**РТУ МИРЭА**

**Кейс 3 - “Система поселения студентов в общежитии”**

**Название команды:**

Объединение Программистов Здравомыслящих

**Подготовили:**

Аксенов К.В - Teamlead.

Мрясова А.А - Backend.

Святкин С.С - Backend.

Хайбуллов В.Р - Frontend.

Полошкова А.Ю - Frontend.

# ОГЛАВЛЕНИЕ

[**ОГЛАВЛЕНИЕ 1**](#_lu7vies4u76f)

[**Вступление 2**](#_d3296s3ecpnx)

[Идея: 2](#_55p49tz6p75m)

[Проблемы, решаемые проектом: 2](#_94of3fbgj61c)

[**Структура проекта: 3**](#_9s6v8nvcww3q)

[1. Back-end: 3](#_8ef13jjwfnx)

[2. Frontend: 4](#_fj6ize2gyy3z)

[3. База данных: 4](#_reo22z6iraef)

[4. Система аккаунта студента: 4](#_o77r9oj8nktl)

[**Back-end Стек Технологий 5**](#_mrzii77ga3c2)

[1. Node.js 5](#_tksi1qvlkncr)

[2. Express.js 7](#_xug3q5m4w2mm)

[3. PostgreSQL 7](#_y4ko36lh1cjg)

[4. Python 8](#_1i2vpwkr5wb3)

[**Микросервисы: 9**](#_fbey8ozbppsd)

[1. Authentication Service (Сервис аутентификации): 9](#_hye9mwadt7in)

[2. Student Data Service (Сервис управления данными студентов): 9](#_wvlvgxg1iipi)

[3. Residence Management Service (Сервис управления проживанием): 9](#_3pq9zzgriws)

[4. Document Management Service (Сервис управления документами): 10](#_g8glrv61uknp)

[5. Student Requests Management Service (Сервис управления заявками студентов): 10](#_bjue9emjayhp)

[6. Notification Service (Сервис уведомлений): 10](#_hcvoyo8esmco)

[7. Announcement Service (Сервис объявлений): 11](#_35huhenghstm)

[**Безопасность 11**](#_7hjfknpfpcn)

[1. TLS/SSL (Transport Layer Security/Secure Sockets Layer). 11](#_12tn4v8d3m0q)

[2. AES (Advanced Encryption Standard). 11](#_o2uwwwalemch)

[3. RSA (Rivest-Shamir-Adleman). 11](#_er8anix11vv8)

[4. SHA-256 (Secure Hash Algorithm 256-bit). 11](#_ctw2s0wupoth)

[5. Использование JWT-токенов. 12](#_z9d0tzbe3e8)

[6. Аудит и журналирование. 12](#_1dnc6lm050fr)

[**User Story 13**](#_cr62hba1uvk7)

[User Story 1 “Выбор комнаты”: 13](#_iyet816g4axa)

[User Story 2 “Новости для студентов”: 14](#_napm5o45bs28)

[User Story 3 “Создание новости”: 14](#_bdwhug409iq1)

[User story 4 “Заявки на заселение”: 15](#_xz3hh73ed34t)

[User Story 5 “Оплата общежития”: 16](#_ndb2yi9ubfk9)

[User story 6 “Подача заявления на ремонт”: 17](#_kbggn05wjgmk)

[User story 7 “Подача заявление на временный отъезд”: 18](#_wwgxurf0mu9i)

[User story 8 “Оформление временного пропуска для посетителя”: 19](#_1pn961mt72oy)

[**RESTful API 20**](#_keblga85q2sg)

[GET-запросы 20](#_mg4zqr5ad9oo)

[POST-запрос 22](#_1616gsn7iu3r)

[PUT-запрос 27](#_di65z6npmiw8)

[DELETE-запрос 28](#_lyj186k0bc4k)

# **Вступление**

## Идея:

Наш проект зародился из стремления улучшить систему управления в общежитиях, предоставляя студентам и администрации инновационные инструменты для эффективного взаимодействия и повышения качества услуг.

## Проблемы, решаемые проектом:

Мы решаем ряд проблем, с которыми сталкиваются студенты и администрация общежитий, такие как неэффективное управление заселением, долгими и многоступенчатыми процессами подачи заявок, а также недостаточной информационной прозрачностью и ограниченным доступом к важным сервисам. Наш проект нацелен на создание инновационной платформы, которая решит эти проблемы, сделает процессы более простыми и удобными для всех участников.

# 

# 

# 

# 

# **Структура проекта:**

## 1. **Back-end**:

*Управление заселением*:

Модуль для добавления, выселения, переселения студентов, в том числе управление информацией об этом (комната, этаж, статус проживания и т.д.)

*Документы для студента*:

Создание электронных документов для студента (требования для предоставления, справки, квитанции) и их загрузка.

*Работа с электронными документами*:

Реализация функций для работы с электронными документами, их

генерация, хранение и предоставление для студентов по запросу.

*Оформление пропусков*:

Модуль для оформления и управления пропусками для посетителей студентов.

*Уведомления и оповещения*:

Разработка системы уведомлений для студентов о просроченности документов или справок, оповещения об изменениях в заселении или других событиях.

## 2. **Frontend**:

*Интерфейс для студента*:

Страницы для отправки запросов на ремонт, оформления пропусков для гостей, уведомлений об отъезде и приезде.

Формы для загрузки документов.

*Интерфейс для коменданта*:

Панель управления для одобрения заявок на предоставление общежития, управления заселением, формирования документов, создания объявлений и т.д.

*Общие функциональные модули*:

Компоненты для поиска студентов в базе, формирования квитанций на оплату общежития, создания общих объявлений.

## 3. **База данных**:

*Хранение информации*:

Хранение данных о студентах, их заселении, документах, пропусках, истории заявок и прочей связанной информации.

## 4. **Система аккаунта студента**:

*Аутентификация через API вуза*:

Идентификация студентов:

Настройка системы аутентификации для общего доступа студентов через API вуза.

Реализация механизма, который позволяет студентам аутентифицироваться на вашем сайте, используя свои учетные данные из личного кабинета вуза.

*Уровни доступа*:

Управление разрешениями:

Разграничение уровней доступа студентов в зависимости от их статуса в университете (например, обучающийся, аспирант, выпускник и т.д.).

Предоставление различных привилегий доступа на основе этого статуса (например, доступ к определенным функциям на сайте общежития).

*Синхронизация данных*:

Использование данных из базы вуза:

Получение и синхронизация данных о студентах (имя, фамилия, и т.д.) из базы данных вашего вуза.

Автоматическое обновление информации о студентах на сайте общежития при изменении данных в личном кабинете вуза.

*Интеграция с API*:

Взаимодействие с API университета:

Создание функциональных возможностей для взаимодействия вашего сайта общежития с API вашего вуза для получения и обновления данных студентов.

# **Back-end Стек Технологий**

Backend (Серверная часть):

* Node.js для создания сервера и обработки запросов от клиента.
* Python для обработки данных на стороне сервера
* Express.js для упрощения маршрутизации и обработки HTTP-запросов.
* PostgreSQL для хранения рецептов, пользовательских данных, истории и избранных рецептов.
* MongoDB для хранения документов студентов
* Open-source библиотека node-postgres для взаимодействия между базой данных и серверной частью

## Node.js

Node.js — это open-source кроссплатформенная среда выполнения для JavaScript, которая работает на серверах. С момента выпуска этой платформы в 2009 году она стала чрезвычайно популярной и в наши дни играет весьма важную роль в области веб-разработки.

Платформа Node.js построена на базе JavaScript движка V8 от Google, который используется в браузере Google Chrome. Данная платформа, в основном, используется для создания веб-серверов, однако сфера её применения этим не ограничивается.

JavaScript значительно упрощает написание асинхронного и неблокирующего кода с использованием единственного потока, функций обратного вызова и подхода к разработке, основанной на событиях. Асинхронные механизмы позволяют единственному Node.js-серверу одновременно обрабатывать тысячи подключений, не нагружая при этом программиста задачами по управлению потоками и по организации параллельного выполнения кода. Подобные вещи часто являются источниками ошибок.

Разработчик [Эмиль Привер](https://github.com/emilpriver/go-rust-bun) провел [простой HTTP-тест](https://www.priver.dev/blog/benchmark/go-vs-rust-vs-bun-vs-node-http-benchmark/) «Go против Rust против Node». Он создал эталонный тест для измерения количества HTTP-вызовов, которые сервер может обработать в секунду. В тесте сравнивается производительность GO, Rust и Node, поскольку эти языки часто используются веб-разработчиками для аналогичных целей.

Хоть по чистой производительности Go - бесспорный лидер, в реальной жизни Node.js, Go и Rust показывают одинаково хорошую производительность. У Node.js есть отличные варианты почти для любого типа задач разработки. Несмотря на то, что Go способен справиться с более высокой нагрузкой, Node.js существенно упрощает разработку в связи с широким выбором фреймворков и интеграций.

*Таблица 1. Сравнение производительности*

|  | Rust | Go | Node |
| --- | --- | --- | --- |
| Задержка (ms) | 42.33 | 43.38 | 52.50 |
| Запросов / секунду (тыс. / с) | 1.5 | 1.46 | 1.23 |
| Всего запросов | 718755 | 698266 | 587719 |

## Express.js

Этот минималистичный фреймворк предоставляет базовые инструменты для создания API и веб-приложений. Он известен своей простотой и гибкостью, что делает его отличным выбором для начинающих и опытных разработчиков.

Маршрутизация в Express.js позволяет определить, как приложение реагирует на различные URL-адреса и HTTP-методы. Маршруты могут быть параметризованными и поддерживать разнообразные обработчики запросов. В ExpressJS вы можете определить маршруты с помощью методов, соответствующих HTTP-методам (GET, POST, PUT, DELETE и другие).

Express.js также позволяет использовать множество решений для связующего программного обеспечения, что помогает ускорять работу клиентских приложений, избегать зависимости от API и снижать затраты на разработку.

Достоинства Express.js:

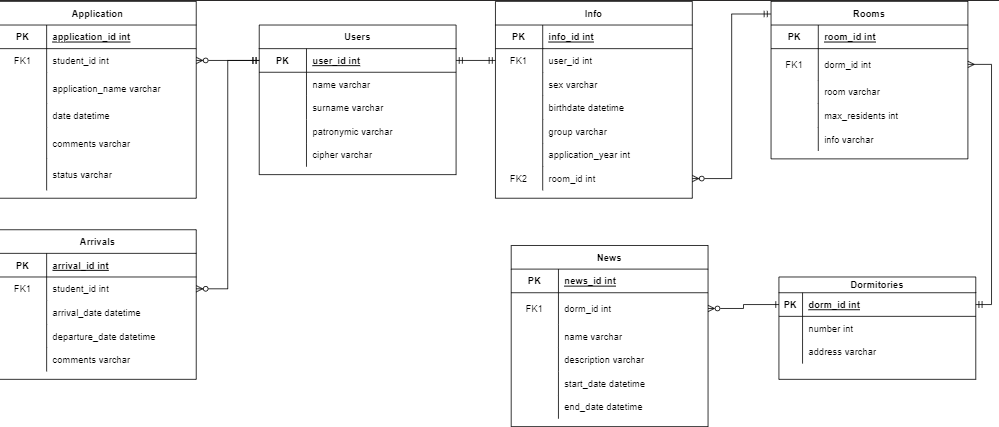
1. Обширный функционал как для веб-приложений, так и предназначенных для мобильных устройств и персональных компьютер
2. Поддерживает интеграцию с различными вариациями middleware.
3. Простота разработки и нативная поддержка в javascript.

## PostgreSQL

База данных — это один из основных компонентов любого приложения, и правильный выбор базы данных является ключевым моментом в разработке любого проекта. PostgreSQL это одна из самых востребованных баз данных на сегодняшний день благодаря своим широким возможностям, open-source подходу и развитому сообществу.

PostgreSQL — объектно-реляционная СУБД. Это значит, что она поддерживает и объектный, и реляционный подход.

Вышеперечисленные преимущества существенно облегчат разработку нашего проекта, так как специфика данных, используемых приложением, таких как записи о рецептах, предполагает хранение различных структур данных, а также использование взаимосвязей между записями в базе данных.

Архитектура базы данных

## Python

Python — это язык программирования высокого уровня для решения обширного перечня задач, в первую очередь ориентированный на быстрое решение поставленной задачи. У него много библиотек, API и вспомогательных инструментов, потому что он существует уже более трех десятилетий. В результате Python широко используется в коммерческих приложениях и подходит для многих различных стилей и методов программирования.

Django - это бесплатная платформа веб-разработки на Python с открытым исходным кодом, используемая при создании веб-сайтов. Он был создан в 2003 году c использованием шаблона Model-Template-View. Он популярен благодаря своей надежной и простой природе. Это один из самых популярных фреймворков во всем мире.

Среди фреймворков Django, Flask и FastAPI у Django больше всего модулей, которые позволяют повторно использовать код. Это платформа веб-разработки с полным стеком, в отличие от Flask и FastAPI, которые являются минималистичными фреймворками, используемыми для создания быстрых веб-сайтов.

Основными плюсами Django являются:

* Эффективная структура кода, позволяющая разработчикам добавлять большое количество функций на свои веб-сайты.
* DjangoRestFramework, являющийся гибким набором инструментов для разработки WebAPI для Django проектов.
* Разработка WebAPI достаточно проста, в связи с модульностью и настраиваемой архитектурой библиотеки.
* Высокая степень безопасности, включающая защиту от SQL-инъекций и DDos-атак

## MongoDB

MongoDB — система управления базами данных, которая работает с документо ориентированной моделью данных. В отличие от реляционных СУБД, MongoDB не требуются таблицы, схемы или отдельный язык запросов. Информация хранится в виде документов либо коллекций.

Это файловая СУБД, данные могут храниться в форматах JSON, BSON, благодаря чему можно хранить фото, видео и аудиоматериалы и более эффективно хранить данные, например, об одном предмете/продукте/человеке.

MongoDB нативно поддерживает интеграцию с JavaScript, который также используется в нашем проекте. Таким образом, ее использование значительно облегчит и ускорит разработку проекта.

MongoDB обладает хорошей отказоустойчивость за счет журналирования и репликации данных, что крайне важно для хранения документов и прочих важных данных студентов. Также MongoDB практически не имеет ограничение на масштабируемость, что также важно для хранения данных, таких как документы, объем которых заранее нельзя предсказать.

Из минусов MongoDB можно отметить больших расход серверной памяти из-за механизмов защиты данных, а также более медленную скорость работы в сравнении с реляционными базами данных.

# 

# **Front-end Стек Технологий**

* React.js для построения пользовательского интерфейса и взаимодействия с back-end частью.
* Redux для управления состоянием приложения (хранение и обновление данных о рецептах, ингредиентах и действиях пользователя).
* HTML/CSS для визуального представления контента и стилей.
* JavaScript для логики и взаимодействия с API.

## HTML + CSS:

HTML (HyperText Markup Language) - это стандартный язык разметки, используемый для создания структуры веб-страниц. Он определяет, как элементы контента (текст, изображения, ссылки и другие) должны быть организованы на странице. Без HTML веб-страницы были бы просто текстом без смысла.

CSS (Cascading Style Sheets) - это язык, который определяет визуальное оформление веб-страниц. CSS позволяет задавать цвета, шрифты, размеры и расположение элементов на странице. Он также позволяет создавать адаптивный дизайн, чтобы страница выглядела хорошо на разных устройствах и разрешениях экрана.

## JavaScript, фреймворк React.js:

JavaScript - это язык программирования, который позволяет создавать интерактивные и динамические веб-страницы. Он может обрабатывать события, выполнять асинхронные запросы к серверу, изменять содержимое страницы без перезагрузки и многое другое. JavaScript является фундаментальным элементом веб-разработки.

React.js - это современная библиотека JavaScript, которая позволяет организованно разрабатывать пользовательские интерфейсы для веб-сайтов и приложений. Она предоставляет инструменты для организации кода в виде компонентов, что упрощает управление и поддержку проекта. React любят за его высокую производительность. У него отличная поддержка рендеринга на стороне сервера, что делает его мощной платформой для контент-ориентированных приложений.

Сравнение с Vue.js:

Vue - ещё один популярный фронтенд-фреймворк, который имеет простой и интуитивно понятный API. Однако иногда у Vue.js могут возникнуть проблемы при интеграции в большие проекты, и пока еще нет опыта возможных решений. Система рендеринга React более функциональна. Она предоставляет больше возможностей для отладки.

Почему был выбран React.js:

React.js отличается повышенной эффективностью и большим набором инструментов. Используя React JS, вы можете сделать легкие веб приложения. Можно запустить React на стороне сервера, и виртуальный DOM будет доставлен. Также имеет большой плюс в виде платформы для управления состоянием приложений Redux, ее легко учить и использовать.

В итоге, выбор HTML, CSS, JavaScript и React.js как стека фронтенд технологий обеспечивает инструменты для создания функциональных и красочных веб-приложений, сохраняя при этом простоту поддержки и повышенную эффективность.

# 

# **Микросервисы:**

## 1. **Authentication Service** (Сервис аутентификации):

*Функционал*:

Проверка учетных данных студентов через API вашего вуза.

Предоставление токена доступа после успешной аутентификации.

*Задачи*:

Взаимодействие с API университета для проверки учетных данных студентов.

Выдача JWT токена доступа для использования другими сервисами системы.

## 2. **Student Data Service** (Сервис управления данными студентов):

*Функционал*:

Получение данных о студентах из базы данных вашего вуза через API.

Сохранение информации о студентах в базе данных общежития.

*Задачи*:

Разработка функций для синхронизации данных студентов между системой общежития и базой данных вуза.

Обеспечение безопасности данных и их актуальности.

## 3. **Residence Management Service** (Сервис управления проживанием):

*Функционал*:

Добавление, выселение, переселение студентов в общежитии.

Управление информацией о статусе проживания (номер комнаты, этаж, статус заселения).

*Задачи*:

Реализация операций управления проживанием студентов.

Обеспечение корректного отображения информации о проживающих в системе.

## 4. **Document Management Service** (Сервис управления документами):

*Функционал*:

Генерация документов для студентов (требования для предоставления, справки, квитанции) и их загрузка.

Хранение электронных документов студентов.

*Задачи*:

Создание и предоставление документов в соответствии с запросами студентов и администрации.

Обеспечение безопасности и целостности электронных документов.

## 5. **Student Requests Management Service** (Сервис управления заявками студентов)**:**

*Функционал:*

Этот микросервис предоставляет студентам возможность подавать заявки на различные услуги или события, связанные с их проживанием в общежитии.

*Задачи:*

Создание заявок и управление заявками реализация функционала для просмотра и управления статусами заявок принятие к исполнению, отклонение, выполнение и т.д.

## 6. **Notification Service** (Сервис уведомлений):

*Функционал*:

Отправка уведомлений студентам о событиях, изменениях статуса, просроченных документах и других важных событиях.

*Задачи*:

Разработка системы отправки уведомлений на основе событий, требующих уведомления студентов.

## 7. **Announcement Service** (Сервис объявлений):

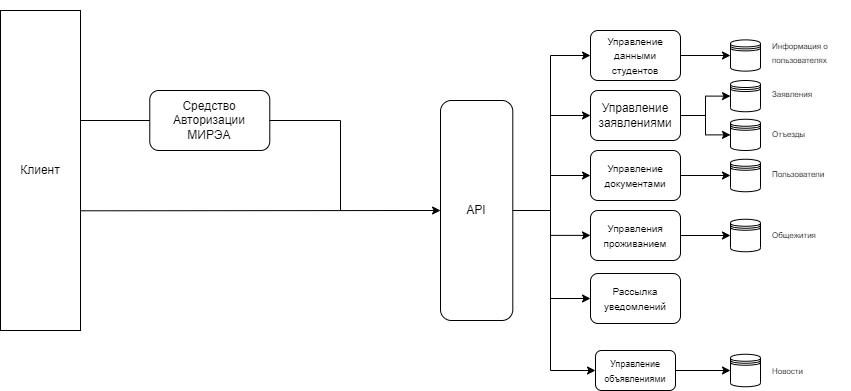
*Функционал*:

Создание и управление общими объявлениями для студентов, рассылка информации.

*Задачи*:

Разработка интерфейса для создания и управления объявлениями.

Реализация функции рассылки объявлений студентам.



# **Безопасность**

## 1. TLS/SSL (Transport Layer Security/Secure Sockets Layer).

Используем TLS/SSL для шифрования данных, передаваемых через сеть, включая мобильные приложения и веб-интерфейсы умного дома. TLS/SSL обеспечивает шифрование данных на уровне транспортного уровня и защищает их от прослушивания и изменения в процессе передачи. Эти протоколы также обеспечивают аутентификацию сервера, что предотвращает атаки MITM-атаки.

## 2. AES (Advanced Encryption Standard).

AES — симметричный алгоритм блочного шифрования, признанный одним из самых надежных. В нашем проекте алгоритм используется для шифрования данных, хранящихся на сервере для обеспечения высокой степени безопасности.

## 3. RSA (Rivest-Shamir-Adleman).

Мы используем криптографический алгоритм RSA для обмена ключами и шифрования данных, таких как пароли и секреты.

## 4. SHA-256 (Secure Hash Algorithm 256-bit).

Используем SHA-256 для хэширования паролей и других конфиденциальных данных. SHA-256 является криптографически стойкой функцией хеширования, что делает ее подходящей для хранения хэшированных паролей и других конфиденциальных данных.

## 5. Использование JWT-токенов.

JWT (JSON Web Token) — это стандартизированный формат для безопасной передачи информации между двумя сторонами в формате JSON. В нашем проекте используется для аутентификации пользователя и безопасной передачи информации между клиентом и сервером.

## 6. Аудит и журналирование.

Журнал событий и аудита позволяет анализировать прошлые события, регистрировать ошибки и подозрительную активность. Журнал аудита также упрощает трассировку событий и анализ используемых ресурсов, что позволит исправлять ошибки и оптимизировать наш сервис.

# 

# **User Story**

## User Story 1 “Выбор комнаты”:

Как студент, я хочу иметь возможность выбрать комнату для заселения в общежитии через интерактивную карту, чтобы принять информированное решение о своем проживании.

1. Как студент, я могу открыть интерактивную карту общежития, на которой отображены все этажи и комнаты с их текущим статусом заселенности.

2. Я могу просматривать каждый этаж и видеть доступные комнаты на схеме с указанием числа свободных мест или количества заселенных человек.

3. Комнаты, которые полностью заняты, отображаются серым цветом.

4. Комнаты с доступными местами отображаются специальным образом, показывая количество заселенных человек (например, "2/3 занято").

5. При выборе комнаты, я могу получить более подробную информацию о ней, такую как площадь, наличие удобств, стоимость и прочее.

6. После выбора комнаты, я могу подтвердить свой выбор и забронировать её для заселения.

7. В случае отсутствия доступных комнат, мне предоставляется информация о том, что нет свободных мест, и возможность поставить себя в список ожидания или выбрать другое общежитие.

Последовательность действий:

1. Зайти на сайт общежития и выбрать опцию "Выбрать комнату".

2. Просмотреть карту общежития с отображением доступных комнат и их заселенностью.

3. Выбрать конкретную комнату, просмотреть информацию о ней.

4. Подтвердить выбор и забронировать комнату для заселения.

Дополнительные возможности:

- Возможность связаться с администрацией общежития для получения дополнительной информации или помощи в выборе комнаты.

## User Story 2 “Новости для студентов”:

Как студент, я хочу иметь доступ к информации о последних новостях и событиях в общежитии на главной странице сайта, чтобы быть в курсе актуальной информации, такой как уведомления о заселении различных курсов.

1. Как студент, я захожу на главную страницу сайта общежития и вижу блок с новостями или уведомлениями.

2. На главной странице отображаются последние новости, включая информацию о заселении различных курсов (например, "График заселения на 2023-2024 учебный год").

3. Каждая новость содержит заголовок, который отражает основную суть информации.

4. Новости отображаются в хронологическом порядке, с последними обновлениями сверху, для удобства доступа к самой актуальной информации.

Последовательность действий:

1. Зайти на главную страницу сайта общежития.

2. На главной странице увидеть блок с новостями или уведомлениями.

3. При необходимости просмотреть подробную информацию по новости для получения дополнительных деталей или контактной информации.

## User Story 3 “Создание новости”:

Как комендант общежития, я хочу иметь возможность добавлять информацию и новости на сайт общежития, чтобы уведомлять жильцов об актуальных событиях и изменениях.

1. Как комендант, я вхожу в систему управления сайтом общежития, имея права доступа к функционалу добавления новостей.

2. На странице управления новостями предусмотрена опция "Добавить новость".

3. Форма добавления новости включает поля для название новости, описания новости, даты проведения, возможно добавление изображений и документов для новости.

5. После заполнения всех необходимых полей, я могу опубликовать новость для отображения на главной странице сайта общежития.

6. Предусмотрена возможность редактирования или удаления ранее опубликованных новостей для обновления информации или корректировки ошибок.

7. Новость отображается в соответствии с дизайном сайта и доступна жильцам общежития для прочтения на главной странице.

Примерная последовательность действий:

1. Войти в систему управления сайтом общежития с помощью своих учетных данных.

2. Найти раздел для управления новостями на сайте.

3. Нажать на опцию "Добавить новость".

4. Заполнить поля названия, описания новости, установить дату проведения и добавить изображения или документ, если необходимо.

5. Проверить информацию и при необходимости внести коррективы.

6. Нажать кнопку "Сохранить" для размещения новости на сайте общежития.

Дополнительные возможности:

- Уведомление жильцов общежития через электронную почту или SMS о новых опубликованных новостях для максимальной охватываемости аудитории.

## User story 4 “Заявки на заселение”:

Как комендант общежития, я хочу получать уведомления о заявках на заселение студентов и иметь доступ к информации о выбранной комнате и предоставленных документах, чтобы проверить и подтвердить заявку на проживание в общежитии.

1. Как комендант, я получаю уведомление о новой заявке от студента на заселение в общежитие.

2. Уведомление содержит основную информацию о студенте, включая имя, и выбранную комнату для заселения.

3. При переходе к просмотру подробной информации о заявке, я вижу список прикрепленных документов, загруженных студентом для подтверждения личности или других необходимых документов.

4. Информация о комнате, в которую студент хочет заселиться, включает номер комнаты, этаж, количество свободных мест и другие детали (например, общее количество койко-мест в комнате).

5. Мной предпринимаются действия по проверке представленных документов студента для соответствия требованиям общежития.

6. После проверки документов я подтверждаю или отклоняю заявку студента на заселение в указанную комнату.

7. При подтверждении заявки студента, его информация добавляется в список жильцов комнаты, указанной в заявке.

Примерная последовательность действий:

1. Получить уведомление о новой заявке на заселение студента.

2. Просмотреть основную информацию о студенте и выбранной комнате.

3. Перейти к просмотру прикрепленных документов для проверки их соответствия требованиям общежития.

4. Принять решение о подтверждении или отклонении заявки на заселение, основываясь на проверенной информации и предоставленных документах.

5. При подтверждении заявки обновить информацию о жильцах выбранной комнаты.

Дополнительные возможности:

- Возможность отправки уведомления студенту о статусе его заявки (подтверждение или отклонение) через сайт, электронную почту или SMS.

- Возможность контактировать со студентом для уточнения информации или запроса дополнительных документов, если это необходимо для принятия решения о заселении.

## User Story 5 “Оплата общежития”:

Как студент, я хочу иметь возможность оплатить проживание в общежитии на полгода через сайт, чтобы удобно и безопасно совершить платеж.

1. Как студент, я захожу на веб-сайт общежития и перехожу в раздел оплаты.

2. На странице оплаты доступны два варианта оплаты: оплата через QR-код или получение банковских реквизитов для перевода средств.

3. При выборе оплаты через QR-код генерируется уникальный QR-код, который я могу отсканировать с помощью мобильного банковского приложения или специализированных сервисов для совершения оплаты.

4. В случае выбора оплаты по банковским реквизитам на странице отображаются необходимые данные: номер счета, реквизиты банка и прочая информация, необходимая для проведения банковского перевода.

5. После успешного выполнения платежа система подтверждает оплату и предоставляет мне подтверждение оплаты в виде электронного чека или уведомления об успешной транзакции.

6. После оплаты на моем личном кабинете или в разделе "История платежей" отображается информация о совершенной транзакции с указанием суммы, даты и способа оплаты.

Примерная последовательность действий:

1. Зайти на веб-сайт общежития и перейти в раздел оплаты проживания.

2. Выбрать желаемый способ оплаты: через QR-код или банковские реквизиты.

3. При выборе оплаты через QR-код сканировать сгенерированный QR-код через мобильное банковское приложение или сервис для осуществления платежа.

4. При выборе оплаты по банковским реквизитам скопировать необходимые данные и осуществить банковский перевод через интернет-банк или банковское отделение.

5. Получить подтверждение об успешной оплате.

Дополнительные возможности:

- Оповещение о необходимости оплаты за несколько дней до окончания срока оплаты для предотвращения просрочки платежа.

## User story 6 “Подача заявления на ремонт”:

Как студент, я хочу иметь возможность подать заявление о необходимости ремонта в моей комнате, чтобы обращаться за помощью и получить оперативное решение проблемы.

1. Как студент, я захожу на сайт общежития в специальный раздел для обращений/заявлений о ремонте.

2. В разделе подачи заявления о ремонте предоставлена форма, где я могу описать проблему подробно, загрузить фотографии, и указать удобное время для осмотра комнаты специалистом.

3. После отправки заявления система предоставляет мне уведомление о получении обращения и информацию о том, что запрос был передан на рассмотрение соответствующему отделу или специалисту по ремонту.

5. В течение установленного срока, установленного правилами общежития или обычной практикой, получаю ответ или уведомление о том, когда можно ожидать приезд специалиста для осмотра проблемы.

Примерная последовательность действий:

1. Войти в личный кабинет на сайте общежития и перейти в раздел для подачи заявлений о ремонте.

2. Заполнить форму заявления, описать проблему, и указать предпочтительное время осмотра проблемы.

3. Отправить заявление.

4. Получить уведомление о получении заявления и информацию о дальнейших шагах по рассмотрению обращения.

Дополнительные возможности:

- Возможность отслеживания статуса обращения о ремонте через личный кабинет для получения актуальной информации о его рассмотрении и решении.

## User story 7 “Подача заявление на временный отъезд”:

Как студент, я хочу иметь возможность подать заявление о временном отъезде из общежития, чтобы предоставить информацию администрации и быть уверенным в сохранении своего места в общежитии во время отсутствия.

1. Как студент, я захожу в личный кабинет на сайте общежития в раздел, предназначенный для подачи заявлений.

2. На странице подачи заявления о временном отъезде доступна форма, где я могу указать даты начала и окончания отсутствия и причину отъезда.

3. После заполнения формы отправляю заявление.

4. Получаю уведомление о получении заявления и информацию о том, что оно передано на рассмотрение администрации общежития.

5. Получаю подтверждение временного отсутствия и информацию о том, что моё место в общежитии сохранено на указанный период времени.

6. Возвращаюсь по окончании временного отсутствия и восстанавливаю своё проживание в общежитии в соответствии с утвержденными датами.

Примерная последовательность действий:

1. Войти в личный кабинет на сайте общежития или в специальный раздел для подачи заявлений.

2. Заполнить форму заявления, указав даты отъезда.

3. Отправить заявление.

4. Получить уведомление об успешной отправке заявления.

5. Получить уведомление о статусе заявления и условиях временного отсутствия.

6. По возвращении в общежитие убедиться, что мое пребывание возобновлено в соответствии с предоставленными датами.

Дополнительные возможности:

- Возможность получения подробной информации о процедуре временного отсутствия и условиях возврата в общежитие.

- Возможность отслеживания статуса заявления в личном кабинете для ведения контроля за процессом рассмотрения и принятия решения.

## User story 8 “Оформление временного пропуска для посетителя”:

Как студент, я хочу иметь возможность оформить временный пропуск для посетителя, чтобы обеспечить его доступ в общежитие в определенное время.

1. Как студент, захожу в личный кабинет на сайте общежития в раздел, предназначенный для запросов на оформление временных пропусков.

2. На странице оформления временного пропуска доступна форма, где я могу указать даты и время посещения посетителя, его персональные данные и цель посещения общежития.

3. Заполняю форму и отправляю запрос на получение временного пропуска для посетителя.

4. Получаю уведомление об успешной отправке запроса и информацию о том, что он передан на рассмотрение администрацией общежития.

5. По утверждению запроса, получаю временный пропуск для посетителя с указанными датами и условиями входа в общежитие.

6. Предоставляю временный пропуск посетителю для его доступа внутрь общежития в соответствии с указанными условиями посещения.

Примерная последовательность действий:

1. Захожу в личный кабинет на сайте общежития в специальный раздел для оформления временных пропусков.

2. Заполняю форму оформления временного пропуска для посетителя, указывая даты и время посещения, его персональные данные и цель визита.

3. Отправляю запрос на получение временного пропуска.

4. Получаю уведомление об успешной отправке запроса.

5. По утверждению запроса, получаю временный пропуск для посетителя с указанными датами и условиями входа.

6. Передаю временный пропуск посетителю для его доступа внутрь общежития в соответствии с указанными условиями посещения.

Дополнительные возможности:

- Возможность связаться с администрацией для уточнения деталей или получения помощи в оформлении временного пропуска для посетителя.

- Возможность отслеживания статуса запроса на получение временного пропуска в личном кабинете для контроля за процессом его рассмотрения и принятия решения.

# **RESTful API**

## **GET-запросы**

**Авторизация пользователей в системе**

Тестовые данные:

Пользователь:

Email: example@edu.mirea.ru,

Пароль: password

Тело запроса:

{

"email": “example@edu.mirea.ru”,

"password": “password”

}

Тело ответа (200 ОК):

{

"status\_response": true,

"description\_ru": “Авторизация успешно завершена.”

}

Тело ответа (404 Not found) при некорректном вводе информации:

{

"statusCode": 404,

"status\_response": false,

"description": "Ошибка авторизации. Проверьте введенные данные."

}

**Получить информацию о студенте**

Тестовые данные:

Информация:

ID студента: 1289318

Тело запроса:

{

"ID": "1289318"

}

Тело ответа (200 ОК):

{

"name": "name",

"surname": "surname",

"lastname": "lastname",

"gender": "m",

"birthdate": "21.03.2004",

"group": "ИКБО-43-24",

"applicationyear": "2024",

"cipher": "24К132",

"roomID": "422"

}

Тело ответа (404 Not Found) при некорректном вводе информации:

{

"statusCode": 404,

"status": false,

"description": "Пользователь не найден."

}

## **POST-запрос**

**Создание записи о студенте**

Тестовые данные:

Пользователь:

имя: name  
 фамилия: surname  
 отчество: lastname  
 пол: m  
 дата рождения: 21.03.2004  
 учебная группа: ИКБО-43-24  
 год поступления: 2024  
 шифр: 24К132  
 комната: 422

Тело запроса:

{

"name": "name",

"surname": "surname",

"lastname": "lastname",

"gender": "m",

"birthdate": "21.03.2004",

"group": "ИКБО-43-24",

"applicationyear": "2024",

"cipher": "24К132",

"roomID": "422"

}

Тело ответа (200 ОК):

{

"status\_response": true,

"description\_ru": "Данные успешно сохранены.”

}

Тело ответа (403 Forbidden):

{

"statusCode": 403,

"status": false,

"description": "Нет доступа."

}

Тело ответа (406 Not Acceptable) при некорректном вводе информации:

{

"statusCode": 406,

"error": "Not Acceptable",

"message": "Неверный параметр {parameter}"

}

**Загрузить электронный документ**

Тестовые данные:

Файл:

Название файла: name  
 Пользователь: 1289318

Тело запроса:

{

"userID": "1289318",

"date":"2023-03-01T13:00:00Z",

"filename": "name",

"file-data":"SGVsbG8gd29ybGRIZWxsbyB3b…3JsZEhlbGxvIHdvcmxkybGRIZWx"

}

Тело ответа (200 ОК):

{

"status": true,

"description": "Файл успешно загружен."

}

Тело ответа (404 Not Found) при некорректном вводе информации:

{

"statusCode": 404,

"status": false,

"description": "Пользователь не найден."

}

Тело ответа (406 Not Acceptable) при некорректном вводе информации:

{

"statusCode": 406,

"error": "Not Acceptable",

"message": "Неверная кодировка документа"

}

**Оформить гостевой пропуск**

Тестовые данные:

Информация:

Имя: name  
 Описание: description  
 Время: 2023-12-16, 13:00:00

Тело запроса:

{

"userID": "8126312",

"name": "name",

"description": "description",

"date": "2023-12-16T13:00:00Z"

}

Тело ответа (200 ОК):

{

"status": true,

"description": "Заявка на пропуск оформлена."

}

Тело ответа (406 Not Acceptable) при некорректном вводе информации:

{

"statusCode": 406,

"error": "Not Acceptable",

"message": "Неверная Дата."

}

**Оформить онлайн заявление**

Тестовые данные:

Информация:

Описание: description  
 Время: 2023-12-16, 13:00:00  
 Дата (опционально): 2023.12.16 - 2023.12.20

Тело запроса:

{

"userID": "8126312",

"type": "requestType",

"description": "description",

"date": "2023.12.16 - 2023.12.20"

}

Тело ответа (200 ОК):

{

"status": true,

"description": "Заявка оформлена."

}

Тело ответа (406 Not Acceptable) при некорректном вводе информации:

{

"statusCode": 406,

"error": "Not Acceptable",

"message": "Неверная Дата."

}

## **PUT-запрос**

**Изменение информации пользователя**

Тестовые данные:

Пользователь:

имя: name  
 фамилия: surname  
 отчество: lastname  
 пол: m  
 дата рождения: 21.03.2004  
 учебная группа: ИКБО-43-24  
 год поступления: 2024  
 шифр: 24К132  
 комната: 422

Тело запроса:

{

"name": "name",

"surname": "surname",

"lastname": "lastname",

"gender": "m",

"birthdate": "21.03.2004",

"group": "ИКБО-43-24",

"applicationyear": "2024",

"cipher": "24К132",

"roomID": "422"

}

Тело ответа (200 ОК):

{

"status\_response": true,

"description\_ru": "Данные успешно обновлены.”

}

Тело ответа (403 Forbidden):

{

"statusCode": 403,

"status": false,

"description": "Нет доступа."

}

Тело ответа (406 Not Acceptable) при некорректном вводе информации:

{

"statusCode": 406,

"error": "Not Acceptable",

"message": "Неверный параметр {parameter}"

}

## **DELETE-запрос**

**Удалить информацию о студенте**

Тестовые данные:

Рецепт:

ID студента: 51223  
 Комментарий: description

Тело запроса:

{

"userID": "1234132",

"studentID": "9123123",

"description": "description"

}

Тело ответа (200 ОК):

{

"status": true,

"description": "Запись удалена удален."

}

Тело ответа (403 Forbidden) при некорректном вводе информации:

{

"statusCode": 403,

"status": false,

"description": "Нет доступа."

}

Тело ответа (404 Not Found) при некорректном вводе информации:

{

"statusCode": 404,

"status": false,

"description": "Пользователь не найден."

}