿вий

Таблица 55

Описание операции «Очистить историю дейст﻿вий»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Очистить историю дейст﻿вий |
| Тип | Системная |
| Ссылки | Вариант использования «Очистить историю дейст﻿вий» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Очищенная история действий |

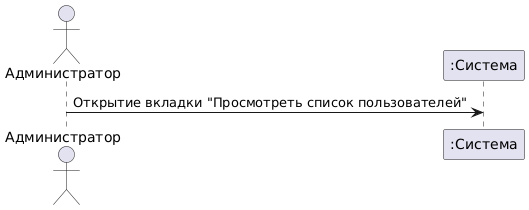


Рис.16. Диаграмма последовательности системы. Просмот﻿реть список пользователей

Таблица 56

Описание операции «Просмот﻿реть список пользователей»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Просмот﻿реть список пользователей |
| Тип | Системная |
| Ссылки | Вариант использования «Просмот﻿реть список пользователей» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Список пользователей |

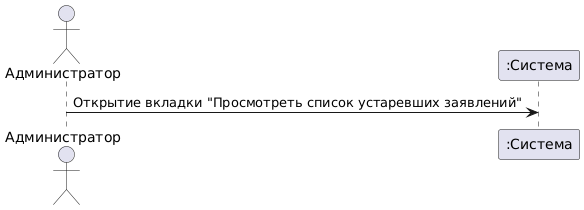


Рис.17. Диаграмма последовательности системы. Просмотреть список устаре﻿вших заявлений

Таблица 57

Описание операции «Просмотреть список устаре﻿вших заявлений»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Просмотреть список устаре﻿вших заявлений |
| Тип | Системная |
| Ссылки | Вариант использования «Просмотреть список устаре﻿вших заявлений» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Список устаревших заявлений |

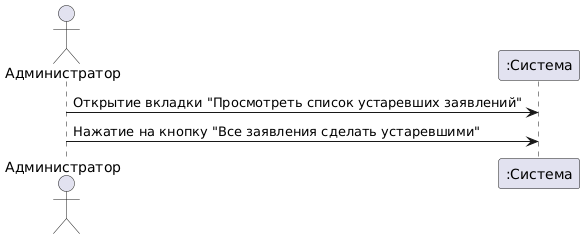


Рис.18. Диаграмма последовательности системы. Все заявления сделать устарев﻿шими

Таблица 58

Описание операции «Все заявления сделать устарев﻿шими»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Все заявления сделать устарев﻿шими |
| Тип | Системная |
| Ссылки | Вариант использования «Все заявления сделать устарев﻿шими» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Все заявления переходят в состояние устаревших |

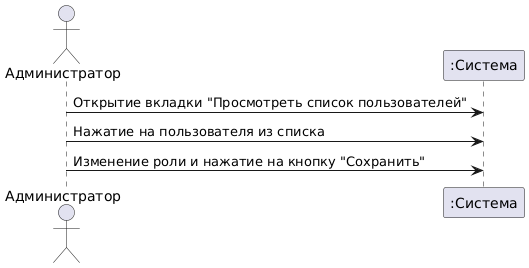


Рис.19. Диаграмма последовательности системы. Сменит﻿ь роль пользователю

Таблица 59

Описание операции «Сменит﻿ь роль пользователю»

|  |  |
| --- | --- |
| Раздел | Описание |
| Имя | Сменит﻿ь роль пользователю |
| Тип | Системная |
| Ссылки | Вариант использования «Сменит﻿ь роль пользователю» |
| Исключения | Отображение ошибки на веб-странице |
| Вывод | Смененная роль пользователя |

### 1.4.4. Построение диаграмм деятельностей вариантов использования

На этапе анализа требований и уточнения спецификаций диаграммы деятельностей позволяют конкретизировать основные функции разрабатываемого программного обеспечения. Под деятельность в данном случае понимают задачу (операцию), которую необходимо выполнить вручную или с помощью средства автоматизации.

Для варианта использования была взята операция у Системы «Формирование отчета» была построена схема, которая показывает, как выполняются её функции (рис.20).

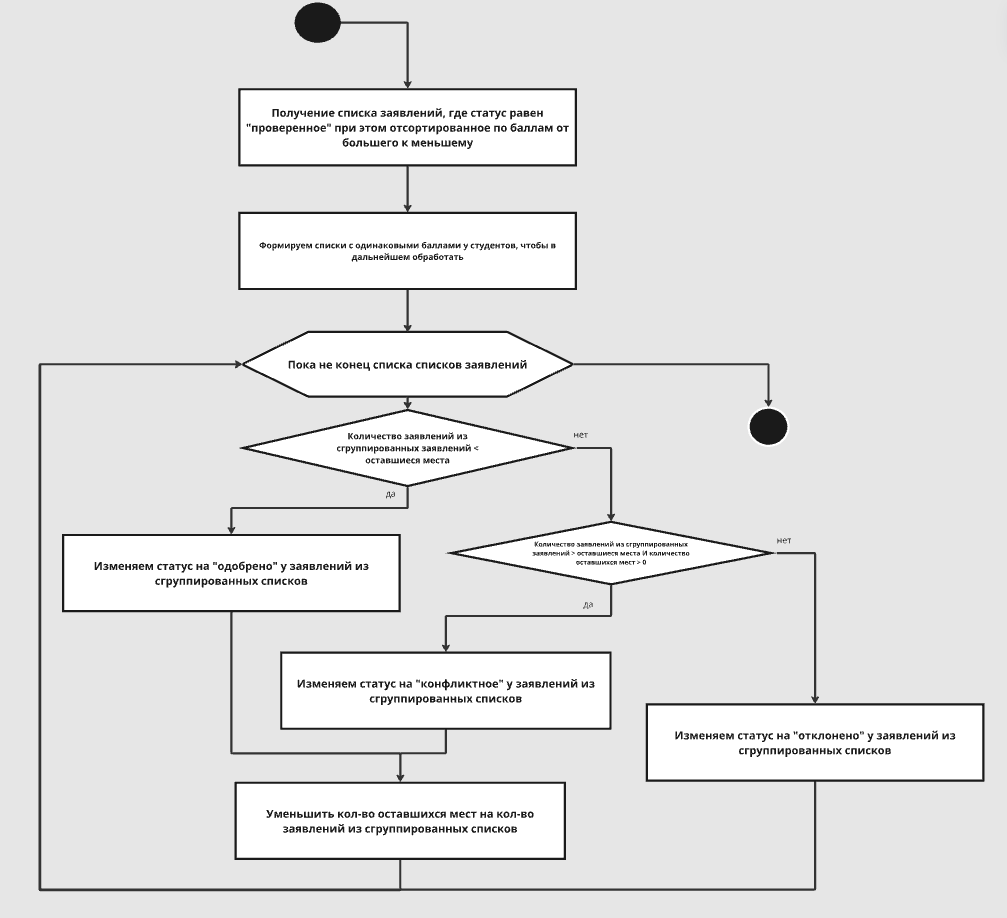


Рис.20. Диаграмма деятельности вариантов использования. Формирование отчета

Для того чтобы оптимизировать процесс формирования итогового отчета о студентах, получивших повышенную стипендию создана данная операция.

### 1.4.5. Построение диаграммы переходов состояний

Диаграмма переходов состояний является графической формой предоставления конечного автомата – математической абстракции, используемой для моделирования детерминированного поведения технических объектов или объектов реального мира. На этапе анализа требований и определения спецификаций диаграмма переходов состояний демонстрирует поведение разрабатываемой программной системы при получении управляющих воздействий. Под управляющими воздействиями или сигналами в данном случае понимают управляющую информацию, получаемую системой извне.

Чтобы показать, как разрабатываемая система переходит между состояниями при получении внешних сигналов была построена диаграмма переходов состояний (рис.21).

В диаграмме показано, что после запуска веб-приложения пользователь после авторизации может перейти на разные вкладки в зависимости от его роли. Студент – может подать заявление и просмотреть список заявлений, если понадобится может поменять что-то в заявлении или добавить. Проверяющий – может просмотреть списки заявлений со статусом "Проверенное","На проверке","Ошибочное", а также просмотреть конкретное заявление, он может его оценить либо просто выйти обратно в список заявлений. Жюри – может просмотреть списки заявлений со статусом "Отклонён﻿но﻿е" и "Конфликтное", "Одобренное", а также просмотреть конкретное заявление, он может его одобрить либо отклонить или просто выйти обратно в список заявлений. Администратор – может просмотреть список пользователей, так и заявлений, также может просматривать заявления и пользователей, у пользователей может менять роли, у заявления же статус, в его возможностях также указание дат начала и конца сбора заявлений, на случай проверки корректности сайта есть возможность просмотра логов и их очистка, а помимо этого просмотр устаревших заявлений и перевод заявлений в статус «Устаревшие».

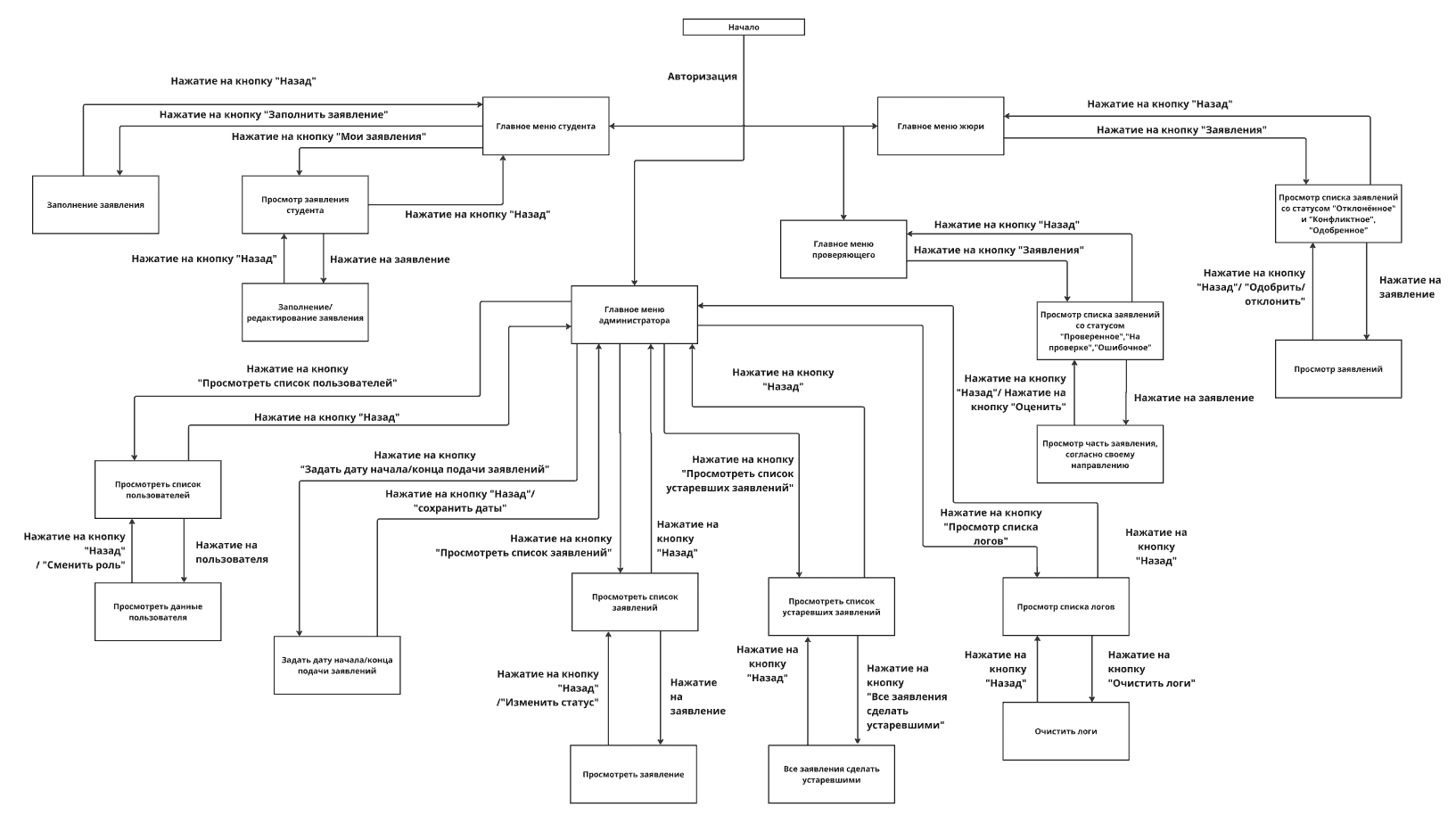


Рис.21. Диаграмма переходов состояний

### 1.5. Проектирование системы

1.5.1. Проектирование структуры системы и построение диаграмм пакетов

Важной задачей систематизации информации о предметной области является разбиение большой системы на небольшие подсистемы. Именно здесь особенно заметны структурные и объектно-ориентированные различия между подходами. Одна из идей заключается в группировке классов в компоненты более высокого уровня. В UML такой механизм группировки носит название пакетов. Диаграммой пакетов является диаграмма, содержащая пакеты классов и зависимости между ними. Строго говоря, пакеты являются элементами диаграммы классов, то есть диаграмма пакетов – это всего лишь диаграмма классов. Отличаются эти диаграммы практическим назначением и использованием.

Для разрабатываемого веб-приложения классы были сгруппированы в пакеты (рис.22).

Для описания пакетов была построена табл. 60.

Таблица 60

Описание диаграммы пакетов

|  |  |
| --- | --- |
| Название пакета | Описание |
| Static | Пакет, включающий классы для работы со статичными элементами веб-интерфейса, которые могут быть кэшированы (относится к frontend) |
| Templates | Пакет, включающий классы для работы со страницами и элементами веб-интерфейса (относится к frontend) |
| Services | Пакет, включающий классы для выполнения функций обработки информации, переданной пользователем через Presentation Layer (относится к backend) |
| Models | Пакет, включающий классы работы с данными в базе данных (относится к backend) |

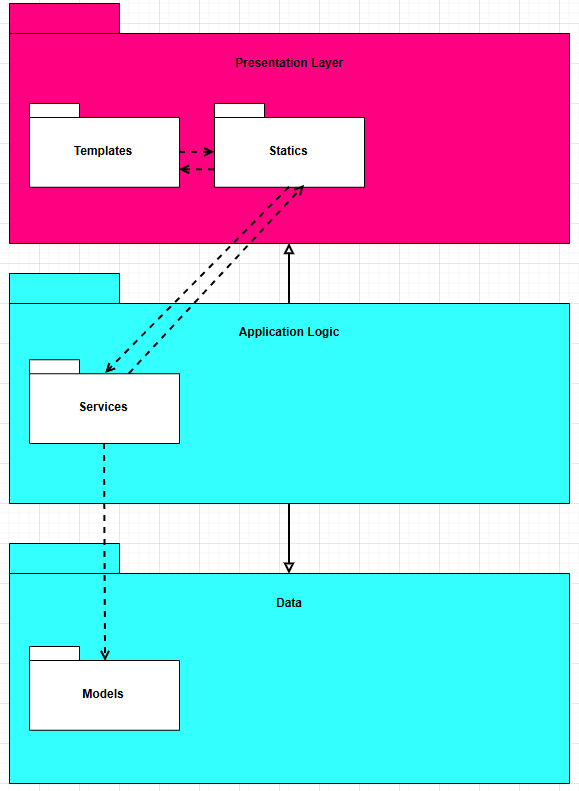


Рис.22. Диаграмма пакетов

### 1.5.2. Проектирование классов в пакетах

### 1.5.2.1. Проектирование классов пакета «Models»

### 1.5.2.1.1. Построение исходной диаграммы классов пакета «Models»

Диаграмма классов представляет описание структуры классов в системе и их взаимосвязи. Она отображает как статические аспекты системы, включая классы, атрибуты и методы, а также динамические аспекты, такие как связи между объектами и выполнение методов во время выполнения программы.

Связи между классами на диаграмме классов могут представлять отношения, такие как ассоциация, наследование и реализацию. Ассоциация показывает отношения между классами, которые могут быть однонаправленными или двунаправленными. Наследование показывает, как класс наследует свойства другого класса, который является его родительским классом.

Для пакета «Model» была построена исходная диаграмма классов, чтобы описать его структуру (рис.23).

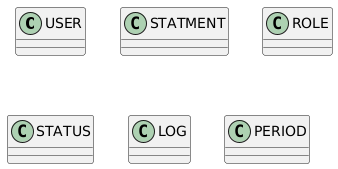


Рис.23. Исходная диаграмма классов пакета «Model»

Для описания классов была построена табл. 60.

Таблица 61

Описание исходной диаграммы классов пакета «Model»

|  |  |
| --- | --- |
| Название класса | Описание |
| USER | Класс, представляющий пользователя системы, включая учетные данные для входа. |
| STATMENT | Класс, представляющий заявление в системе включая данные заявления и информационные данные заявления. |
| ROLE | Справочник значений, позволяющий хранить информацию о ролях пользователей в системе. |
| STATUS | Справочник значений, позволяющий хранить информацию о статусах заявлений |
| LOG | Класс, представляющий информацию о совершенных изменениях в системе, отображенных в хронологический порядок |
| PERIOD | Справочник значений, позволяющий хранить информацию о сроках подачи заявления начала подачи и конца. |

### 1.5.2.1.2. Построение диаграммы последовательности взаимодействия объектов классов пакета «Model»

Диаграммы последовательностей, обычно используемые разработчиками, моделируют взаимодействия между объектами в едином сценарии использования. Они иллюстрируют, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования.

Чтобы показать последовательность выполнения сценария работы пакета «Model» была построена диаграмма последовательности действий, для сценария загрузки данных заявления от студента (рис.24-25).

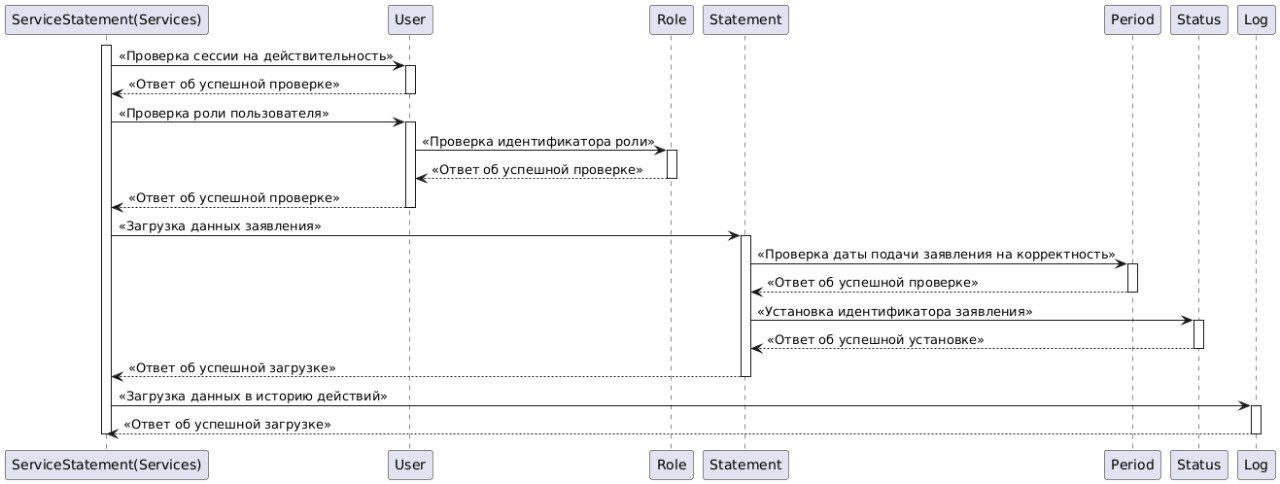


Рис.24. Диаграмма последовательности действий пакета «Model» для сценария загрузки данных заявления от студента

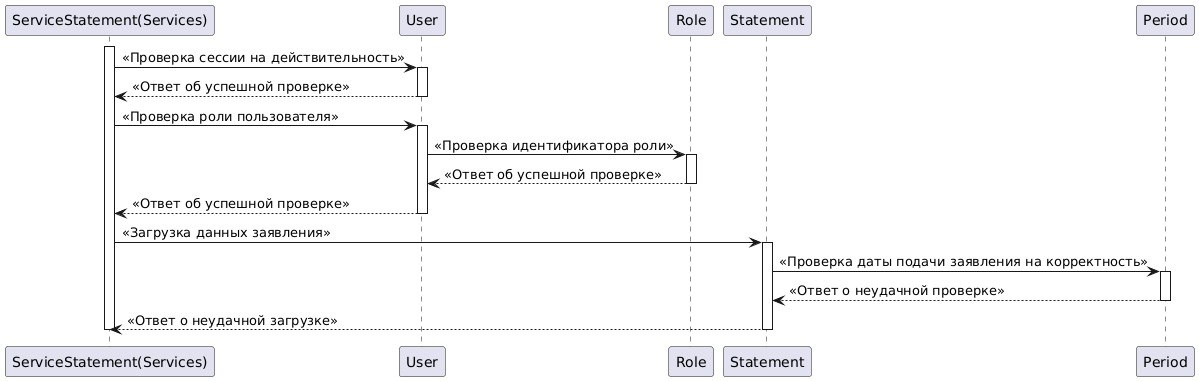


Рис.25. Диаграмма последовательности действий пакета «Model» для сценария загрузки данных заявления от студента с ошибкой

Прерывания оператором не предусмотрены для данного пакета, поэтому соответствующая диаграмма будет отсутствовать.

Диаграмма последовательности действий для пакета Model объединяет основные сценарии работы системы, включая создание загрузку заявления студентом, оценивание части заявления инспектором (проверяющим), утверждение заявления на повышенную стипендию, автоматическое распределение на утвержденные, конфликтные и отклоненные, а также присвоение статуса администратором, что заявление устаревшее. Она охватывает весь цикл взаимодействия между ключевыми классами пакета и базой данных, а также учитывает обработку ошибок и завершение объектов, где это необходимо.

Описание сценария:

Загрузка данных заявления: студент инициирует процесс загрузки заявления, изначально проверяется сессия пользователя на действительность, далее идет проверка роли пользователя, если предыдущие этапы успешны, то начинается процесс, который обрабатывается классом Statement. Идет проверка сроков для подачи заявления, если текущая дата попадает в заданный Администратором диапазон, то выполняется загрузка заявления, у которого статус становится 0. После загрузки заявления выполняется запись в историю действий.

### 1.5.2.1.3. Построение диаграммы кооперации пакета «Models»

Диаграмма кооперации — это альтернативный способ представления взаимодействия объектов в процессе реализации сценария, который позволяет по-другому взглянуть на ту же информацию. В отличие от диаграмм последовательностей диаграммы кооперации показывают потоки данных между объектами классов, что позволяет уточнить связи между ними.

В рамках проектирования пакета «Models» будет построенная диаграмма размещения.

### 1.5.2.1.4. Построение уточнённой диаграммы классов пакета «Models»

Уточненная диаграмма классов необходима для того, чтобы показать уточнения отношений между ними.

Для пакета «Models» была построена данная диаграмма, на которой видно, как классы взаимодействуют друг с другом (рис.26).

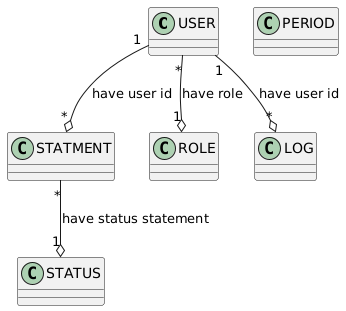


Рис.26. Уточненная диаграмма классов пакета «Models»

На каждой уточненной диаграмме показано, какие отношения между собой имеют все классы пакета.

## 1.5.2.1.5. Построение детальной диаграммы классов пакета «Models»

Для того чтобы показать все функции и переменные, которые используются в классах пакета «Models», была спроектирована детальная диаграмма классов (рис.27).

Для данных классов описание полей соответствует описанию полей в физической модели данных пункта 1.4.5.



Рис. 27. Детальная диаграмма классов пакета «Data»

Таблица 62. Описание полей класса STATMENT

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | String | Уникальный идентификатор заявления |
| USER | Int | Пользователь отправивший заявление |
| STATUS | Int | Статус заявления |
|  |  |  |
| DATE | Date | Дата последнего изменения заявления |
| JSON | JSON | Данные заявления |
| MARK\_STUDIES | Int | Оценка части заявления «Обучение» |
|  |  |  |
| COMMENT\_STUDIES | String | Комментарий к части заявления «Обучение» |
| MARK\_SCIENCE | Int | Оценка части заявления «Наука» |
| COMMENT\_SCIENCE | String |  |
| Комментарий к части заявления «Наука» |
| MARK\_ACTIVITIES | Int | Оценка части заявления «Общественная деятельность» |
| COMMENT\_ACTIVITIES | String | Комментарий к части заявления «Общественная деятельность» |
| MARK\_CULTURE | Int | Оценка части заявления «Культура и творчество» |
| COMMENT\_CULTURE | String | Комментарий к части заявления «Культура и творчество» |
| MARK\_SPORT | Int | Оценка части заявления «Спорт» |
| COMMENT\_SPORT | String | Комментарий к части заявления «Спорт» |
| POINTS | Int | Итоговые баллы |
| CACHE\_STATUS | Bool | Флаг кэша заявлений |

Таблица 63. Описание методов класса STATMENT

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Возвр. знач. | Описание |
| get\_data | self:Statment | Object | Получить информацию о заявлении |
| get\_json\_data | self:Statment | Object | Получить информация заявления |
| mark | user:Int, value:Int,comment:String | void | Оценить часть заявления |
| set\_mark\_studies | value:Int,comment:String | void | Оценить конкретную часть заявления «Обучение» |
| set\_mark\_science | value:Int,comment:String | void | Оценить конкретную часть заявления «Наука» |
| set\_mark\_activities | value:Int,comment:String | void | Оценить конкретную часть заявления «Общественная деятельность» |
| set\_mark\_culture | value:Int,comment:String | void | Оценить конкретную часть заявления «Культура и творчество» |
| set\_mark\_sport | value:Int,comment:String | void | Оценить конкретную часть заявления «Спорт» |
| check\_all\_marks | self:Statment | void | Проверка всех оценок выставлены ли они |
| set\_status\_up | self:Statment | void | Поставить заявление на статус выше |
| set\_status\_down | self:Statment | void | Поставить заявление на статус ниже |
| remove\_files | self:Statment | void | Удалить приложенные файлы заявления |
| upload | user:User,json:Object | Bool | Загрузить заявление |
| system\_checkout | counts:Int | void | Распределить заявления на одобренные и отклоненные автоматически |
| get\_by\_user | user:User | Object | Получить заявление по пользователю |
| get\_by\_id | statement\_id:String | Object | Получить заявление по id |
| get\_statements\_by\_statuses | statuses:String[] | Object | Получить заявление по статусу |
| get\_statements\_cache | - | Object | Получить заявления, которые закешированы |
| get\_part\_statement | user:User, statement\_id:String | Object | Получить часть заявления |
| generate\_id | - | String | Сгенерировать id |
| cache | - | void | Поставить флаг кэша в значение true |

Таблица 64. Описание полей класса LOG

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | Int | Уникальный идентификатор записи в логах |
| EVENT | String | Строка, описывающая события |
| STATMENT\_COPY | JSON | Копия json объекта заявления |
| DATA | String | Комментарий к событию |
| USER | Int | Пользователь, совершивший события |
| DATE | Date | Дата совершения события |

Таблица 65. Описание методов класса LOG

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Возвр. знач. | Описание |
| get\_data | self:Log | Object | Получить данные записи в истории действий |
| add | user:User, event:String, data:String, json:Object | void | Добавить запись в историю действий |
| get\_list | - | String[] | Получить список истории действий |
| reset | - | void | Очистить все записи в истории действий |

Таблица 66. Описание полей класса PERIOD

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | Int | Уникальный идентификатор интервала подачи заявления |
| DATE | String | Дата интервала подачи заявления |
| DESCRIPTION | String | Описание интервала подачи заявления |

Таблица 67. Описание методов класса PERIOD

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Возвр. знач. | Описание |
| get\_period | - | String | Получить интервал подачи заявления |
| is\_require | date:Date | Bool | Проверить подходит ли заданный интервал данная дата |
| set\_start\_and\_end | datestr\_start:String, datestr\_end:String | void | Задать дату начала и конца |
| create\_default\_records | class:Period | void | Создать записи по умолчанию |

Таблица 68. Описание полей класса ROLE

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | Int | Уникальный идентификатор роли |
| ROLE | String | Текстовое название роли |

Таблица 69. Описание методов класса ROLE

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Возвр. знач. | Описание |
| create\_default\_records | cls:Class | void | Создать записи по умолчанию |

Таблица 70. Описание полей класса STATUS

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | Int | Уникальный идентификатор статуса |
| STATUS | String | Текстовое название статуса |

Таблица 71. Описание методов класса STATUS

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Возвр. знач. | Описание |
| create\_default\_records | cls:Class | void | Создать записи по умолчанию |

Таблица 72. Описание полей класса USER

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | String | Уникальный идентификатор пользователя |
| SESSION | String | Уникальный идентификатор сессии |
| LOGIN | String | Логин пользователя |
| ROLE | Int | Роль пользователя |
| SURNAME | String | Фамилия пользователя |
| NAME | String | Имя пользователя |
|  |  |  |
| MIDDLENAME | String | Отчество пользователя |
| EMAIL | String | Почта пользователя |
| PHONE | String | Номер телефона пользователя |
| GROUP | String | Группа пользователя |

### 1.5.3. Построение диаграмм компонентов

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.

Компонентами могут быть программные компоненты, такие как база данных или пользовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройство; или бизнес-подразделение, такое как поставщик, платежная ведомость или доставка.

Для того, чтобы показать представление организации системы между различными видами компонентов были построены диаграммы компонентов и модульная структура (рис.28-39).

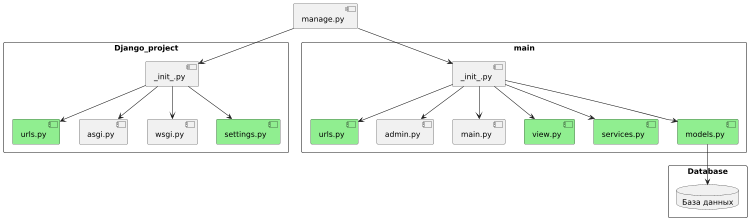


Рис.28. Диаграмма компонентов

Назначение компонентов описано в табл. 81.

Таблица 81. Описание компонентов

Описание компонентов

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Назначение | Входные данные | Выходные данные |
| View.py | Предоставляет пользователю возможность оперировать системой | Действия пользователя | Вывод данных в браузер |
| База данных | Хранение данных | Данные пользователя | Информация из БД |
| Services.py | Обеспечивает обработку информации, переданную пользователем, и запросов | Данные пользователя | Отсутствуют |
| Models.py | Описывает таблицы в базе данных, которые передает веб-приложение для хранения информации | Созданные классы в пакете «Models» | Таблицы в базе данных |

В компонентах settings.py и urls.py были расписаны настройки приложения, адреса и порты приложения.

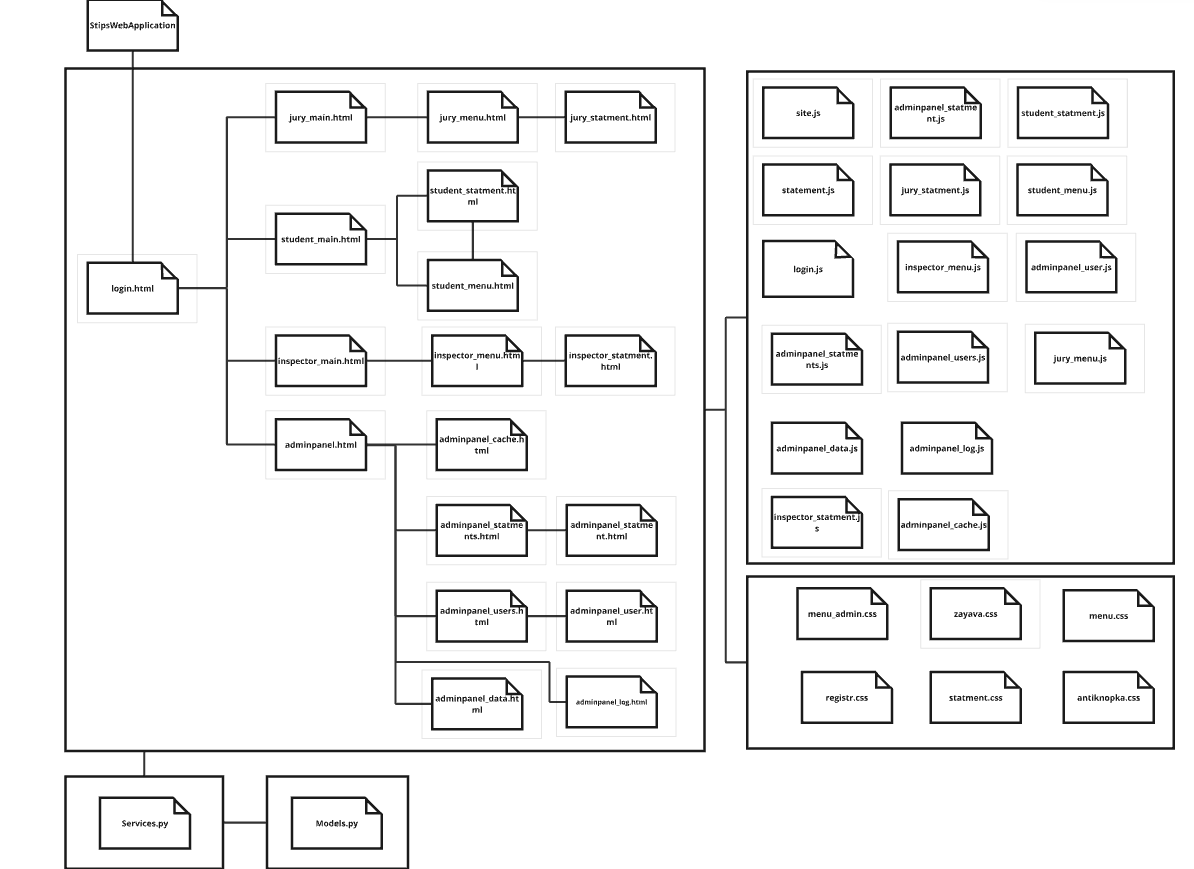


Рис.29. Модульная структура

Описание модулей программы представлено в табл. 82.

Таблица 82. Описание модулей программы

Описание модулей

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Назначение | Входные данные | Выходные данные |
| 1 | 2 | 3 | 4 |
| StipsWebA﻿pplication | Инициализация и запуск приложения | Конфигурация приложения | Запуск компонентов приложения |
| login.html | Страница входа | - | HTML-страница |
| jury\_m﻿ain.html | Главная страница жюри | Сессия | HTML-страница |
| 1 | 2 | 3 | 4 |
| student\_﻿main.html | Главная страница студента | Сессия | HTML-страница |
| inspector﻿\_main.html | Главная страница проверяющего | Сессия | HTML-страница |
| adminpanel.html | Главная страница админа | Сессия | HTML-страница |
| jury\_me﻿nu.html | Страница просмотра списка заявлений | Сессия | HTML-страница |
| jury\_statment.html | Страница подробного просмотра заявления | Сессия, id stament | HTML-страница |
| student\_statment.html | Страница подробного просмотра заявления | Сессия, id stament | HTML-страница |
| student\_﻿menu.html | Страница просмотра списка заявлений | Сессия | HTML-страница |
| inspector\_menu.htm﻿l | Страница просмотра списка заявлений | Сессия | HTML-страница |
| inspector\_s﻿tatment.html | Страница подробного просмотра заявления | Сессия, id stament | HTML-страница |
| adminpanel\_cache.htm﻿l | Страница для просмотра всех устаревших заявлений и также с возможностью сделать их устаревшими | Сессия | HTML-страница |
| adminpanel\_statments.﻿html | Страница просмотра списка заявлений | Сессия | HTML-страница |
| adminpanel\_statment﻿.html | Страница подробного просмотра заявления | Сессия, id stament | HTML-страница |
| adminpanel﻿\_users.html | Страница просмотра списка пользователей | Сессия | HTML-страница |
| adminpanel\_user.html | Страница подробного просмотра данных пользователя | Сессия, id stament | HTML-страница |
| 1 | 2 | 3 | 4 |
| adminpanel\_data.html | Страница для указания дат подачи заявлений | Сессия | HTML-страница |
| adminp﻿anel\_﻿log.html | Страница для просмотра истории действий на сайте и также с возможностью очистить историю | Сессия | HTML-страница |
| menu\_admin﻿.css | Стили для страницы входа | HTML-структура | Стили отображения |
| zayava.css | Стили для главной страницы | HTML-структура | Стили отображения |
| menu.c﻿ss | Стили для страницы пользователей | HTML-структура | Стили отображения |
| registr.css | Стили для страницы ролей | HTML-структура | Стили отображения |
| statment.css | Стили для страницы заявлений | HTML-структура | Стили отображения |
| antikn﻿opka.css | Стили для страницы справочников | HTML-структура | Стили отображения |
| site.js | Скрипты для всех странц | HTML-структура | Динамическое поведение страницы |
| statement.js | Скрипты для заявлений | HTML-структура | Динамическое поведение страницы |
| log﻿in.js | Скрипт для авторизации | HTML-структура | Динамическое поведение страницы |
| adminpanel\_statments.js | Скрипт для заявлений администратора | HTML-структура | Динамическое поведение страницы |
| 1 | 2 | 3 | 4 |
| adminpanel\_﻿data.js | Скрипт для задачи дат подачи заявлений | HTML-структура | Динамическое поведение страницы |
| inspector\_statment.js | Скрипт для заявлений инспектора | HTML-структура | Динамическое поведение страницы |
| adminpanel\_statment.js | Скрипт для заявлений администратора | HTML-структура | Динамическое поведение страницы |
| adminp﻿anel\_log.js | Скрипт для истории действий | HTML-структура | Динамическое поведение страницы |
| admin﻿panel\_cache.js | Скрипт для устаревших заявлений | HTML-структура | Динамическое поведение страницы |
| student\_statment.js | Скрипт для заявленя студента | HTML-структура | Динамическое поведение страницы |
| student\_menu.js | Скрипт для списка заявлений студента | HTML-структура | Динамическое поведение страницы |
| inspector\_menu.js | Скрипт для списка заявлений проверяющего | HTML-структура | Динамическое поведение страницы |
| adminpanel\_users.js | Скрипт для списка пользователей | HTML-структура | Динамическое поведение страницы |
| jury\_statment.js | Скрипт для заявлений жюри | HTML-структура | Динамическое поведение страницы |
| jury\_menu.js | Скрипт для списка заявлений жюри | HTML-структура | Динамическое поведение страницы |
| 1 | 2 | 3 | 4 |
| adminpane﻿l\_user.js | Скрипт для просмотра пользоваля | HTML-структура | Динамическое поведение страницы |
| services.py | Сервисы для обработки запросов | - | Методы обработки запросов |
| Models.py | Модели | - | Модели и методы их обработки |

Таблица 83

Спецификация компонентов Services.py

|  |  |
| --- | --- |
| Функция или свойство | Назначение |
| 1 | 2 |
| signIn | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью User, |
| getPeriod | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Period, возвращая данные о сроках подачи заявлений |
| getRole | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Role, возвращая роль пользователя, который отправил запрос |
| uploadSta﻿tement | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, создавая или изменяя данные заявления |
| getMyStatements | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, возвращая список заявлений принадлежащих пользователю |
| getMyStatemen﻿t | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, возвращая определенное заявление |
| getListStatementsInspector | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, возвращая список заявлений |
| 1 | 2 |
| rateStatement | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, устанавливая оценку определенного заявляения |
| getStatementInspect﻿or | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, возвращая данные определенного заявления |
| getListStatementsJury | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, возвращая список заявлений |
| getStatementJury | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, возвращая данные определенного заявления |
| confirmStatement | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement и Status, устанавливая статус «одобренное» определенного заявления |
| denyStatement | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement и Status, устанавливая статус «отклоненное» определенного заявления |
| **getListUsers** | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью User, возвращая список заявлений |
| getUser | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью User, возвращая данные определенного пользователя |
| changeRol﻿e | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью User и Role, изменяя роль у определенного пользователя |
| setPeriod | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Period, устанавливая сроки подачи заявлений |
| **getListStatements** | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, возвращая список заявлений |
| **getStatement** | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, возвращая данные определенного заявления |
| **1** | 2 |
| **changeStatus** | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделями Statement и Status, меняя у заявления статус |
| **setOld** | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, ставя флаг «устаревшее» определенному заявлению |
| **getStatementsOld** | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Statement, возвращая все записи, которые с флагом «устаревшее» |
| **getLo**﻿**g** | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Log, возвращая все записи |
| **resetLog** | Метод, который принимает запросы на адрес с соответственным названием и выполняется манипулирование моделью Log, удаляя все записи |

Таблица 84

Спецификация компонентов Models.py

|  |  |
| --- | --- |
| Функция или свойство | Назначение |
| User | Модель для манипулирования данными базы данных и связями между сущностями, отображая сущность пользователя |
| Statement | Модель для манипулирования данными базы данных и связями между сущностями, отображая сущность заявления |
| Status | Модель для манипулирования данными базы данных и связями между сущностями, отображая сущность статус заявления |
| Role | Модель для манипулирования данными базы данных и связями между сущностями, отображая сущность роль пользователя |
| Log | Модель для манипулирования данными базы данных и связями между сущностями, отображая сущность история действий |
| Period | Модель для манипулирования данными базы данных и связями между сущностями, отображая сущность сроки подачи заявлений |

### 1.5.4. Построение диаграмм размещения

Диаграммы размещения используются для визуализации аппаратных процессоров/узлов/устройств системы, каналов связи между ними и размещения программных файлов на этом аппаратном обеспечении.

Данная система программного обеспечения используется на нескольких компьютерах, расположенных в одной локальной сети. Поэтому было выполнено проектирование схемы размещения (рис.30).

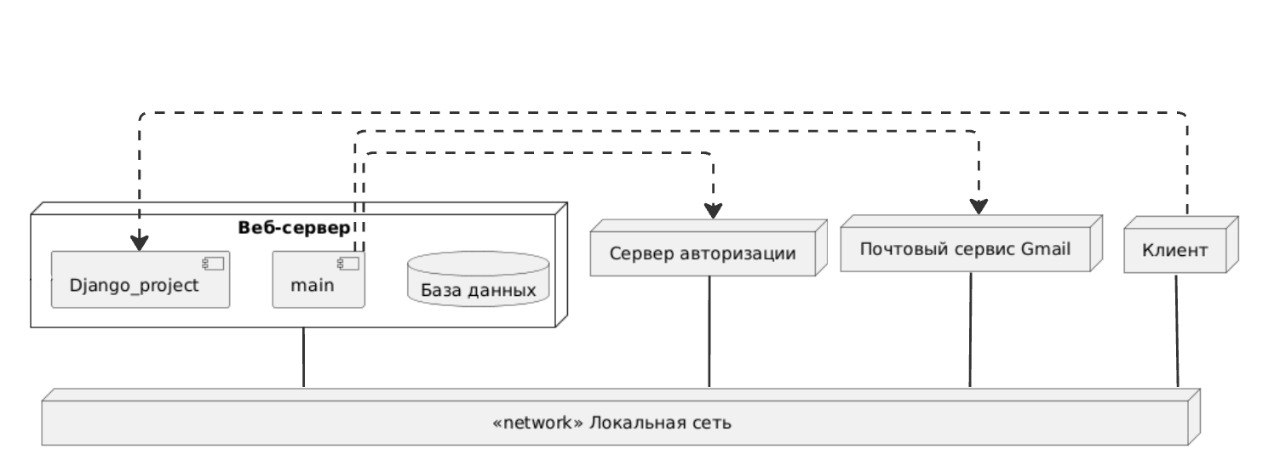


Рис.30. Диаграмма размещения

Данная диаграмма размещения показывает взаимодействие между клиентом, веб-сервером, сервером авторизации и почтовым сервисом Gmail. Пользователь работает с системой через веб-браузер, который отправляет HTTP/HTTPS-запросы на веб-сервер к компоненту «Django\_project».

На веб-сервере размещено веб-приложение, разработанное с использованием Django, обрабатывающее запросы пользователя и взаимодействующее с базой данных. База данных SQLite, содержит ключевые таблицы, такие как заявления, пользователи, роли и история действий.

Связь между веб-приложением и серверами авторизации и почтовым сервисом осуществляется через написанное приложение main.

### 1.6. Результаты проведения испытаний

Для тестирования была составлена специальная таблица, в которой показаны результаты испытаний (табл.85).

Таблица 85

Функциональное тестирование backend

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата тестирования | Тестируемое требование | Кто проводил тестирование | Способ тестирования | Результат тестирования |
| 1 | 2 | 3 | 4 | 5 |
| 23.11.2024 | Проверка получения даты начала и конца сбора заявлений | Ульянов А.С. | Системное тестирование | Успех |
| 23.11.2024 | Авторизация | Ульянов А.С. | Системное тестирование | Успех |
| 25.11.2024 | Просмотреть список своих заявлений | Ульянов А.С. | Системное тестирование | Успех |
| 26.11.2024 | Просмотреть информацию о своем заявлений | Ульянов А.С. | Системное тестирование | Успех |
| 27.11.2024 | Отправка заявления на проверку | Ульянов А.С. | Системное тестирование | Ошибка |
| 27.11.2024 | Отправка заявления на проверку | Ульянов А.С. | Системное тестирование | Ошибка |
| 28.11.2024 | Отправка заявления на проверку | Ульянов А.С. | Системное тестирование | Успех |
| 28.11.2024 | Смена роли пользователю | Ульянов А.С. | Системное тестирование | Успех |
| 28.11.2024 | Оценить часть заявления | Ульянов А.С. | Системное тестирование | Успех |
| 29.11.2024 | Одобрить заявление | Ульянов А.С. | Системное тестирование | Успех |
| 29.11.2024 | Просмотреть историю действий | Ульянов А.С. | Системное тестирование | Успех |
| 03.12.2024 | Все заявления сделать устаревшими | Ульянов А.С. | Системное тестирование | Ошибка |
| 03.12.2024 | Все заявления сделать устаревшими | Ульянов А.С. | Системное тестирование | Успех |

Проведенные испытания подтвердили соответствие разработанного программного обеспечения заявленным требованиям. Тестирование функционала системы, настройку ролей, редактирование данных, показало стабильность работы, высокую производительность и соответствие спецификации.

Общая оценка испытаний позволяет сделать вывод о том, что система готова к внедрению с учетом внесения необходимых доработок и дальнейшего мониторинга ее работы в реальных условиях.

# Заключение

В рамках данной курсовой работы была разработана спецификация и архитектура серверной части программного обеспечения, предназначенного для автоматизации обработки заявлений на повышенную стипендию.

Основное внимание уделялось проектированию ключевых элементов архитектуры, выбору оптимальных технологий. На первоначальном этапе была проведена аналитическая работа, включающая исследование аналогичных решений и подбор наиболее подходящих технологий разработки. Этот анализ позволил определить основные требования к системе: удобство использования, гибкость настройки, интеграцию с существующими системами аутентификации и поддержку различных форматов данных. Для выполнения поставленных задач были выбраны современные инструменты, а для проектирования архитектуры использовался UML.

Особое место в работе заняло проектирование и реализация пакета Models, являющегося ключевым компонентом системы, отвечающим за управление данными. Были созданы модели, описывающие основные сущности предметной области, такие как пользователи, роли, их поля и права доступа. Для этих моделей разработаны диаграммы классов, уточнены структуры данных и проработаны сценарии взаимодействия, что обеспечило целостность и логическую согласованность архитектуры.

Разработка веб-приложения позволила оптимизировать процесс обработки заявлений, сократить временные затраты на ручную проверку и исключить использование устаревших инструментов, таких как Excel. Благодаря этому «Система управления стипендиями Череповецкого государственного университета» готова стать надежным инструментом, обеспечивающим удобное управление стипендиями и повышение эффективности работы всех участников процесса.

# Список литературы

1. Что такое UML и почему его понимают во всём мире[Электронный ресурс] - https://practicum.yandex.ru/blog/uml-diagrammy/. Дата обращения: 20.11.2024;
2. Методология объектно-ориентированного моделирования. ЯЗЫК UML [Электронный ресурс] - https://kpfu.ru/staff\_files/F\_968641481/Metodichka\_UML.pdf. Дата обращения: 20.11.2024;
3. Django [Электронный ресурс] - https://developer.mozilla.org/ru/docs/Learn\_web\_development/Extensions/Server-side/Django. Дата обращения: 20.11.2024;
4. Django css js html [Электронный ресурс] - https://qna.habr.com/q/235509.Дата обращения: 20.11.2024;
5. Иванова ГС \_Технология программирования 241c 2002г.Дата обращения: 20.11.2024;
6. Ершов, Виноградова и др. Методика и организация самостоятельной работы студентов 208с 2015г.Дата обращения: 20.11.2024;
7. Буч Г., Рамбо Д., Якобсон И. Б90 Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс. – 496 с.: ил.Дата обращения: 20.11.2024;
8. Django backend [Электронный ресурс] - https://habr.com/ru/articles/755898/.Дата обращения: 20.11.2024;
9. Django models [Электронный ресурс] - https://docs.djangoproject.com/en/5.1/topics/db/models/.Дата обращения: 20.11.2024;
10. Дакетт, Дж. HTML и CSS. Разработка и дизайн веб-сайтов / Дж. Дакетт. — М.: Вильямс, 2018. — 512 с.

### Приложение 1. Техническое задание

МИНОБРАНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт информационных технологий

наименование института (факультета)

Математическое и программное обеспечение ЭВМ

наименование кафедры

Модуль: Технология разработки программного обеспечения

УТВЕРЖДАЮ

Зав. Кафедрой МПО ЭВМ

д. т.н. \_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2025 г.

Проектирование программного обеспечения

Разработка программного обеспечения «Стипендии ЧГУ»: серверная часть.

Техническое задание к курсовому проекту

Листов 10

Руководитель: Варфоломеев И. А.

Исполнитель: студент гр.1ПИб-01-2оп-21

Ульянов Александр Сергеевич

2025 г.

Введение

В рамках курсового проекта планируется разработать программное обеспечение, которое оптимизирует процесс подачи и обработки заявлений на повышенную стипендию. Цель проекта — ускорить выполнение задач и сократить объем ручной работы, повышая общую эффективность и автоматизируя ключевые этапы процедуры.

1. Основание для разработки

Основанием для разработки является задание на курсовой проект бакалавра, выданное на кафедре МПО ЭВМ «Череповецкий Государственный Университет» по заказу УИТ ЧГУ.

Дата утверждения: 10.02.2025 года.

Наименование темы разработки: Разработка программного обеспечения «Стипендии ЧГУ»: серверная часть.

1. Назначение разработки

Разрабатываемое программное обеспечение предназначено для автоматизации процесса подачи и обработки заявлений на повышенную стипендию. Система позволит сократить время обработки заявок, минимизировать количество ошибок при ручной обработке и обеспечить упрощенное взаимодействие между студентами и администрацией учебного заведения. Автоматизация ключевых этапов подачи заявлений позволит улучшить контроль за процессом и повысить оперативность принятия решений.

1. Требования к программе
   1. Требования к функциональным характеристикам.

Программное обеспечение должно выполнять следующие функции описанные ниже.

1. Авторизация пользователя и применение роли: студент, проверяющий, жюри, администратор.
2. Предоставлять доступ для заполнения заявления, в котором необходимо заполнить данные по направлениям: общие сведение, спорт, учеба, культура и творчество, наука, общественная деятельность. Ниже приведены поля для каждого направления:

1. Общие сведения

* ФИО (студента)
* Группа
* Контактный телефон
* Электронная почта
* Направление подготовки
* Курс
* Уровень подготовки (бакалавриат/магистратура/специалитет)
* Средний балл за предыдущую сессию
* Дата подачи заявления

2. Поля для направления "Учеба":

* Название мероприятия
* Дата проведения
* Организатор
* Форма участия (очная/заочная)
* Вид мероприятия (олимпиада, конкурс и т. д.)
* Достижения (место, награда)
* Подтверждающий документ (возможность прикрепления файла)

3. Поля для направления "Наука":

* Наименование научного мероприятия
* Дата и место проведения
* Тип мероприятия (конференция, конкурс научных работ)
* Уровень (международный, всероссийский, региональный и т. д.)
* Наименование работы/проекта
* Результат участия (доклад, публикация, награда)
* Подтверждающий документ

4. Поля для направления "Общественная деятельность":

* Вид деятельности (организация мероприятий, участие в волонтерских акциях и т. д.)
* Название проекта/мероприятия
* Дата участия
* Роль в мероприятии (организатор, участник и т. д.)
* Результат (благодарственное письмо, сертификат, диплом)
* Подтверждающий документ

5. Поля для направления "Культура и творчество":

* Название творческого мероприятия
* Дата и место проведения
* Тип участия (индивидуальное/групповое)
* Достижение (награда, участие, выступление)
* Подтверждающий документ

6. Поля для направления "Спорт":

* Вид спорта
* Название спортивного мероприятия
* Дата и место проведения
* Достижение (место, награда, участие)
* Уровень соревнований (международный, всероссийский и т. д.)
* Подтверждающий документ

1. Заявление должно обладать одним из статусов: на проверке, ошибочно, проверено, одобрено, отклонено, конфликтное. Предусмотреть возможность изменять его в зависимости от этапа проверки.
2. В зависимости от роли пользователя, система должна предоставлять следующие функциональные возможности:

* Студент:
  + просмотр своих ранее поданных заявлений и отслеживание их статуса;
  + заполнить/редактировать (до определенной даты) заявление на повышенную стипендию с возможностью прикрепления файлов и заполнения полей с информацией о себе и пройденных мероприятиях.
* Проверяющий:
  + поставить оценку части заявления, выбранного из списка согласно своему направлению;
  + посмотреть список заявлений на повышенную стипендию со статусом «на проверке», «проверено», «ошибочно»;
  + посмотреть часть выбранного из списка заявления, согласно своему направлению.
* Жюри:
  + одобрить/отклонить заявление выбранного из списка;
  + просмотреть список заявлений на повышенную стипендию со статусом «проверено», «одобрено», «конфликтное»;
  + просмотреть заявление, выбранное из списка.
* Администратор:
* просмотреть список заявлений;
* просмотреть список устаревших заявлений;
* администрирование студентов (просмотреть список пользователей, просмотреть студента выбранного из списка);
* администрирование жюри (Снять/назначить на должность жюри, просмотреть список жюри, просмотреть жюри выбранного из списка);
* администрирование проверяющих (Снять/назначить на должность проверяющего, просмотреть список проверяющих, просмотреть проверяющего, выбранного из списка);
* возможность пометить все заявления как устаревшие;
* просмотр истории действий, а также возможность ее очистить;
* задать дату начала и конца подачи заявления на повышенную стипендию.

1. Предусмотреть возможность разграничения заявлений текущего периода от заявлений прошедших периодов.
2. Предусмотреть возможность сохранения истории действий для того, чтобы отслеживать через нее ошибочные действия для своевременного устранения.
3. Доступ к данным студентов будет осуществляться по логину и паролю через API предоставленные заказчиком.
4. Возможность отправки сообщений на электронную почту студенту при изменении статуса заявления, которое им подано.

Серверная часть программного обеспечения должна выполнять следующие функции:

* работа с базой данных:
  + чтение данных;
  + изменение данных;
  + удаление данных;
  + сохранение после каждой проведенной операции.
* делать запросы к API с данными студентов и работников ЧГУ и обрабатывать полученные ответы;
* делать запросы к Gmail API для отправки электронных писем;
* отвечать на запросы клиентов, в зависимости от их доступа, выполняя соответствующие функции, указанные в П.1 пункт 3.1.4;
* обрабатывать исключительные ситуации;
* при наступлении заданной администратором даты окончания подачи заявлений выполнить изменение статусов не устаревших заявлений, так чтобы разделить заявления на 3 группы по статусам:
  + отклоненное;
  + одобренное;
  + конфликтное.

3.2 Входные данные

* Данные для доступа к API c данными студентов и работников ЧГУ;
* Данные для доступа к Gmail API (Почта, пароль, ключ);
* Локальная база данных;
* Запросы клиентов.

3.3 Выходные данные

* Запросы к Gmail API;
* Ответы сервера клиентам на их запросы;
* Изменение локальной базы данных;
* Файлы, прикрепленные клиентами и отправленные через запросы.

3.4 Требования к надежности

Программное обеспечение должно обеспечить сохранность информации при наступлении следующих событий:

* Неверно заполненное заявление;
* Некорректно указанные данные в заявлении;
* Некоректные запросы к серверу;
* Ошибки в сессии клиента;
* Отсутствие необходимых данных в базе данных;
* Отсутствие связи с сервисами через API;
* Аварийное завершение работы компьютера, в данных ситуациях желательно иметь бесперебойный источник питания.

Для устранения и отслеживания непредвиденных ошибок предусмотренна история действий.

3.5 Условия эксплуатации

Для оптимальной работы с приложением должны быть соблюдены следующие условия:

1. Компьютер, предназначенный для работы в закрытом отапливаемом помещении при следующих условиях окружающей среды:

* температура окружающего воздуха от +10°C до +35°C;
* относительная влажность воздуха не более 80%;
* запыленность воздуха не более 0,75 мг/м³.

1. Для работы с веб-приложением необходим веб-браузер.

3.6 Требования к составу и параметрам технических средств

Программа должна корректно работать при соблюдении минимальных системных требований:

* процессор с тактовой частотой не менее 2.5 ГГц и 4 ядрами;
* свободное место на жестком диске: 150 МБ и более;
* оперативная память: не менее 8 Гб;
* наличие устройств ввода вывода: монитор, клавиатура, мышь.

3.7 Требования к информационной и программной совместимости

* Система должна быть совместима с Windows 10;
* Стабильный интернет;
* На системе должен быть установлен Python версии 3.12;
* У системы должен быть доступ к Gmail API;
* У системы должен быть доступ к API c данными студентов ЧГУ;
* Клиенты должны использовать браузеры Google Chrome, Opera и Mozilla FireFox;

4. Требование к программной документации

* 1. Содержание расчётно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку и приложения, такие как:

* Техническое задание
* Текст программы;
* Спецификации;
* Руководство пользователя.

4.2 Требования к оформлению

Требования к оформлению, установленные ЕСПД ГОСТ и пособии по самостоятельной работе, должны быть выполнены на протяжении всей работы без каких-либо изменений (табл. П1.2).[6]

Таблица П1.2.

Требования к оформлению

|  |  |
| --- | --- |
| Документ | Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код. Листы и диск в конверте вложены в пластиковую папку скоросшивателя. |
| Страницы | Ориентация – книжная; отдельные страницы, при необходимости, альбомная.  Поля: верхнее, нижнее – по 2 см, левое – 3 см, правое – 1 см. |
| Абзацы | Межстрочный интервал – 1,5, перед и после абзаца – 0. |
| Шрифты | Кегль – 14. В таблицах шрифт 11. Шрифт листинга – 8 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: Рис.Х. Название  В приложениях: Рис.П1.3. Название |
| Таблицы | Подписывается: над таблицей, выравнивание по правому: «Таблица Х».  В следующей строке по центру Название Надписи в «шапке» (имена столбцов, полей) – по центру. В теле таблицы (записи) текстовые значения – выровнены по левому краю, числа, даты – по-правому. |

5. Стадии и этапы разработки

В данном пункте продемонстрированы стадии и этапы разработки, которые должны быть пройдены во время работы с программой (табл. П1.3).

Таблица П1.3.

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результаты  выполнения | Отметки о  выполнении |
| Разработка ТЗ | 10.02.25 | Оформленное ТЗ |  |
| Сравнительный анализ зарубежных аналогов | 22.02.25 | Проведен анализ аналогов, выявлены их недостатки относительно разрабатываемого ПО |  |
| Выбор технологии, среды и языка программирования | 07.03.25 | Выбраны технологии, среда и язык программирования |  |
| Анализ процесса обработки информации, выбор структур данных для её хранения | 24.03.25 | Выбраны структуры данных для хранения информации |  |
| Разработка спецификаций проектируемой системы | 10.04.25 | Спроектированы спецификации системы |  |
| Проектирование системы | 20.04.25 | Спроектирована структура системы |  |
| Проектирование интерфейса пользователя | 15.05.25 | Спроектирован интерфейс пользователя |  |
| Выбор стратегии тестирования, разработка тестов программа и методика испытаний | 20.05.25 | Выбрана стратегия тестирования, составлены тесты, выполнено тестирование |  |
| Технико-экономическое обоснование выполняемой разработки | 26.05.25 | Сформулировано технико-экономическое обоснование разработки |  |
| Оформление РПЗ | 01.06.25 | Оформлено РПЗ |  |
| Подготовка презентации | 01.06.25 | Подготовлена презентация |  |

6. Порядок контроля и приемки

Порядок контроля и приемки описан в табл. П1.4.

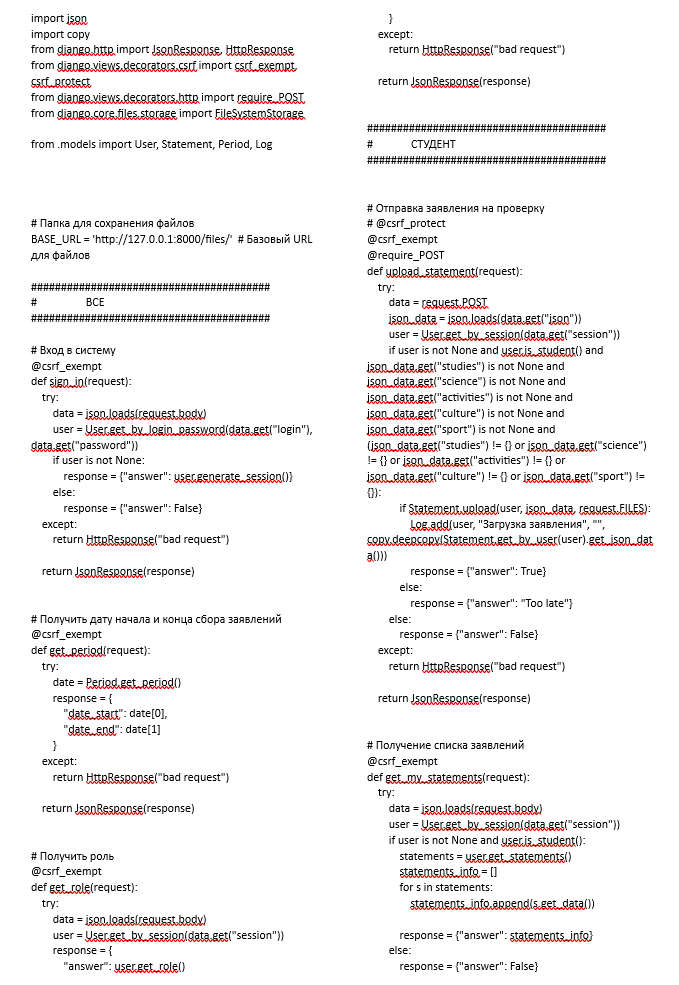
Таблица П1.4.

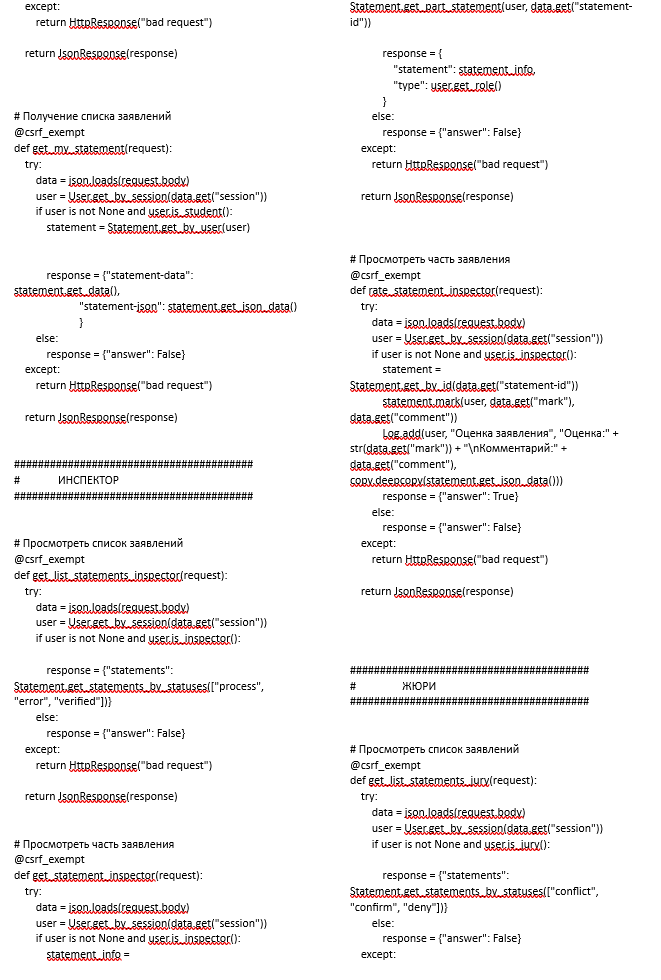
Порядок контроля и приемки

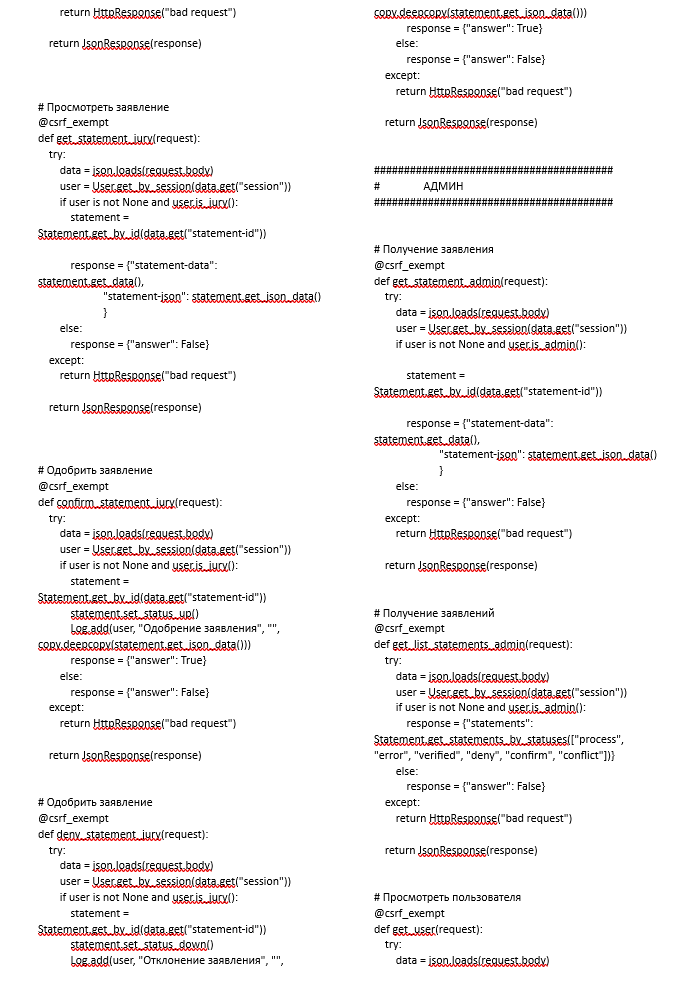
|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения выпускной квалификационной работы | Сроки разработки | Результаты  выполнения | Отметка о приемке результата контрольного этапа |
| Разработка ТЗ | 10.02.25 | Оформленное ТЗ |  |
| Сравнительный анализ зарубежных аналогов | 22.02.25 | Проведен анализ аналогов, выявлены их недостатки относительно разрабатываемого ПО |  |
| Выбор технологии, среды и языка программирования | 07.03.25 | Выбраны технологии, среда и язык программирования |  |
| Анализ процесса обработки информации, выбор структур данных для её хранения | 24.03.25 | Выбраны структуры данных для хранения информации |  |
| Разработка спецификаций проектируемой системы | 10.04.25 | Спроектированы спецификации системы |  |
| Проектирование системы | 20.04.25 | Спроектирована структура системы |  |
| Проектирование интерфейса пользователя | 15.05.25 | Спроектирован интерфейс пользователя |  |
| Выбор стратегии тестирования, разработка тестов программа и методика испытаний | 20.05.25 | Выбрана стратегия тестирования, составлены тесты, выполнено тестирование |  |
| Технико-экономическое обоснование выполняемой разработки | 26.05.25 | Сформулировано технико-экономическое обоснование разработки |  |
| Сдача РПЗ | 01.06.25 | Сдана РПЗ |  |
| Нормоконтроль документ | 01.06.25 | Пройдена проверка? |  |
| Предзащита КП | 10.06.25 | Пройдена предзащита КП |  |
| Защита КП | 25.06.25 | Защищена КП |  |

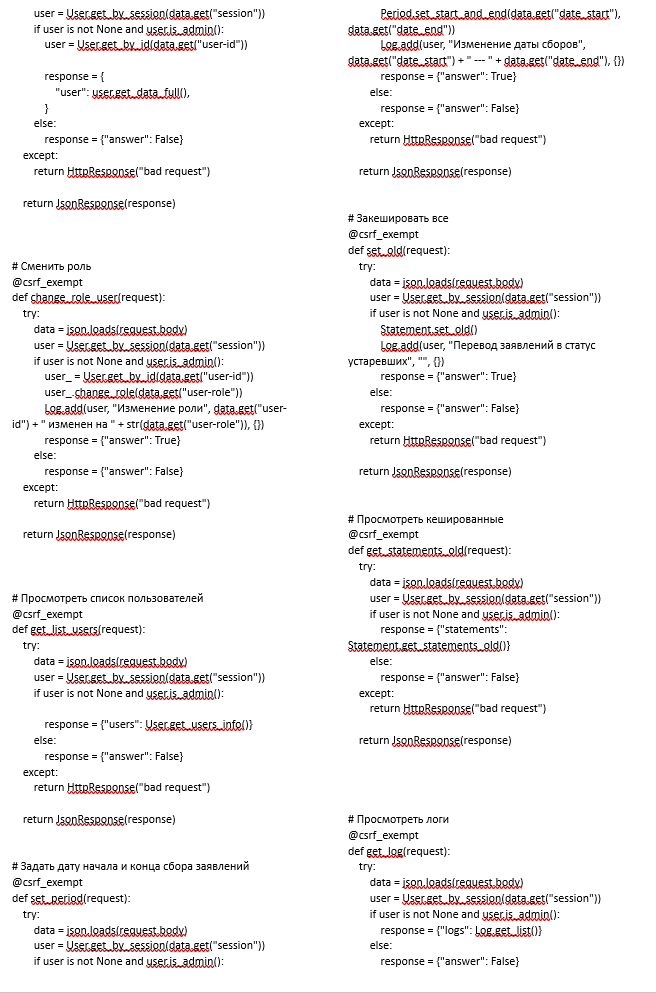
### Приложение 2. Текст программы

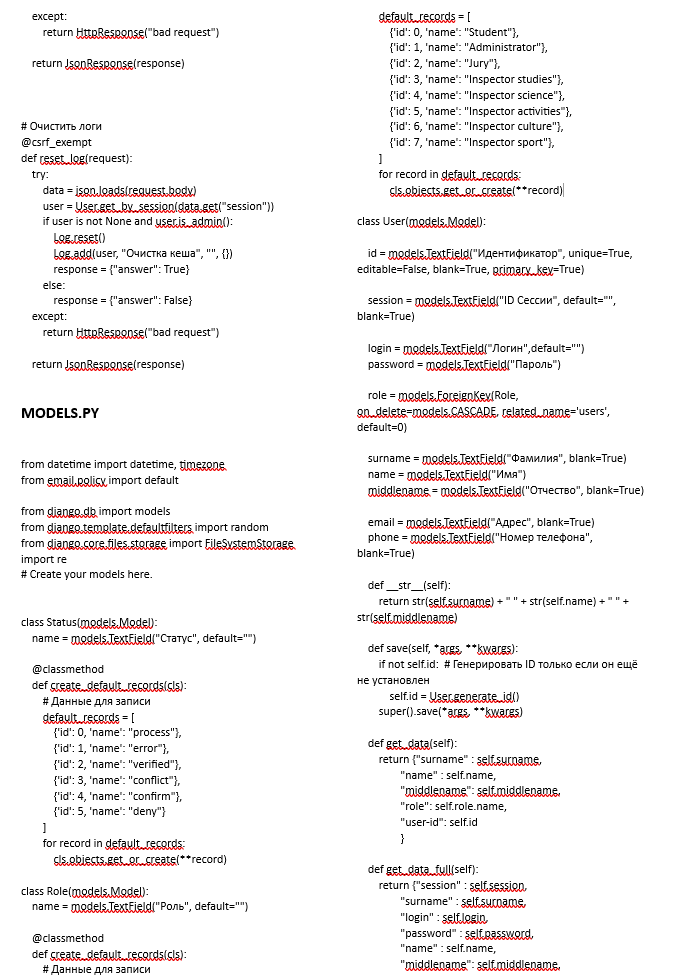
**services.py**

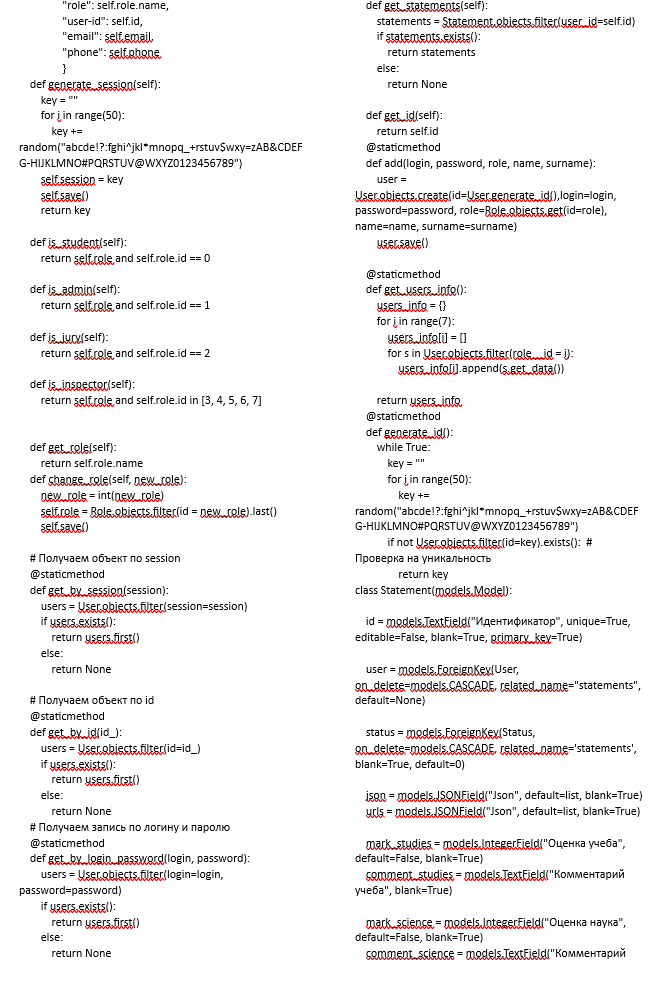
 Рис.П2.1. Код

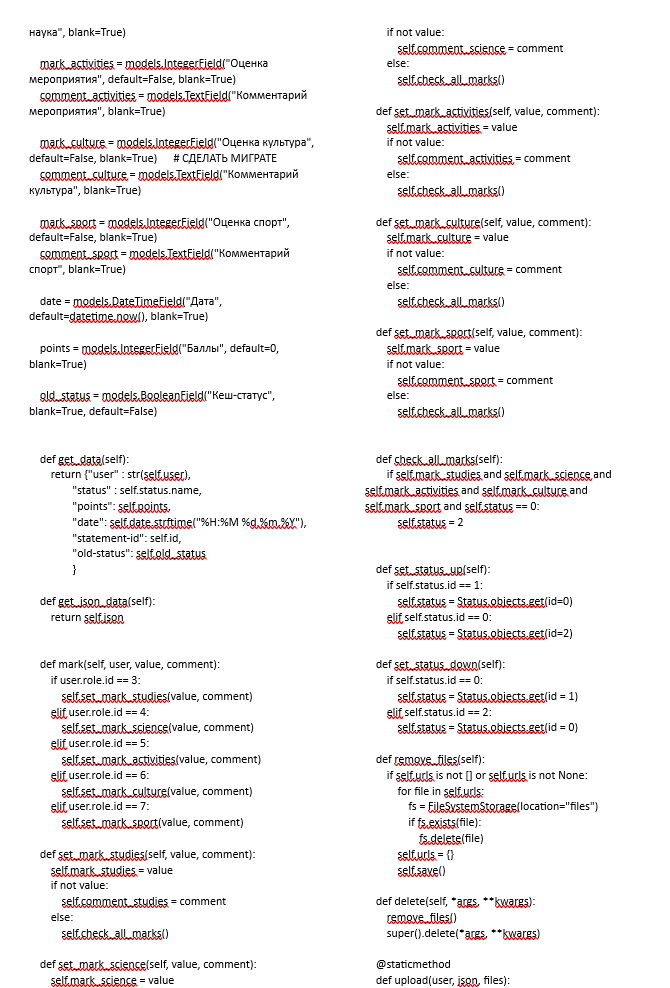
 Рис.П2.2. Код

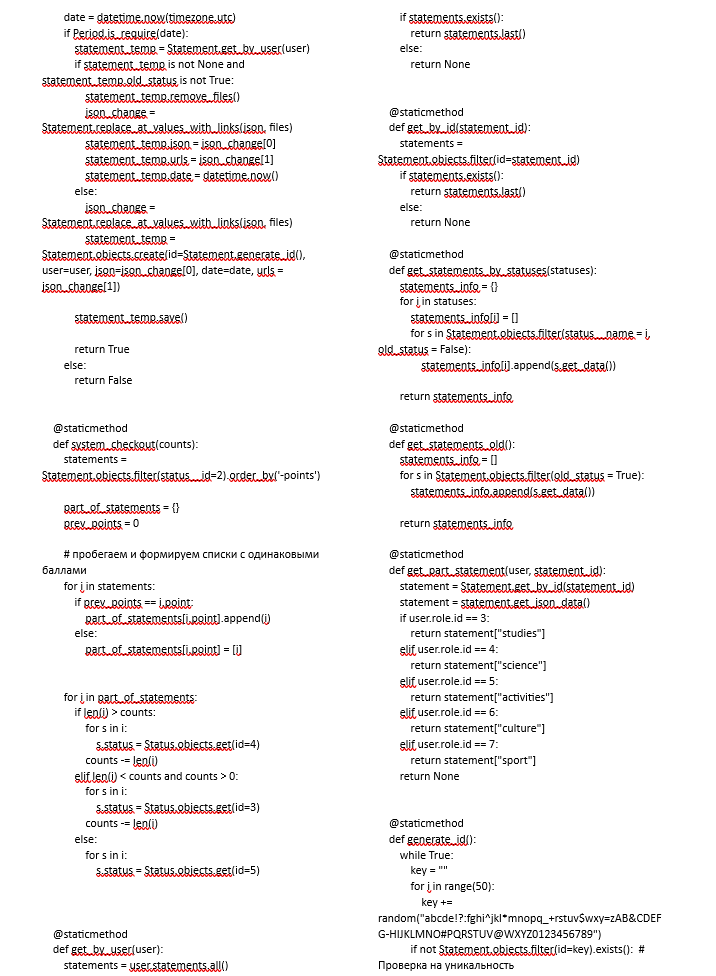
 Рис.П2.3. Код

 Рис.П2.4. Код

 Рис.П2.5. Код

 Рис.П2.6. Код

 Рис.П2.7. Код

 Рис.П2.8. Код

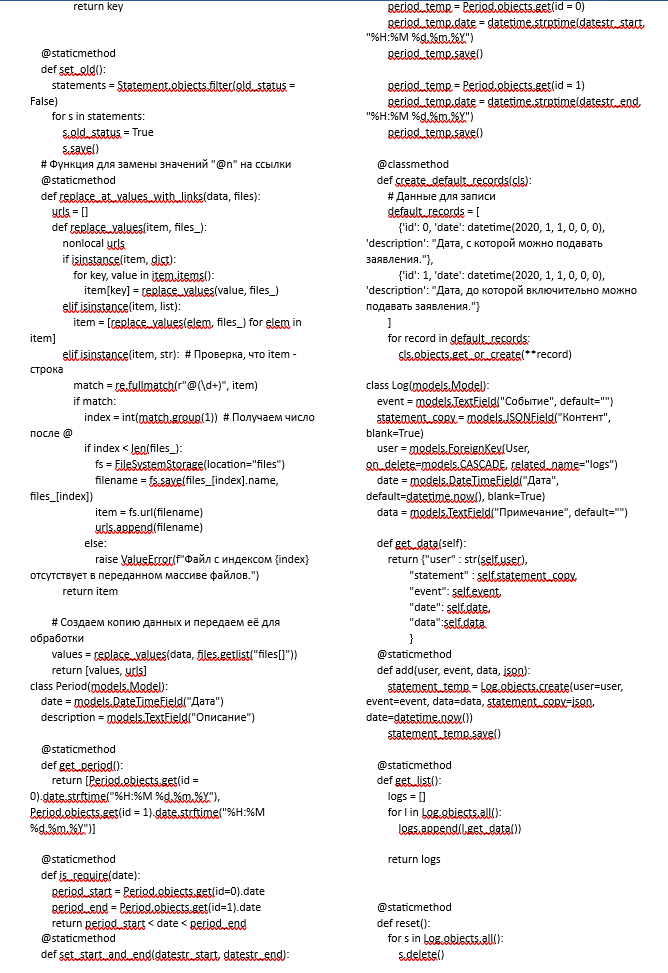


Рис.П2.9. Код