МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01 «Информатика и технологии программирования»

Отчет по преддипломной практике

на тему: «Программный комплекс автоматизации обслуживания жилого фонда студенческого общежития»

Исполнитель: студент гр. ИП-42 Пархоменко П.Л.

Руководитель от предприятия:

Макаревич В.Л.

Руководитель: преподаватель

Шибеко В.Н.

Дата проверки:

Дата допуска к защите:

Дата защиты:

Оценка работы:

Подписи членов комиссии

Гомель 2023

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc136201926)

[1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ АВТОМАТИЗАЦИИ ЖИЛОГО ФОНДА СТУДЕНЧЕСКОГО ОБЩЕЖИТИЯ 4](#_Toc136201927)

[1.1 Обзор существующих систем автоматизации 4](#_Toc136201928)

[1.2 Анализ разработки и проектирования веб приложений 5](#_Toc136201929)

[1.3 Анализ используемых технологий для реализации поставленной задачи 6](#_Toc136201930)

[1.4 Анализ инструментальных средств автоматизации разработки и тестирования 10](#_Toc136201931)

[1.5 Техническое задание для клиент-серверного программного продукта «Программный комплекс автоматизации обслуживания жилого фонда студенческого общежития» 15](#_Toc136201932)

[2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И АЛГОРИТМЫ 16](#_Toc136201933)

[2.1 Анализ предметной области 16](#_Toc136201934)

[2.2 Функциональная модель программного комплекса 16](#_Toc136201935)

[2.3 Информационная модель программного комплекса 19](#_Toc136201936)

[3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ КОМПЛЕКСА 25](#_Toc136201937)

[3.1 Архитектура приложения 25](#_Toc136201938)

[3.2 Уровень представления клиентской части приложения 26](#_Toc136201939)

[3.3 Уровень бизнес-логики приложения 39](#_Toc136201940)

[3.4 Уровень данных в приложении 44](#_Toc136201941)

[4 ТЕСТИРОВАНИЕ, ВЕРИФИКАЦИЯ И ВАЛИДАЦИЯ ПРОГРАММНОГО КОМПЛЕКСА 45](#_Toc136201942)

[4.1 Тестирование пользовательских форм 45](#_Toc136201943)

[4.2 Тестирование бизнес-логики 48](#_Toc136201944)

# **ВВЕДЕНИЕ**

Автоматизация процессов позволяет увеличить эффективность работы, сократить время и затраты на выполнение задач, снизить вероятность ошибок и улучшить качество услуг. Также автоматизация помогает оптимизировать использование ресурсов, повысить скорость реакции на изменения внешней среды и упростить процессы для пользователя.

Целью дипломной работы является создание программного комплекса автоматизации жилого фонда студенческого общежития. Данный программный комплекс будет предназначен для оптимизации управления ресурсами общежития, упрощения и автоматизации процессов, связанных с проживанием студентов и поддержанием работы всего жилого комплекса.

Задачами дипломной работы являются:

* изучение методик разработки клиент-серверных приложений на базе стека MERN;
* классификация ролей пользователей и их ролевые политики;
* изучение методов реализации серверной части для приложения;
* проектирование структуры приложения, базы данных для хранения информации, формирование пользовательских правил для доступа к ресурсам и функциям приложения;
* разработка программных модулей, обеспечивающих авторизацию и аутентификацию пользователей; работу с данными с помощью графического интерфейса;
* верификация и опытная эксплуатация разработанного программного обеспечения.

1. АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ АВТОМАТИЗАЦИИ ЖИЛОГО ФОНДА СТУДЕНЧЕСКОГО ОБЩЕЖИТИЯ
   1. Обзор существующих систем автоматизации

Многие университеты имеют свои личные жилые фонды (общежития), но не многие пытаются их автоматизировать, ведь в них протекают много различных и сложных процессов. Например, управление жилплощадью, распределение мест, контроль доступа, управление уборкой и ремонтом, а также учет платежей за проживание и услуги – все эти процессы могут быть оптимизированы и автоматизированы с помощью специальных программных решений. Такая автоматизация помогает университетам повысить эффективность управления общежитиями, снизить затраты и улучшить качество обслуживания студентов.

В большинстве случаем университеты предоставляют общую информацию об общежитии, сформированную в виде одной онлайн страницы. Например, на сайте университета БГПУ приведены следующие данные:

* адреса общежитий;
* фамилия, имя и отчество заведующего;
* фамилии, имена и отчества воспитателей и педагогов-организаторов,
* общее описание здания (количество мест, год постройки);
* наличие в общежитии общественных пространств (теннисные и тренажерные залы, актовый зал, кафе, прачечная, комната самоподготовки).

На сайте «Белорусского государственного университета» имеется информация, выраженная в виде рекомендаций и фотографий, например:

* какие бытовые приборы лучше всего взять с собой;
* как распланировать бюджет родителям.

Также существую другие средства автоматизация жилого фонда студенческого общежития. Эти системы включают в себя различные программные и аппаратные компоненты, такие как системы безопасности, системы управления доступом, системы учета и бронирования проживающих, системы мониторинга и управления энергопотреблением и т.д. Такие системы могут быть полезны для упрощения управления общежитием и улучшения комфорта проживания студентов. Например, системы управления доступом позволяют управлять доступом в здание и в отдельные комнаты, что обеспечивает безопасность и предотвращает несанкционированный доступ. Системы мониторинга и управления энергопотреблением позволяют управлять энергоресурсами и экономить деньги на коммунальных услугах.

Кроме того, системы учета и бронирования проживающих позволяют упростить процесс бронирования мест в общежитии и учета проживающих, что может значительно сократить время, затрачиваемое на административные процессы.

В целом, автоматизация студенческих общежитий может быть очень полезна для улучшения условий проживания студентов и оптимизации управления общежитием.

* 1. Анализ разработки и проектирования веб приложений

Разработка и проектирование веб-приложений – это процесс создания программного обеспечения, которое работает на сервере и обеспечивает взаимодействие с пользователем через веб-браузер. Веб-приложения могут быть использованы для различных целей, включая электронную коммерцию, социальные сети, онлайн-банкинг, различные средства автоматизации и т.д.

Согласно [1], для создания веб-приложения нужно выполнить несколько шагов. Сначала определяются требования к приложению, которые могут включать в себя функциональные и нефункциональные требования, дизайн интерфейса, спецификации базы данных и т.д. Затем происходит проектирование архитектуры приложения, разработка базы данных и программного кода, а также тестирование и развертывание приложения на сервер.

Веб-приложение – это программное обеспечение, которое работает через веб-браузер и взаимодействует с пользователем посредством веб-интерфейса. В общем случае, веб-приложение состоит из следующих компонентов:

1. Клиентская часть (frontend) – это часть веб-приложения, которая выполняется в браузере пользователя и отображает интерфейс. Она написана на языке HTML, CSS и JavaScript, и обеспечивает взаимодействие пользователя с приложением. Клиентская часть может включать в себя различные компоненты, такие как фреймворки, библиотеки и т.д.
2. Серверная часть (backend) – это часть веб-приложения, которая работает на сервере и обрабатывает запросы пользователя, взаимодействует с базой данных и обеспечивает доступ к ресурсам и функциям приложения. Серверная часть может быть написана на различных языках программирования, таких как PHP, Python, Java, Ruby и т.д.
3. База данных – это хранилище данных, которые используются в приложении. База данных может быть реляционной или нереляционной и хранить информацию о пользователях, продуктах, заказах, и т.д.
4. API (Application Programming Interface) – это интерфейс, который позволяет взаимодействовать между клиентской и серверной частями приложения. API может использовать различные протоколы, такие как HTTP или WebSocket, и форматы данных, такие как JSON или XML.
5. Дополнительные компоненты – веб-приложение может содержать и другие компоненты, такие как библиотеки, плагины, сервисы сторонних разработчиков и т.д.

Компоненты веб-приложения взаимодействуют между собой и обеспечивают полноценное функционирование приложения. Они позволяют пользователям работать с приложением, выполнять различные действия и получать результаты их работы.

Разработка веб-приложений может включать в себя использование различных языков программирования, фреймворков, библиотек, систем управления базами данных и других инструментов. Например, для создания веб-приложения может использоваться язык программирования JavaScript, фреймворк React, база данных MySQL и сервер Apache.

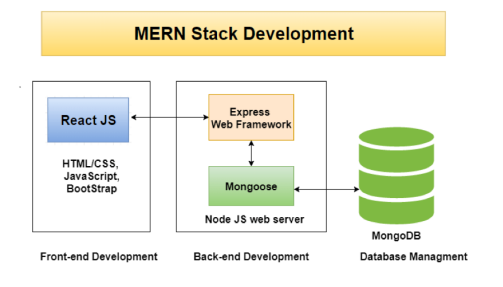
Работа веб-приложения обычно происходит следующим образом: пользователь запрашивает определенную страницу через веб-браузер, сервер обрабатывает запрос и отправляет ответ обратно в браузер в виде HTML-страницы. Эта страница может содержать данные из базы данных, такие как пользовательский профиль или список продуктов, а также интерактивные элементы, такие как кнопки и формы, которые позволяют пользователю взаимодействовать с приложением.

В целом, веб-приложения могут быть очень полезны для обеспечения взаимодействия пользователей с программным обеспечением через веб-браузер.

Разработка и проектирование веб-приложений требует определенных навыков и знаний в области программирования, веб-технологий, дизайна пользовательского интерфейса и баз данных. Для успешной разработки веб-приложений необходимо понимание требований к приложению и его пользователей, умение разрабатывать эффективную архитектуру, использовать соответствующие инструменты и библиотеки, а также тестировать и развертывать приложение на сервер. Важным аспектом является также поддержка и обновление приложения после его запуска, а также обеспечение безопасности при обработке пользовательских данных и защите от взломов.

* 1. Анализ используемых технологий для реализации поставленной задачи

Веб-приложения могут быть написаны на разных языках программирования и фреймворках, но одним из наиболее популярных стеков технологий для разработки веб-приложений является стек MERN. MERN – это аббревиатура, состоящая из четырех популярных инструментов для веб-разработки: MongoDB, Express, React и Node.js.



1. – Компоненты стека MERN

В основе этих фреймворков и библиотек находится язык программирования JavaScript.

JavaScript – мультипарадигменный (с одновременным использованием множества парадигм) язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript (стандарт ECMA-262). Его обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты:

* динамическая типизация;
* слабая типизация;
* автоматическое управление памятью;
* прототипное программирование;
* функции как объекты первого класса.

JavaScript включает в себя объектную модель браузера – браузер-специфичная часть языка, являющаяся прослойкой между ядром и объектной моделью документа. Основное предназначение объектной модели браузера — управление окнами браузера и обеспечение их взаимодействия. Каждое из окон браузера представляется объектом window, центральным объектом DOM. Объектная модель браузера на данный момент не стандартизирована, однако спецификация находится в разработке WHATWG и W3C.

В JavaScript используется в AJAX, популярном подходе к построению интерактивных пользовательских интерфейсов веб-приложений, заключающемся в «фоновом» асинхронном обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью и интерфейс веб-приложения становится быстрее, чем это происходит при традиционном подходе (без применения AJAX) [2].

MongoDB – это документо-ориентированная NoSQL база данных, которая используется для хранения, управления и обработки больших объемов данных. Она позволяет хранить данные в формате документов BSON (Binary JSON), которые имеют структуру, аналогичную формату JSON, но могут содержать бинарные данные, даты и другие типы данных.

MongoDB разрабатывалась с учетом масштабируемости и производительности, что делает ее особенно привлекательной для проектов, которые требуют быстрой и эффективной обработки больших объемов данных. Она имеет множество функций, которые делают ее полезной для многих типов приложений, включая:

1. Гибкий документо-ориентированный подход: MongoDB предоставляет гибкую модель документо-ориентированной базы данных, что позволяет хранить данные любого формата, включая структурированные, полуструктурированные и неструктурированные данные;
2. Высокая доступность: MongoDB предоставляет высокую доступность благодаря встроенному механизму репликации, который позволяет хранить несколько копий данных на разных серверах. Если один сервер не работает, MongoDB может быстро переключиться на другой сервер и продолжить работу;
3. Масштабируемость: MongoDB легко масштабируется, что позволяет ей обрабатывать большие объемы данных. MongoDB поддерживает масштабирование горизонтальным и вертикальным способами;
4. Мощный язык запросов: MongoDB предоставляет мощный язык запросов, который позволяет быстро и эффективно извлекать данные из базы данных;
5. Индексы: MongoDB поддерживает множество типов индексов, которые позволяют ускорить поиск и фильтрацию данных;

MongoDB широко используется в различных областях, включая социальные сети, онлайн-магазины, приложения для анализа данных и многие другие. Ее популярность объясняется ее гибкостью, производительностью и масштабируемостью [3].

Express.js – это легковесный фреймворк для Node.js, который используется для разработки веб-приложений и API. Он предоставляет удобный и гибкий механизм для обработки запросов и ответов, маршрутизации и создания модульной структуры приложений.

Основные преимущества Express.js:

1. Удобство: Express.js облегчает разработку веб-приложений благодаря простому и интуитивно понятному API. Он позволяет быстро создавать маршруты, обрабатывать запросы и ответы, а также работать с различными middleware-пакетами;
2. Гибкость: Express.js позволяет разработчикам создавать приложения с различными функциональными возможностями. Он не навязывает строгую структуру приложения, что дает возможность гибко настраивать его под конкретные задачи;
3. Масштабируемость: Express.js позволяет легко масштабировать приложения, что особенно важно для больших и сложных проектов. Он поддерживает работу с кластерами и позволяет распределять нагрузку между несколькими серверами;
4. Поддержка middleware: Express.js предоставляет широкие возможности для работы с middleware, что позволяет улучшать функциональность приложения и повышать его безопасность. С помощью middleware можно добавлять авторизацию, обработку ошибок, логгирование и многое другое;
5. Большое сообщество: Express.js имеет большое сообщество разработчиков, которые создают и поддерживают множество пакетов и расширений. Это позволяет разработчикам быстро решать задачи и получать поддержку при возникновении проблем.

Express.js используется для создания различных типов приложений, включая API, веб-серверы, приложения для обработки данных и многие другие. Он позволяет быстро создавать и масштабировать приложения, обеспечивая при этом гибкость и удобство разработки [4].

React – это библиотека JavaScript, разработанная Facebook, которая используется для создания пользовательских интерфейсов. React использует декларативный подход для описания компонентов пользовательского интерфейса, что делает его более простым и понятным для разработчиков. Он позволяет создавать переиспользуемые компоненты, которые могут быть легко использованы для создания сложных пользовательских интерфейсов.

Основные преимущества React:

1. Декларативный подход: React использует декларативный подход для описания пользовательского интерфейса, что делает его более понятным для разработчиков. Он позволяет описывать, как должен выглядеть интерфейс, а не как его создать;
2. Переиспользуемые компоненты: React позволяет создавать переиспользуемые компоненты, которые могут быть легко использованы для создания сложных пользовательских интерфейсов. Это сокращает время разработки и улучшает качество кода;
3. Эффективный: React использует виртуальный DOM, который позволяет изменять только те элементы, которые действительно изменились. Это уменьшает количество дорогостоящих операций, связанных с обновлением интерфейса, и улучшает производительность приложения;
4. Большое сообщество: React имеет большое сообщество разработчиков, которые создают и поддерживают множество пакетов и расширений. Это позволяет разработчикам быстро решать задачи и получать поддержку при возникновении проблем;
5. Простота: React является относительно простым и понятным инструментом для создания пользовательских интерфейсов. Это делает его доступным для начинающих разработчиков и уменьшает время на обучение;

React используется для создания интерактивных пользовательских интерфейсов, включая веб-приложения, мобильные приложения, игры и многое другое. Он позволяет создавать переиспользуемые компоненты, которые могут быть легко использованы для создания сложных пользовательских интерфейсов [5].

Node.js – это среда выполнения JavaScript на стороне сервера, которая позволяет разрабатывать высокопроизводительные и масштабируемые веб-приложения. Она основана на движке V8, разработанном Google для браузера Chrome, и позволяет использовать JavaScript для создания приложений на серверной стороне.

Основные преимущества Node.js:

1. Высокая производительность: Node.js основан на движке V8, который обеспечивает быстрое выполнение JavaScript. Node.js также позволяет использовать асинхронное программирование, что улучшает производительность приложений;
2. Масштабируемость: Node.js позволяет создавать масштабируемые приложения с помощью механизма обработки запросов в нескольких потоках. Это позволяет распределять нагрузку на несколько серверов и обеспечивать высокую доступность приложения;
3. Широкие возможности: Node.js имеет большое количество библиотек и модулей, которые позволяют упростить разработку и расширить функциональность приложения. Это позволяет создавать приложения для различных сфер, включая веб-приложения, мобильные приложения, игры и многое другое;
4. Единый язык: Node.js использует JavaScript как единый язык для программирования на серверной и клиентской стороне, что упрощает разработку и повышает эффективность работы разработчика;
5. Активное сообщество: Node.js имеет большое сообщество разработчиков, которые создают и поддерживают множество пакетов и расширений. Это позволяет разработчикам быстро решать задачи и получать поддержку при возникновении проблем;

Node.js используется для создания различных приложений на серверной стороне, включая веб-приложения, микросервисы, API и многое другое. Он позволяет разработчикам создавать высокопроизводительные и масштабируемые приложения с использованием JavaScript на стороне сервера, что упрощает разработку и повышает эффективность работы разработчика.

* 1. Анализ инструментальных средств автоматизации разработки и тестирования

Для создания программного комплекса по обслуживанию жилого фонда студенческого общежития будет использована среда разработки Webshtorm от компании JetBrains. Для визуализации базы данных MongoDb лучше всего подходит приложение MongoDb Compass. Хранить исходный код только на локальном компьютере плохая практика, поэтому будет создан удаленный репозиторий на GitHub. Для постройки различных диаграмм, которые позволят упростить разработку, а также предоставят полное понимание работы приложения, будет использоваться StarUml.

Рассмотрим подробнее каждый из этих инструментов.

Webshtorm – это интегрированная среда разработки (IDE), которая предоставляет обширный функционал для разработки веб-приложений. Среда разработки позволяет работать с различными языками программирования, включая JavaScript, TypeScript, HTML, CSS, Node.js, Angular, React и другие. Webshtorm включает в себя встроенные инструменты отладки, систему автодополнения кода, автоматическую проверку ошибок, систему контроля версий, анализаторы кода и многие другие полезные функции.

Webshtorm обладает многоплатформенностью, что позволяет разрабатывать на разных операционных системах, включая Windows, macOS и Linux. Среда разработки также имеет мощную систему плагинов, которая позволяет расширять функционал IDE, добавляя поддержку новых языков программирования и инструментов. Она также имеет множество инструментов для работы с базами данных, включая поддержку MongoDB, MySQL, PostgreSQL, Oracle и других, обеспечивает интеграцию с браузерами для отладки веб-приложений в режиме реального времени.

Одним из основных преимуществ Webshtorm является его эффективность и производительность. Среда разработки использует многопоточную архитектуру и оптимизированный механизм работы с памятью, что обеспечивает быстродействие и позволяет работать с большими проектами. Кроме того, Webshtorm имеет обширную документацию и активное сообщество пользователей, которые создают полезные плагины, советы и обучающие ресурсы, что делает процесс разработки еще более комфортным и эффективным.

Webshtorm – это мощная среда разработки, которая облегчает и ускоряет процесс создания высококачественных веб-приложений, идеально подходящая для опытных и начинающих разработчиков [6].

MongoDB Compass – это визуальный интерфейс для работы с базами данных MongoDB. Этот инструмент позволяет разработчикам и администраторам баз данных MongoDB легко просматривать, анализировать и манипулировать данными. Он предоставляет графический интерфейс для создания, редактирования и удаления коллекций, документов и индексов базы данных MongoDB. Он также обеспечивает визуализацию структуры коллекций, что помогает быстро понять структуру данных.

MongoDB Compass имеет встроенный механизм запросов, который позволяет быстро и легко создавать и выполнять запросы к базе данных. Он также обеспечивает удобный механизм фильтрации и сортировки данных, а также поддержку агрегационных запросов.

Среди других возможностей MongoDB Compass – поддержка графического интерфейса для выполнения команд в MongoDB, создание и сохранение запросов для повторного использования, анализ статистики запросов и многое другое.

MongoDB Compass также обеспечивает удобный механизм подключения к серверу базы данных MongoDB, что позволяет работать с удаленными базами данных. Он также поддерживает механизм аутентификации и авторизации, что обеспечивает безопасность при работе с базами данных.

В целом, MongoDB Compass – это удобный и мощный инструмент для работы с базами данных MongoDB, который обеспечивает широкий спектр возможностей для работы с данными, а также удобный интерфейс для управления базой данных. Этот инструмент идеально подходит для разработчиков и администраторов баз данных MongoDB, которые хотят упростить и ускорить работу с данными [7].

GitHub – это веб-сервис для хранения и совместной работы над Git-репозиториями. Это платформа, которая позволяет разработчикам хранить и совместно работать над кодом, отслеживать ошибки, создавать ветки, а также просматривать и редактировать код, управлять версиями, изменениями и запросами на слияние.

Репозиторий в GitHub – это хранилище для кода и других файлов, которые могут быть загружены и управляемы в Git. Репозиторий в GitHub позволяет хранить и управлять кодом в облаке, а также предоставляет множество функций для работы с кодом, включая возможность комментирования кода, управления задачами и многое другое.

Если хранить код только на локальном компьютере, то это может привести к потере данных в случае сбоя жесткого диска или других проблем с компьютером. Кроме того, хранение кода на локальном компьютере не обеспечивает возможность совместной работы и синхронизации изменений между различными разработчиками.

Таким образом, GitHub – это мощный и удобный инструмент для хранения и совместной работы над кодом. Он предоставляет широкий спектр возможностей для управления кодом, удобный интерфейс для просмотра и редактирования кода, а также возможность совместной работы и синхронизации изменений между различными разработчиками.

StarUML – это приложение для создания UML-диаграмм, которое позволяет разработчикам создавать модели проектов, планировать архитектуру и дизайн приложений. StarUML имеет графический интерфейс пользователя, который предоставляет множество инструментов для создания и редактирования диаграмм, включая диаграммы классов, диаграммы последовательностей, диаграммы состояний, диаграммы активностей и многое другое.

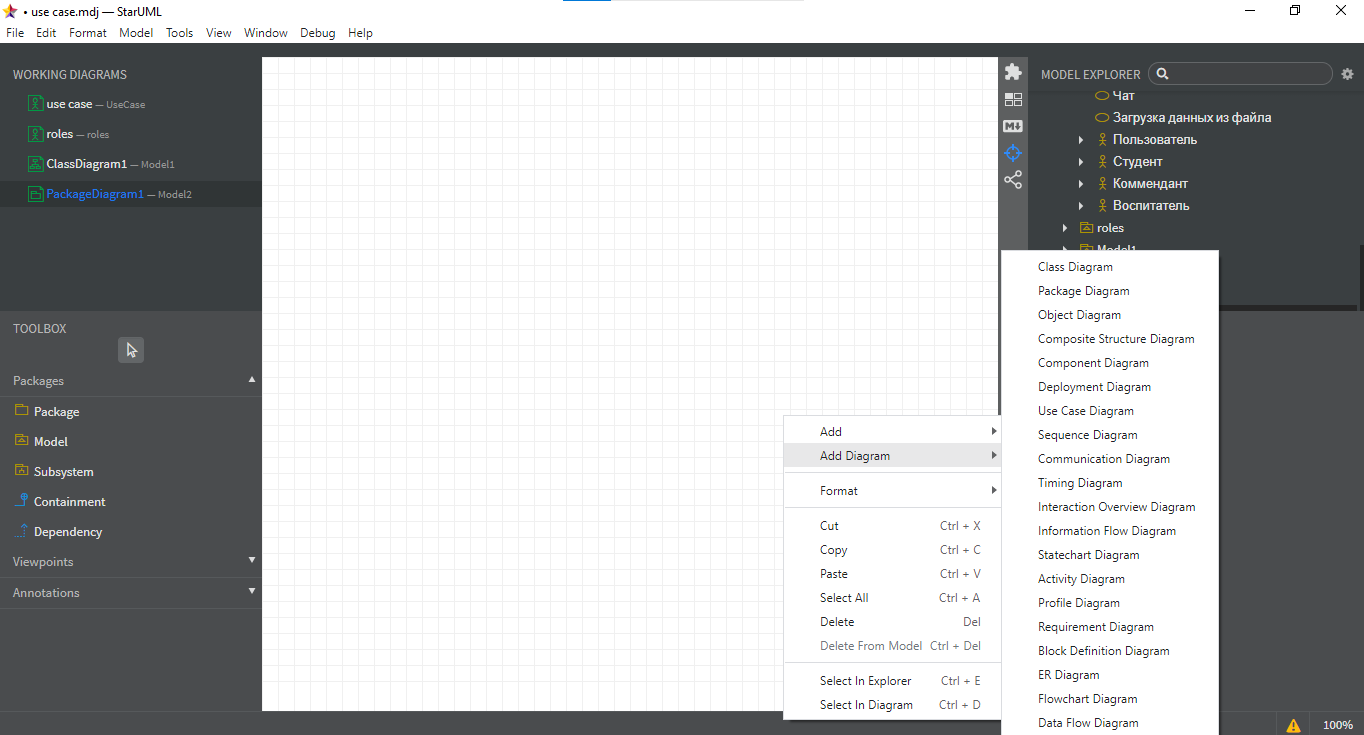


Рисунок 1.2 – Интерфейс главного окна StarUml

В StarUML можно создавать различные типы диаграмм, включая:

1. Диаграммы классов – используются для описания структуры классов и их отношений;
2. Диаграммы последовательностей – используются для описания взаимодействия между объектами и процессов, происходящих во времени;
3. Диаграммы состояний – используются для описания жизненного цикла объекта и его состояний;
4. Диаграммы компонентов – используются для описания структуры и отношений между компонентами системы;
5. Диаграммы развертывания – используются для описания физического размещения компонентов и системы в целом;
6. Диаграммы активностей – используются для описания последовательности действий и процессов в системе.

На рисунке 1.2 в виде выпадающего меню представлен весь список возможных диаграмм.

StarUML поддерживает различные языки моделирования, включая UML 2.x, SysML, ERD, BPMN, SoaML и другие. Он также позволяет пользователям импортировать и экспортировать диаграммы в различных форматах файлов, включая XMI, SVG, PDF, PNG, JPEG и другие. Данное приложение имеет множество функций, которые облегчают создание и редактирование диаграмм, такие как автоматическое выравнивание элементов, возможность группировки элементов, подсветка синтаксиса, подсказки и многое другое. Кроме того, StarUML позволяет пользователю создавать свои собственные элементы, шаблоны и плагины, чтобы улучшить функциональность и адаптировать инструмент под свои потребности.

StarUML – это мощный и удобный инструмент для создания UML-диаграмм, который предоставляет широкий спектр возможностей для создания и редактирования диаграмм, а также импорта и экспорта диаграмм в различных форматах файлов. Он также имеет множество функций, которые облегчают создание и редактирование диаграмм, и может быть расширен за счет пользовательских элементов, шаблонов и плагинов [8].

Postman – это инструмент для тестирования API, который позволяет разработчикам быстро и удобно отправлять запросы к API, тестировать их и анализировать ответы.

С помощью Postman можно отправлять запросы на сервер и получать ответы в разных форматах, таких как JSON, XML, HTML и другие. Также в Postman есть множество функциональных возможностей, таких как автоматическое генерирование документации, управление авторизацией, создание и выполнение тестовых сценариев и другие. Он поддерживает различные типы запросов, включая GET, POST, PUT, DELETE и многие другие. Вы можете отправлять запросы с параметрами, заголовками и телом запроса в формате JSON или форм-данных.

Один из самых удобных и полезных аспектов Postman – это коллекции запросов, которые можно создавать и организовывать для более эффективного управления тестированием API. Коллекции могут быть экспортированы и импортированы в различных форматах, таких как JSON и XML.

Postman имеет множество преимуществ, которые делают его популярным среди разработчиков, вот некоторые из них:

1. Удобный интерфейс. Интерфейс Postman интуитивно понятен и удобен в использовании. Для отправки запроса не нужно писать код, все нужные функции находятся в графическом интерфейсе.
2. Простота в использовании. Postman не требует от разработчика особых навыков, чтобы начать использовать его. Для создания запросов нужно просто заполнить соответствующие поля.
3. Расширяемость. Postman можно легко расширять, используя плагины. Например, с помощью плагина можно добавить поддержку авторизации на основе OAuth или создать пользовательский набор инструментов для тестирования API.
4. Автоматизация. Postman позволяет автоматизировать тестирование API. Это особенно полезно при наличии большого количества запросов, которые нужно тестировать.
5. Множество функций. Postman имеет множество функций, таких как создание коллекций, генерация документации и выполнение тестовых сценариев, что делает его полезным инструментом для разработки и тестирования приложений.
6. Бесплатность. Postman имеет бесплатную версию, которая покрывает большинство потребностей разработчиков, а также платную версию с дополнительными функциями.

Postman является полезным и удобным инструментом для тестирования API, который может существенно ускорить процесс разработки приложений [9].

* 1. Техническое задание для клиент-серверного программного продукта «Программный комплекс автоматизации обслуживания жилого фонда студенческого общежития»

Цель разработки: разработать приложение, предназначенное для автоматизации обслуживания жилого фонда студенческого общежития.

Данный программного комплекс предназначается для студентов и работников студенческого общежития.

Приложение позволит коменданту следующее:

* вести учет технического оборудования, которое ввозят студенты для личного пользования;
* вести учет количества свободных и занятых мест с возможность фильтрации за год или за месяц;
* для заполнения списка студентов комендант сможет импортировать файл формата Excel в базу данных;
* мониторинг оплаты за проживание.

Для студентов будет реализовано:

* личный кабинет, где они смогут следить за текущими новостями и событиями;
* чат с воспитателем;
* возможность создавать заявки на починку бытового оборудования.

Для воспитателей приложение предусматривает следующие функции:

* создание и ведение ленты новостей;
* создание и ведение ленты событий;
* начисление баллов студентам;
* создание чата со студентом или общей беседы;
* оставлять замечания в личном кабинете студента.

Приложение должно иметь следующую структуру и функциональность:

* работать как веб-приложение основанное на REST архитектуре;
* являться кроссплатформенным;
* являться кроссбраузерным;
* иметь принцип работы, аналогичный веб-приложениям (реализовывать REST архитектуру, быть доступным посредством веб-браузера, иметь пользовательский интерфейс);

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И АЛГОРИТМЫ
   1. Анализ предметной области

Анализ предметной области «Автоматизация обслуживания жилого фонда студенческого общежития» позволяет выделить следующие основные аспекты:

1. Учет и распределение комнат. Для автоматизации данного процесса необходимо иметь возможность вести учет свободных и занятых комнат, а также осуществлять распределение студентов в свободные комнаты.
2. Регистрация и учет студентов. Необходимо иметь возможность регистрировать студентов и вести учет их проживания в общежитии, а также осуществлять контроль за своевременной оплатой за проживание.
3. Предоставление различной информации. Необходимо иметь возможность создавать и просматривать новостные ленты для того.
4. Связь воспитателей и студентов. Необходимо иметь возможность связываться с студентом для решения вопросов, связанных с проживанием в общежитии.

Для автоматизации этих процессов может быть использована специализированная информационная система, разработанная с учетом конкретных потребностей студенческого общежития. Она будет включать в себя базу данных для хранения информации о студентах и комнатах, модули для управления ресурсами и контроля доступа, а также возможности для сотрудников общежития добавлять и изменять информацию о конкретном студенте.

Таким образом, разработка автоматизированной системы позволит упростить и ускорить процессы управления жилым фондом студенческого общежития, повысить эффективность использования ресурсов и улучшить качество обслуживания студентов. Благодаря использованию базы данных, все данные будут храниться в одном месте и сотрудники общежития смогут быстро получать необходимую информацию о студентах и комнатах. А модули управления ресурсами и контроля доступа позволят более точно контролировать использование ресурсов и обеспечить безопасность в общежитии. Разработка системы также может способствовать улучшению учебного процесса, так как облегчит жизнь студентам и позволит им сконцентрироваться на учебе, а не на решении бытовых вопросов.

* 1. Функциональная модель программного комплекса

Для того, чтобы создать качественное приложение, нужно четко понимать, какие роли у нас будут и какой функционал им будет доступен. Для этого нужно знать, какие объекты попадают в предметную область проектируемой системы и какие логические связи между ними существуют. Для формирования такого понимания используются логические модели предметной области. Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области. Для стабильной работы программного обеспечения необходимо чёткое распределение на роли, на основе которых будут формироваться функции взаимодействия со внутренней средой приложения.

Актер – множество логически связанных ролей в UML, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Актером может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности.

Прецеденты представляют действия, выполняемые системой в интересах актеров. Проще говоря, прецедент – это описание последовательности действий (или нескольких последовательностей), которые выполняются системой и производят для отдельного актера видимый результат. Один актер может использовать несколько элементов прецедентов, и наоборот, один прецедент может иметь несколько актеров, использующих его. Каждый прецедент задает определенный путь использования системы. Набор всех прецедентов определяет полные функциональные возможности системы.

Приложение должно реализовывать три роли: комендант, воспитатель, студент.

Основные функции для роли «Комендант»:

* учет личного технического оборудования;
* учет количества свободных и занятых мест;
* возможность импортировать данные о студентах формата Excel;
* производить мониторинг оплаты за проживание.

Основные функции для роли «Студент»:

* просмотр новостей и событий;
* чат с воспитателем;
* создавать заявки на починку бытового оборудования.

Основные функции для роли «Воспитатель»:

* создание и ведение ленты новостей и событий;
* начисление баллов студентам;
* создание чата со студентом;
* создание общей беседы;
* возможность оставлять замечания в личном кабинете студента

В качестве среды для разработки и отображения функциональной структуры программы будет использована StarUml. На рисунке 2.2 представлена Use Case диаграмма приложения.



1. – Диаграмма прецедентов и актеров

Далее приведено описание прецедентов для роли «Комендант».

Прецендент «Учет личного техники» предназначен для просмотра и управления личным техническим оборудованием студента.

Прецендент «Загрузка данных из файла» предназначен для выгрузки личных данных студентов в базу данных.

Прецендент «Мониторинг оплаты общежития» предназначен для просмотра задолженности перед общежитием, а также для информирования студента.

Прецендент «Просмотр занятых и свободных мест» предназначен для мониторинга занятости комнат.

Далее приведено описание прецендентов для роли «Воспитатель».

Прецендент «Чат» предназначен для общения с отдельным студентом или с неким количеством.

Прецендент «Просмотр списка студентов» предназначен для просмотра информации о студентах в целом или более подробно об отдельном студенте.

Прецендент «Начисление баллов» предназначен для начисления баллов отдельному студенту.

Прецендент «Оставлять замечания». Данный прецендент необходим для того, чтобы воспитатель мог оповестить студента о каком-то совершенном нарушении в его личном кабинете.

Прецендент «Ведение ленты новостей» предназначен для просмотра и создания новых новостей.

Прецендент «Ведение ленты мероприятий» предназначен для просмотра и создания с целю оповещения новых событий.

Диаграмма прецендентов и актеров отображает функциональность системы с точки зрения пользователей. Она показывает взаимодействие между пользователем и системой в рамках конкретной задачи или сценария использования.

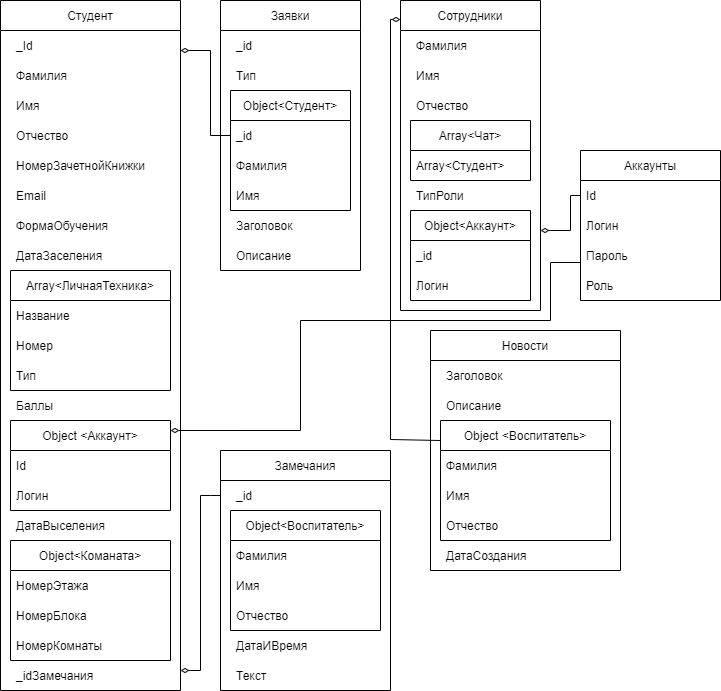
* 1. Информационная модель программного комплекса

На основе анализа предметной области определим набор коллекций и их свойства. В MongoDB коллекция представляет собой группу документов, которые хранятся в базе данных. Коллекции в MongoDB являются аналогом таблиц в реляционных базах данных. Они содержат документы в формате JSON, которые могут иметь различную структуру и не обязательно должны иметь одинаковые поля.

Для отображения информационной модели разрабатываемого продукта были выделены следующие коллекции:

* коллекция «Студент». Хранит в себе информацию о студентах, а также о комнате, где он живет;
* коллекция «Аккаунт». Хранит в себе данные для входа в личный аккаунт пользователя;
* коллекция «Сотрудники». Хранит в себе личные данные сотрудников общежития;
* коллекция «Новость». Хранит в себе весь список новостей;
* коллекция «Чат». Хранит список всех чатов.
* коллекция «Заявки». Хранит информацию о заявке студента на починку бытового оборудования.

На картинке 2.2 приведена схема базы данных.



1. – Схема базы данных

В таблицах 2.1 – 2.6 представлено подробное описание полей в коллекциях.

Таблица 2.1 – Коллекция «Студент»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Обязательное (required) | Уникальное (unique) |
| \_id | ObjectId | Да | Да |
| firstName | string | Да | Нет |

Продолжение таблицы 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Обязательное (required) | Уникальное (unique) |
| secondName | string | Да | Нет |
| middleName | string | Да | Нет |
| numberTest | string | Да | Да |
| email | string | Нет | Да |
| formEducation | union ("платное" | "бесплатное") | Да | Нет |
| dateEntry | string | Да | Нет |
| balls | number | Да | Нет |
| privateTechs | Array<Object> | Нет | Нет |
| privateTech.model | string | Нет | Нет |
| privateTech.number | string | Нет | Нет |
| privateTech.type | string | Нет | Нет |
| room | Object | Да | Нет |
| room.floor | number | Да | Нет |
| room.block | number | Да | Нет |
| room.apartament | number | Да | Нет |
| remarks | Array<Object> | Нет | Нет |
| remark.dateAndTime | string | Да | Нет |
| remark.header | string | Да | Нет |
| remark.status | string | Да | Нет |
| remark.text | string | Да | Нет |
| remark.mentor | Object | Да | Нет |
| remark.mentor.firstName | string | Да | Нет |
| remark.mentor.secondName | string | Да | Нет |
| remark.mentor.middleName | string | Да | Нет |
| account | Object | Да | Да |
| account.login | string | Да | Да |

В коллекции, описанной в таблице 2.1 поле formEducation имет специальный тип union, предоставляемый TypeScript. Данный тип представляет объединение нескольких строк, которые разрешено присваивать данной переменной.

Таблица 2.2 – Коллекция «Новости»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Обязательное (required) | Уникальное (unique) |
| \_id | ObjectId | Да | Да |
| header | string | Да | Нет |
| description | string | Да | Нет |
| mentor | Object | Да | Нет |
| mentor.firstName | string | Да | Нет |
| mentor.secondName | string | Да | Нет |
| mentor.middelName | string | Да | Нет |
| dateCreate | date | Да | Нет |

Как видно из таблицы 2.2, коллекция «Новости» хранит в себе данные воспитателя. Это необходимо для того, чтобы не осуществлять поиск по коллекции «Сотрудники» для установления данных создателя документа, что позволит нам сэкономить время загрузки данных.

Таблица 2.3 – Коллекция «Аккаунты»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Обязательное (required) | Уникальное (unique) |
| \_id | ObjectId | Да | Да |
| login | string | Да | Да |
| password | string | Да | Нет |

В таблице 2.3 приведено описание полей коллекции «Аккаунты». Данная коллекция нужна для быстрого поиска пользователя в системе во время его авторизации.

Таблица 2.4 – Коллекция «Чат»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Обязательное (required) | Уникальное (unique) |
| \_id | ObjectId | Да | Да |
| name | String | Да | Нет |
| messages | Array<Object > | Да | Нет |
| messages.who | Object | Да | Нет |
| messages.who.firstName | String | Да | Нет |
| messages.who.secondName | String | Да | Нет |
| messages.who.middleName | String | Да | Нет |
| messages.when | Date | Да | Нет |
| messages.message | string | Да | Нет |

Таблица 2.5 – Коллекция «Сотрудники»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Обязательное (required) | Уникальное (unique) |
| \_id | ObjectId | Да | Да |
| firstName | String | Да | Нет |
| secondName | string | Да | Нет |
| middleName | String | Да | Нет |
| Role | Union ("mentor" | "main") | Да | Нет |
| chats | Array<ObjectId> | Нет | Нет |

В таблице 2.5 описаны поля для коллекции «Сотрудники». Все поля, кроме chats, обязательные. Поле chats хранит массив типа ObjectId. Данный массив заполняется id из коллекции «Чат», описанной в таблице 2.4.

Таблица 2.6 – Коллекция «Заявки»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Обязательное (required) | Уникальное (unique) |
| \_id | ObjectId | Да | Да |
| type | String | Да | Нет |
| student | Object | Да | Нет |
| student.firstName | string | Да | Нет |

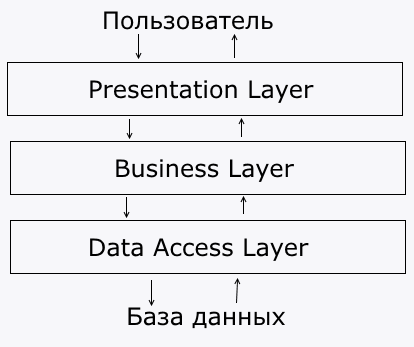
Продолжение таблицы 2.6

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Обязательное (required) | Уникальное (unique) |
| student.secondName | String | Да | Нет |
| student.room | number | Да | Нет |
| student.room.floor | Number | Да | Нет |
| student.room.block | Number | Да | Нет |
| student.room.apartament | Number | Да | Нет |
| header | String | Да | Нет |
| description | string | Да | Нет |

Как можно заметить, каждая коллекция в MongoDB может иметь некоторые поля, которые могут быть обязательными или необязательными. Кроме того, поля могут быть уникальными или неуникальными в пределах коллекции. Поле, которое является уникальным, не позволяет вставлять документ с таким же значением, как уже имеющийся в коллекции. Поле, которое является обязательным, требует, чтобы его значение было указано при вставке документа. Если обязательное поле не указано, то операция вставки завершится неудачей. Коллекция также может иметь индексы, которые помогают ускорить выполнение запросов к базе данных. В целом, использование MongoDB позволяет эффективно хранить и управлять данными, предоставляя удобный интерфейс для доступа к ним.

1. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ КОМПЛЕКСА
   1. Архитектура приложения

Приложение будет использовать трёхуровневую архитектуру (рисунок 2.1). Трехуровневая архитектура приложения – это подход к проектированию программного обеспечения, в котором приложение разбивается на три слоя: представление (presentation layer), бизнес-логика (business logic layer) и уровень доступа к данным (data access layer).



1. Схема архитектуры приложения

Трехуровневая архитектура приложения на примере стека MERN может быть описана следующим образом:

* уровень представления (Presentation Layer) – в MERN-стеке уровень представления представлен React-фреймворком, который используется для разработки клиентской части приложения. React позволяет создавать переиспользуемые компоненты, которые управляют отображением данных на странице;
* уровень бизнес-логики (Application Layer) – этот уровень обрабатывает бизнес-логику приложения. В MERN-стеке этот уровень реализован на Node.js и Express.js. Node.js – это серверная платформа, которая позволяет запускать JavaScript на стороне сервера, а Express.js – это веб-фреймворк для Node.js, который предоставляет удобный API для работы с запросами и ответами;
* уровень доступа к данным (Data Access Layer) – этот уровень отвечает за доступ к базе данных. В MERN-стеке в качестве базы данных используется MongoDB, которая хранит данные в формате документов JSON. Для работы с MongoDB в MERN-стеке используется Mongoose, которая предоставляет удобный API для создания моделей данных и выполнения запросов к базе данных.

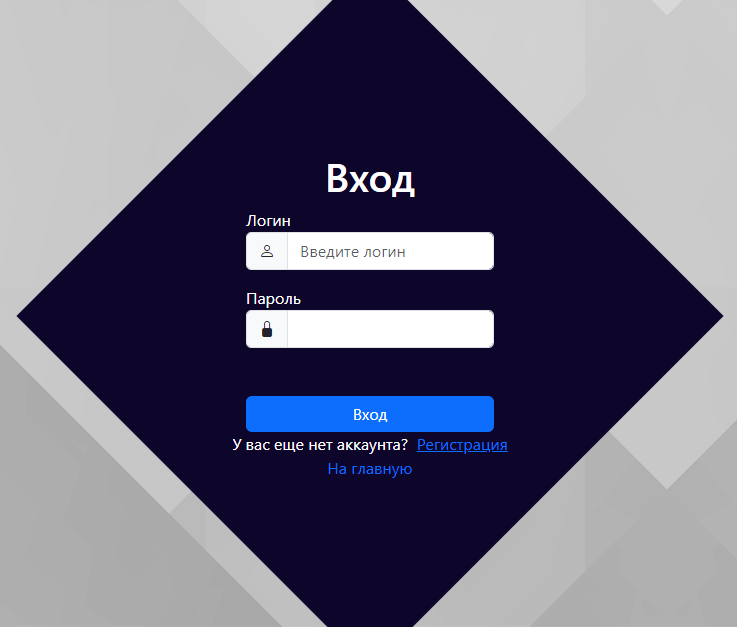
Таким образом, трехуровневая архитектура приложения разделяет приложение на три основных уровня: представление, бизнес-логику и уровень доступа к данным. Это позволяет разработчикам легче поддерживать и расширять приложение, а также упрощает его тестирование и развертывание.

* 1. Уровень представления клиентской части приложения

Уровень представления содержит следующие страницы:

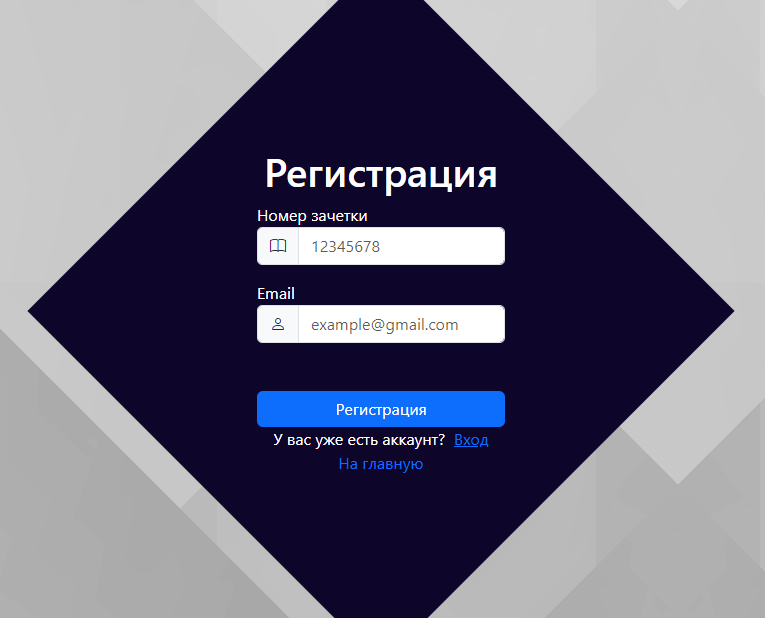
* Login – страница для входа в приложение;
* Registration – страница для регистрации студента в системе;
* MainStudent – главная страница студента;
* ContainerTech – данная страница предназначена для просмотра личной техники студентом;
* PayHostel – страница для учета оплаты общежития;
* Claim – страница для просмотра замечаний;
* NewsPanel – страница для просмотра списка новостей;
* BallsInfo – страница для просмотра истории начисления баллов;
* FullNews – страница для подробного просмотра новости;
* EventPanel – страница для просмотра событий;
* Chat – страница, которая позволяет вести беседу между воспитателем и студентом;
* EmployeeList – страница для просмотра списка воспитателей;
* FullMentor – страница для просмотра подробной информации о воспитателе
* CreateRepair – страница для создания студентом заявки на починку бытового оборудования
* MainMentor – главная страница воспитателя;
* StudentsList – страница, на которой находится таблица с перечислением краткой информации о студентах
* FullStudent – страница для просмотра полной информации о студенте
* CreateClaim – страница для создания замечания для определенного студента
* ChangeBalls – страница для начисления воспитателем баллов студенту;
* NewNewsForm – страница с формой для создания новости;
* CreateEvent – страница с формой для созданий нового события;
* CreateChat – страница для создания нового чата со студентами;
* MainAdmin – главная страница для коменданта;
* AddStudent – страница с формой для добавления нового студента;
* RepairView –страница, на которой комендант может просмотреть входящие заявки на починку бытового оборудования
* CreateTechForm –страница с формой для добавления новой техники для определенного студента
* ImportStudents – страница, на которой комендант может импортировать данный студентов из файла;
* ViewPayHostel – страница для просмотра отчетов об оплате за общежитие;
* CreateMentorForm – страница с формой для регистрации нового воспитателя;
* AccPlaces – страница, на которой отображенный занятые места;
* FreePlaces – страница, на которой отображены свободные места;
* Reports –страница, на которой комендант может запросить различные отчеты.

Страница Login (рисунок 3.2) является одной из начальных страниц, которую должны заполнить все роли при первоначальном запуске программы. Данная форма содержит два поля: логин и пароль. Все поля являются обязательными. В случае, если студент еще не зарегистрирован в системе, ему доступны ссылки «Регистрация» и «На главную».



1. Вид формы входа

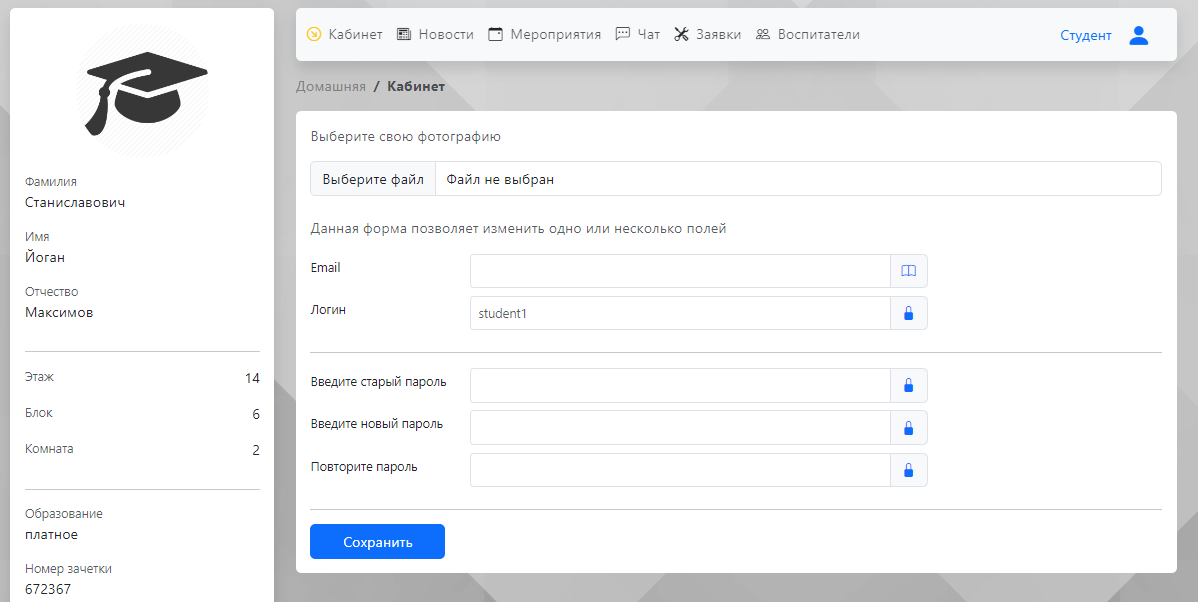
Страница Registration (рисунок 3.3) предназначена для регистрации студента в системе. На данной страницы представлена форма со следующими полями: номер зачетки и email. Все поля являются обязательными для заполнения.



1. Страница регистрации студента

После заполнения всех полей и нажатия на кнопку «Регистрация», студент получит уведомление о том, что заранее сгенерированные системой логин и пароль были высланы ему на почту. Эти данные студент должен ввести в форму Login для успешного входа в приложение.

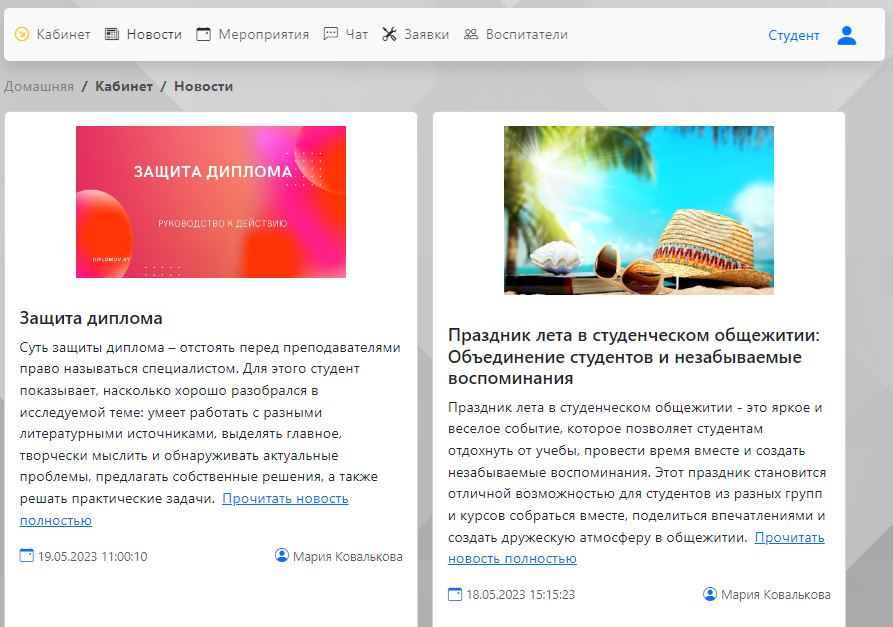
После успешной авторизации студента откроется страница MainStudent (рисунок 3.4).



1. Личный кабинет студента

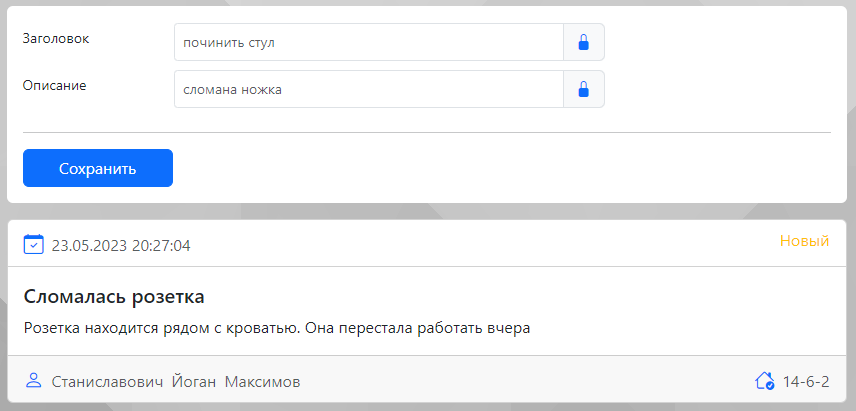
В левой части личного кабинета находится панель с информацией: фамилия, имя, отчество, этаж, блок, комната, форма образования, номер зачетной книжки. В верхней части расположено меню и навигационная панель. По центру расположена форма для изменения личных данных. Данная форма является универсальной, так как позволяет изменить только необходимые данные без заполнения всей формы. После заполнение хотя бы одного поля и нажатия на кнопку «Сохранить», данные отправятся на сервер, где пройдут окончательную проверку. После успешного или не успешного сохранение данных в базе данных, пользователь получит соответствующее уведомление.

Страница NewsPanel представляет ленту новостей (рисунок 3.5). Данная лента новостей одинаковая для роли «Студент» и «Воспитатель». В каждой новости имеется ссылка «Прочитать новость полностью», которая откроет новость на новой странице с более подробным содержанием.



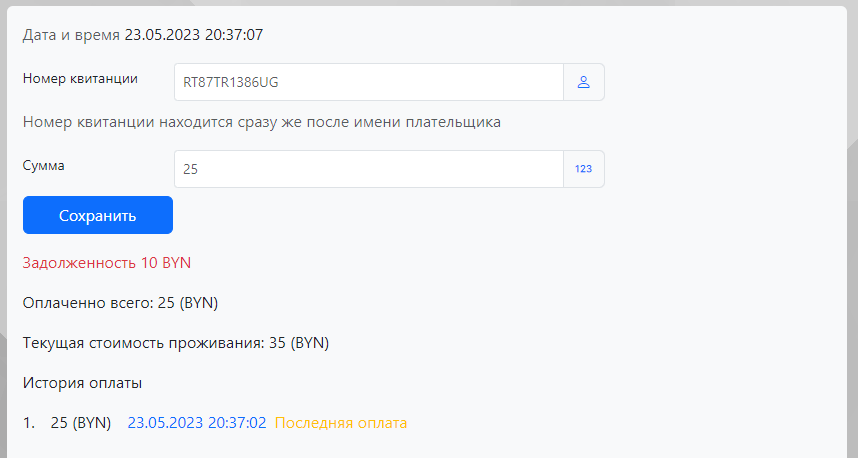
1. Страница с новостями

Страница CreateRepair (рисунок 3.6) предназначена для создание заявок на починку бытового оборудования. На данной странице расположена форма и история заявок студента. После заполнения формы студенту необходимо нажать кнопку «Сохранить», чтобы данные сохранились в базе данных.



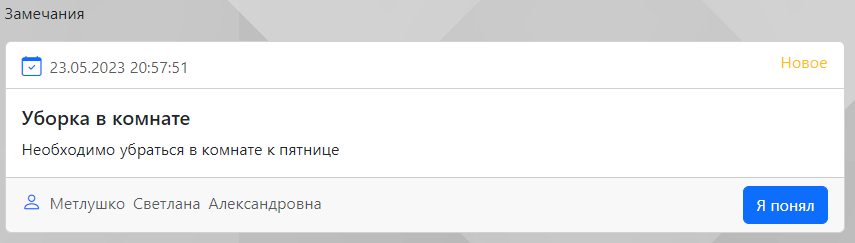
1. Форма для оформления заявки на починку оборудования

В личном кабинет студент может делать отчет об оплате общежития. Для этого была реализована страница PayHostel (рисунок 3.7). Студенту необходимо заполнить форму, а затем нажать кнопку «Сохранить». На данной странице также находится история оплаты, текущая стоимость общежития и оплаченная сумма.



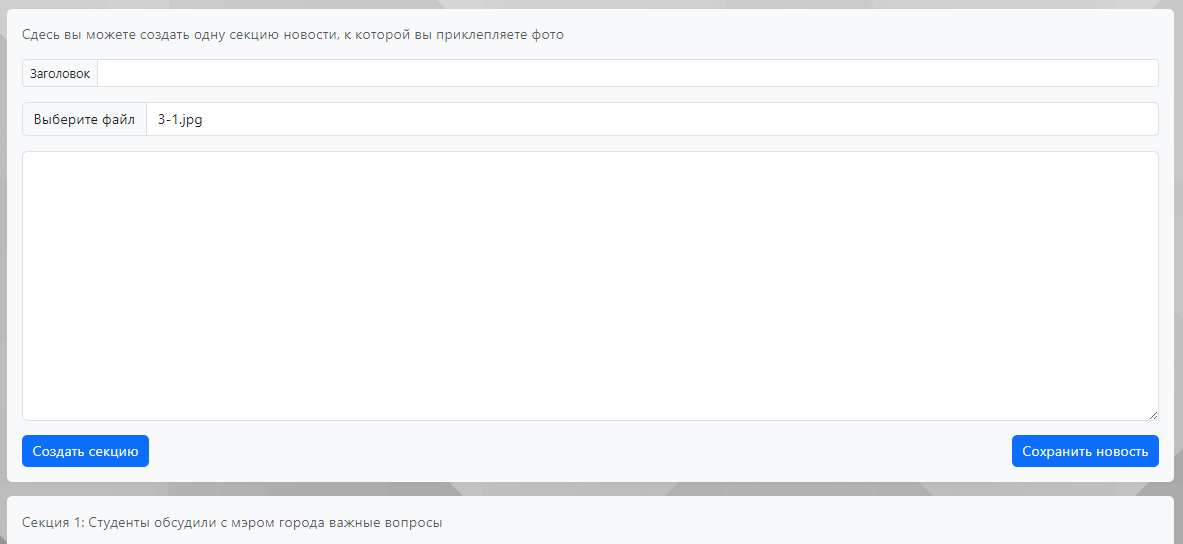
1. Форма для отчетности по оплате за проживания

На странице Claim (рисунок 3.8) предоставлен список замечаний, которые воспитатель сделал определенному студенту. Для того, чтобы изменить статус с «Новое» на «Прочитано», студенту необходимо нажать на кнопку «Я понял», после чего отправится запрос на сервер для изменения статуса. В противном случае, воспитатель увидит, что данное замечание студентом не прочитано.



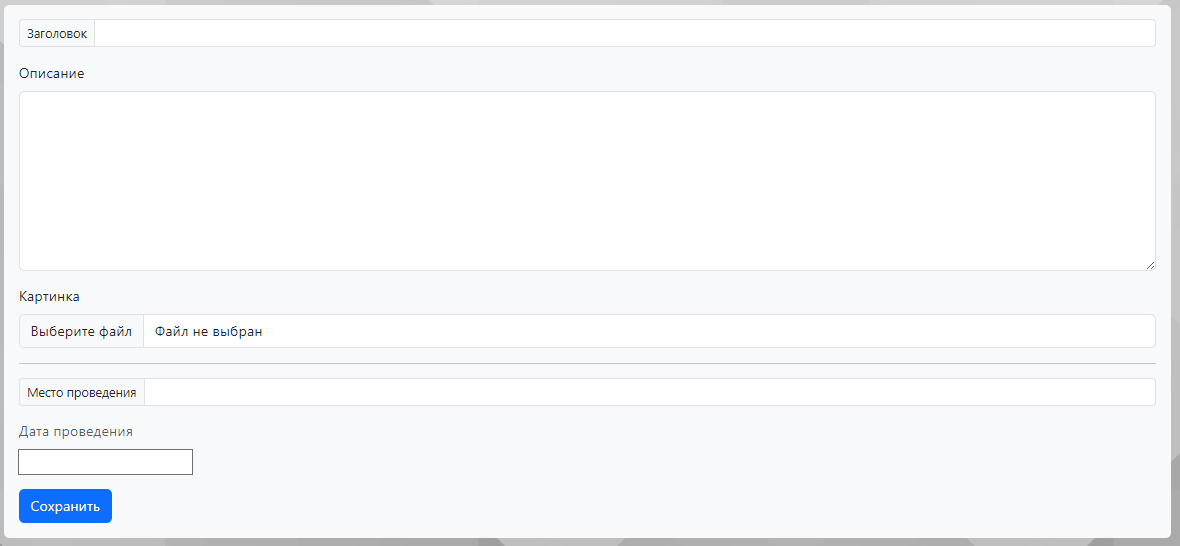
1. Замечание для студента

Воспитатель может создавать новости. Для этого была реализована страница NewNewsForm (рисунок 3.9). После заполнения всех полей, воспитателю необходимо нажать кнопку «Создать секцию», после чего создается массив объектов с полями header, file, description. Таким образом, воспитатель создает полноценную новость, которая состоит из нескольких секций. После создания нескольких секций, воспитателю необходимо нажать кнопку «Сохранить новость» для сохранения ее в базе данных.



1. Форма для создания новости

Для создания мероприятий была реализованная страница CreateEvent (рисунок 3.10). Данная форма имеет дополнительные поля: место проведения и дата проведения. Поле «Дата проведения» является специальной компонентой DatePicker, которая позволяет выбрать дату и время. Все поля являются обязательными. Для сохранения данных необходимо нажать на кнопку «Сохранить».



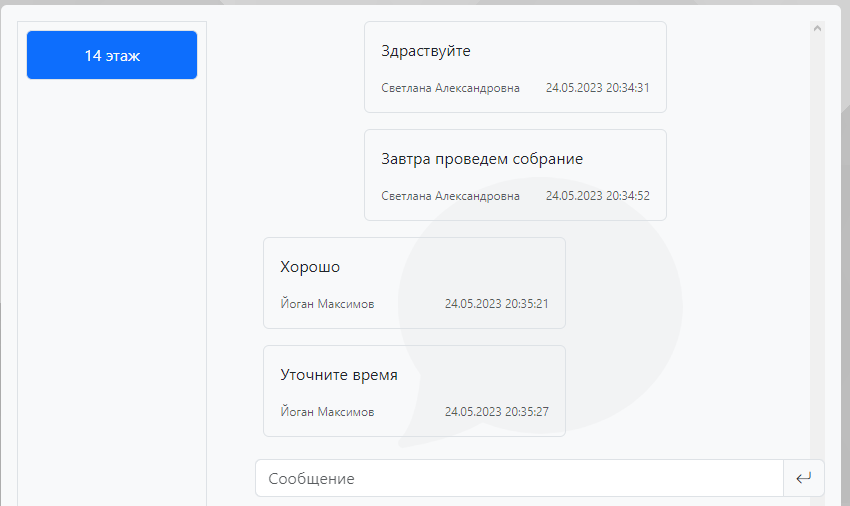
1. Форма для создание событий

Для общения воспитателя со студентами был реализован чат. Чат может создавать только воспитатель. Для этого была реализована страница и полем для ввода названия чата, списком студентов и кнопкой «Создать чат» (рисунок 3.11). Чтобы создать новый чат, воспитателю необходимо ввести названия чата и выбрать от одного до нескольких студентов.



1. Создание нового чата

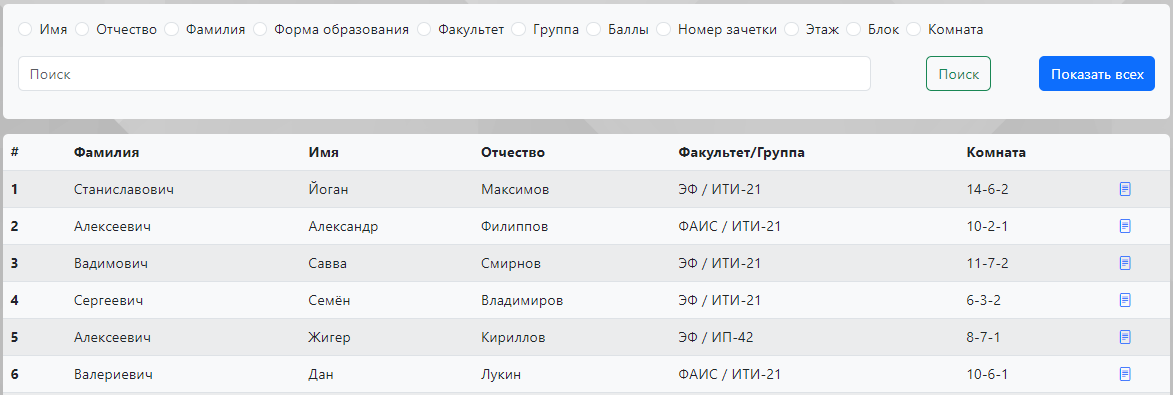
После того, как воспитатель создаст чат, студент и воспитатель могу начать общение. Для этого студенту и воспитателю необходимо перейти на страницу Chat (рисунок 3.12)



1. Вид страницы «Чат»

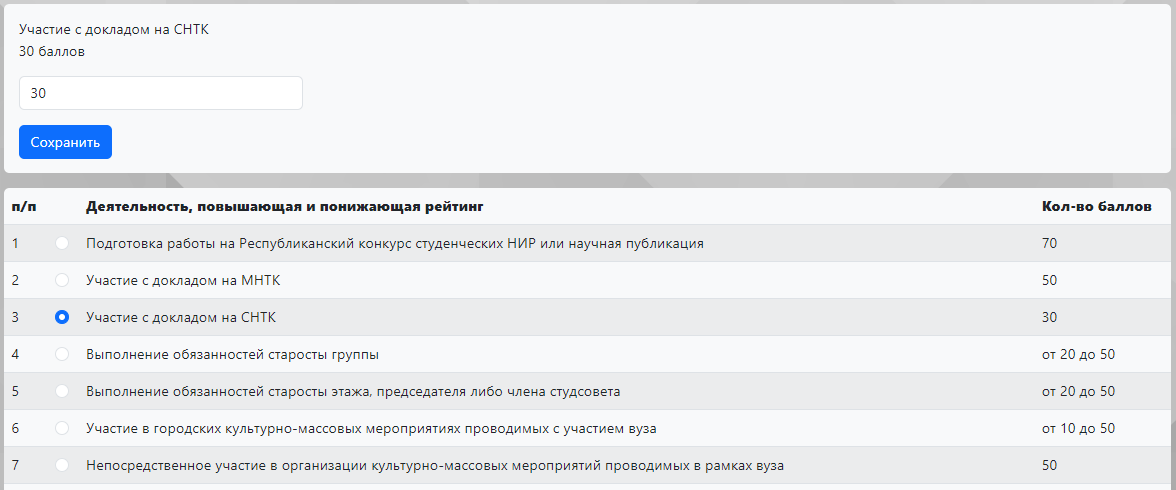
Для просмотра списка студентов, воспитатель может перейти на страницу StudentsList (рисунок 3.13). На данной странице расположен поиск по параметрам и таблица с краткой информацией о студентах. Для того, чтобы посмотреть полную информации о конкретном студенте, необходимо нажать на иконку «Список» в правой части строки таблицы.

Чтобы быстро находить нужного студента в общем списке, воспитателю необходимо выбрать нужный параметр, например, имя, и далее ввести в строку «Поиск» нужную подстроку, после чего нажать на кнопку «Поиск».



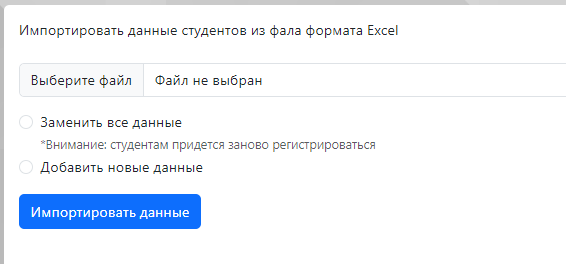
1. Список студентов

Страница ChangeBalls (рисунок 3.14) предназначена для того, чтобы воспитатель мог изменить количество баллов определенному студенту. Для этого ему необходимо выбрать нужный пункт из таблицы «Деятельность, повышающая или понижающая рейтинг» и ввести рекомендованное количество баллов. После чего нажать на кнопку «Сохранить».



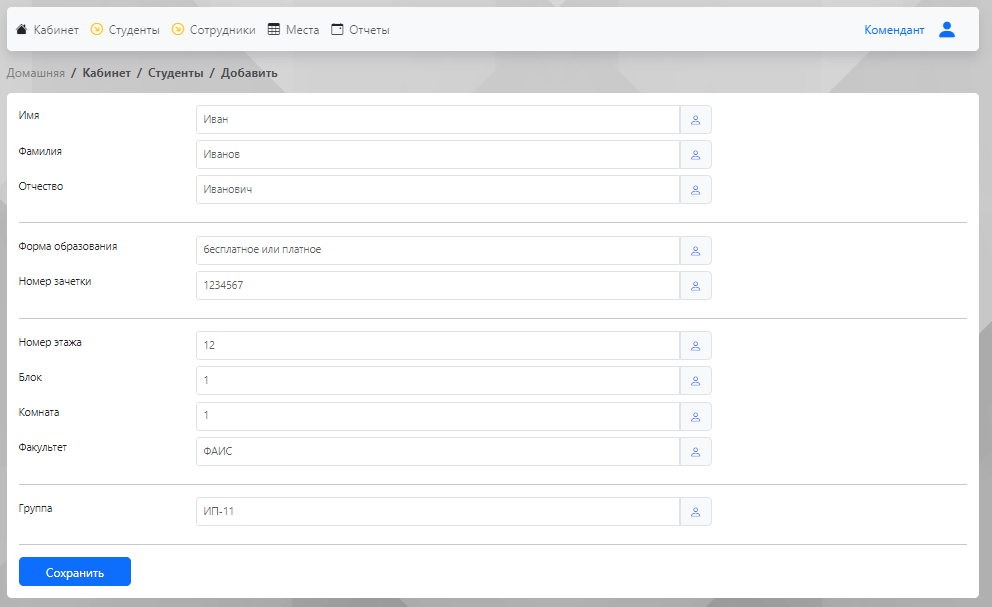
1. Страница изменения количества баллов

Страница ImportStudents (рисунок 3.15) предназначено для добавления новых студентов. Добавление происходит путем импорта файла Excel в базу данных. Для того, чтобы осуществить импорт, коменданту необходимо выбрать файл и один из двух пунктов: заменить все данные или добавить новые данные. В первом случаем данные студентов перезаписываются, а старые данные помещаются в архив. Во втором случаем к текущим данным добавляются новые.



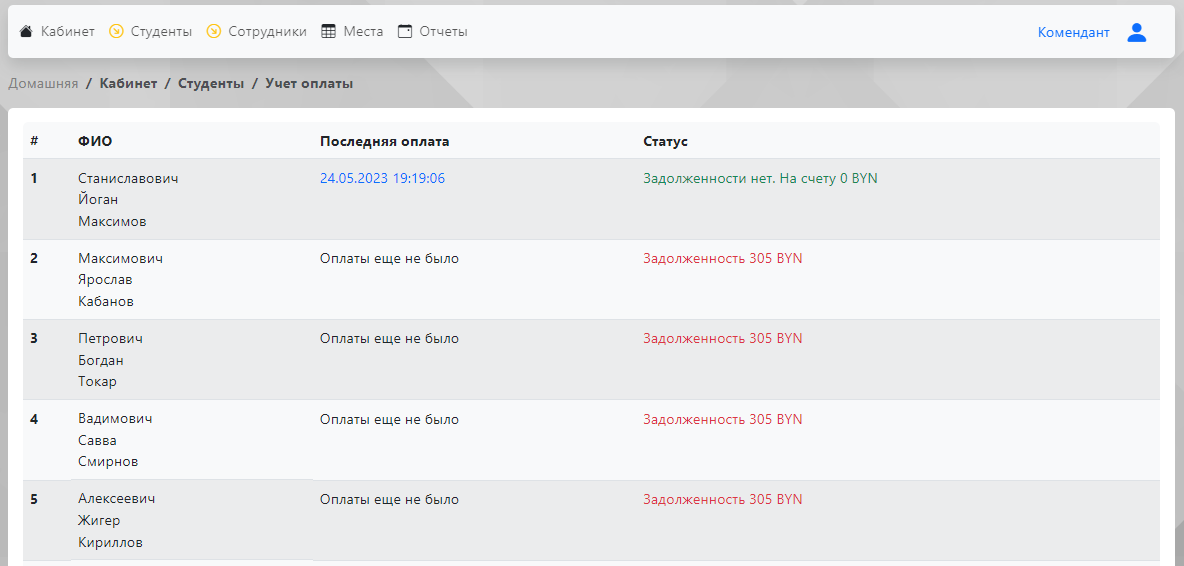
1. Вид страницы «Импорт студентов»

Также комендант может добавить одного студента, используя форму на странице AddStudent (рисунок 3.16). Формы состоит из следующих полей: имя, фамилия, отчество, форма образования, номер зачетной книжки, номер этажа, блок, комната, факультет, группа. В данной форме все поля являются обязательными. Для сохранения данных коменданту необходимо нажать на кнопку «Сохранить», после чего форма отправляется на сервер и проходит проверку на валидность.



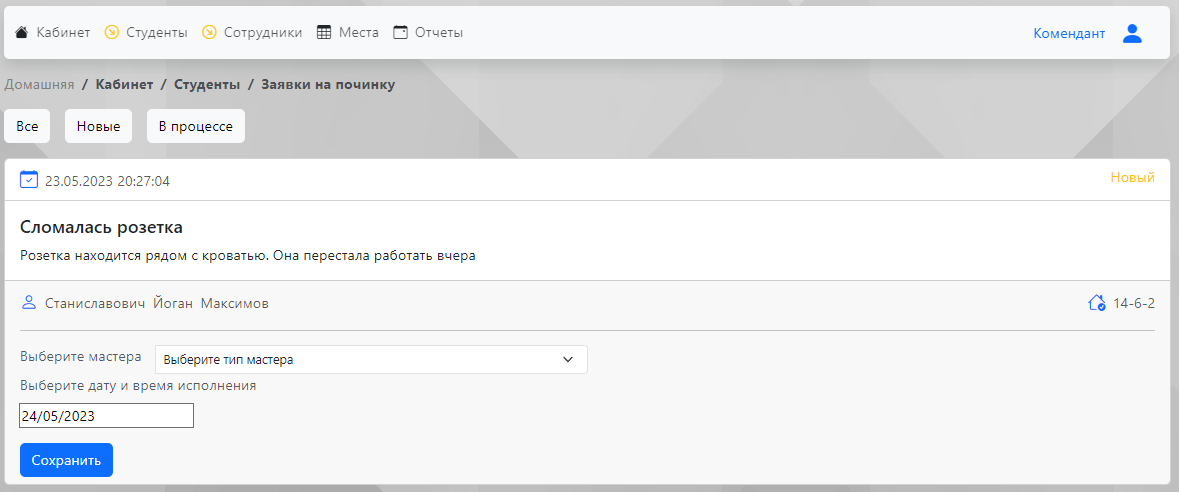
1. Форма «Добавление студента»

Страница ViewPayHostel (рисунок 3.17) предназначена для мониторинга оплаты за проживание в общежитии. На данной странице комендант может провести учет оплаты всех студентов, которые обучаются на платной основе. В случае, если у студента имеется задолженность – эта задолженность выводится красным цветом. Если задолженность отсутствует, комендант увидит его сумму на счету.



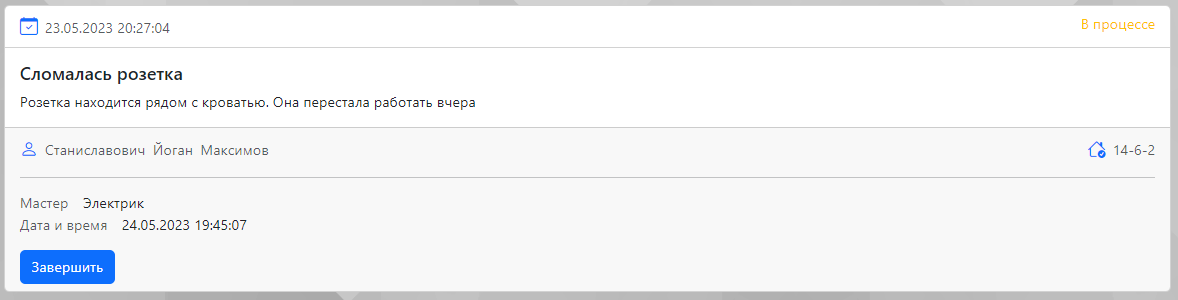
1. Просмотр оплаты за проживание

Страница RepairView (рисунок 3.18) предназначена для просмотра заявок на починку оборудования. Для удобства просмотра была предусмотрена фильтрация, с помощью которой комендант может просмотреть все, только новые, или те, которые сейчас имеют статус «В процессе». После того, как студент сделал новую заявку, комендант может принять её. Для этого нужно выбрать соответствующего мастера и дату и время исполнения и нажать на кнопку «Сохранить».



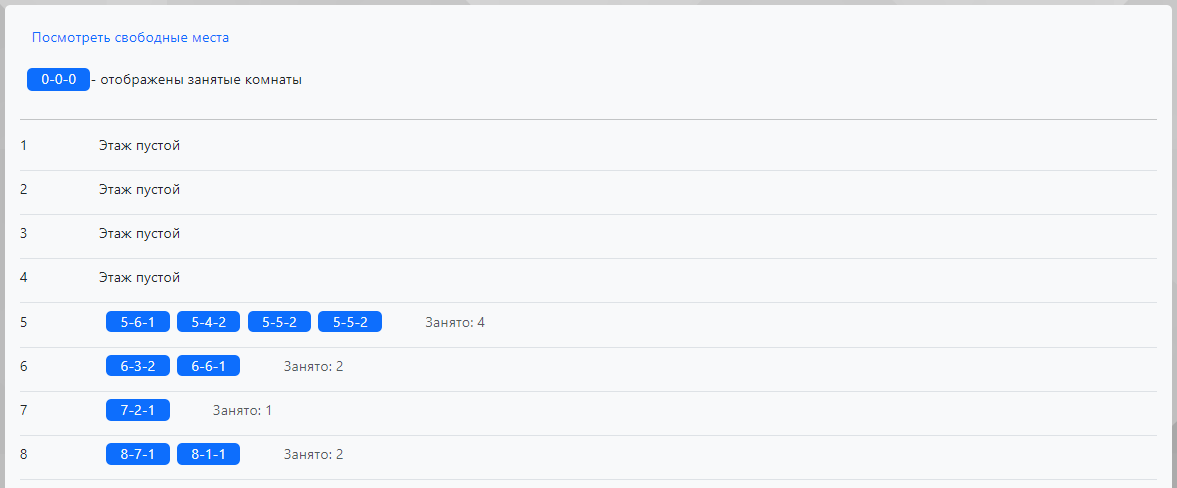
1. Вид страницы «Просмотр заявок на починку оборудования»

После того, как комендант примет заявку, она получит статус «В процессе». Для того, чтобы закрыть заявку, коменданту необходимо будет нажать кнопку «Завершить» (рисунок 3.19).



1. Вид заявки со статусом «В процессе»

Страница AccPlaces (рисунок 3.20) предназначена для мониторинга занятых мест в общежитии. В случае, если на этаже все комнаты пустые, комендант увидит соответствующее сообщение.



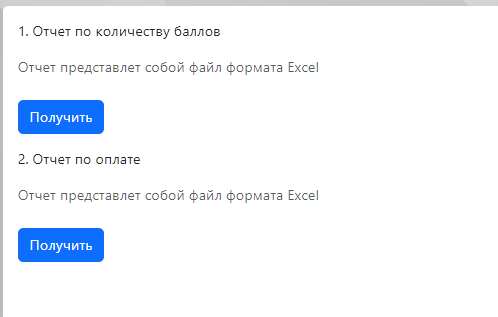
1. Список занятых мест

Страница FreePlaces (рисунок 3.21) предназначена для мониторинга свободных мест в общежитии. На данной странице представлены свободные комнаты на соответствующих этажах. Предполагается, что в каждой комнате два места, и в случае, если эти места свободны, они отображаются с помощью «/2».



1. Свободные места в общежитии

Страница Reports (рисунок 3.22) предназначен для генерации отчетов, которые являются файлами формата Excel. Для того, чтобы сгенерировать отчет, коменданту необходимо нажать одну из нескольких кнопок «Получить», после чего начнется автоматическое скачивание файла.



Уровень представления реализован с использованием библиотеки Bootstrap. Это позволило значительно упростить и ускорить процесс разработки пользовательского интерфейса, а также уделить больше внимания созданию понятной структуры приложения. Уровень представления выполнен в минималистичном стиле, что позволит пользователю сосредоточиться на содержимом и функциональности интерфейса.

* 1. Уровень бизнес-логики приложения

Маршруты в Express позволяют определить, как приложение должно обрабатывать запросы, направленные на конкретные URL-адреса. Каждый маршрут может иметь ассоциированный обработчик (handler), который выполняет определенные действия при получении запроса.

Уровень бизнес-логики представлен следующими маршрутами:

* auth.route – маршрут, отвечающий за обработку данных уровня представления, связанного с авторизацией пользователей;
* student.route – маршрут, отвечающий за обработку данных уровня представления, связанного со студентом;
* mentor.route – маршрут, отвечающий за обработку данных уровня представления, связанного с воспитателем;
* news.route – маршрут, отвечающий за обработку данных уровня представления, связанного с новостями;
* event.route – маршрут, отвечающий за обработку данных уровня представления, связанного с мероприятиями;
* chat.route – маршрут, отвечающий за обработку данных уровня представления, связанного с чатом;
* common.route – маршрут, отвечающий за обработку данных уровня представления, связанного с чатом, воспитателем и со студентом;
* admin.route – маршрут, отвечающий за обработку данных уровня представления, связанного с комендантом;
* repair.route – маршрут, отвечающий за обработку данных уровня представления, связанного с заявками на починку оборудования;

Обработчики маршрутов бизнес-логики приведены в таблицах 3.1 – 3.9.

Таблица 3.1 – Обработчики маршрута auth.route

|  |  |  |
| --- | --- | --- |
| Обработчик | Параметры | Описание |
| router.post('/login') | string login, string password | Возвращает данные пользователя |
| router.post('/registr') | number numberTest, string email | Возвращает сообщение о том, что логин и пароль были отправлены на почту |

Таблица 3.2 – Обработчики маршрута student.route

|  |  |  |
| --- | --- | --- |
| Обработчик | Параметры | Описание |
| router.post('/import-students') | Array<Object> students, string type | Возвращает сообщение и статус запроса |
| router.get('/get-info') | - | Возвращает всех студентов из базы данных |
| router.get('/get-student-id') | string id | Возвращает найденного по id студента |
| router.post('/change-status-claim') | number numberTest, number ind | Возвращает новый объект Claim с измененным статусом |
| router.post('/create-claim') | number numberTest, string header, string text, Object mentor, Date dateAndTime | Создает новое замечание и возвращает новый объект Claim |
| router.get('/get-places') | - | Возвращает массив занятых мест в общежитии |
| router.post('/add-tech') | number numberTest, string model, string type | Добавляет новую запись в массив Techs и возвращает созданный объект |
| router.post('/update-info') | number numberTest, string email, string login, string newPassword, string oldPassword | Изменяет личные данные студента и возвращает новый объект Student |
| router.post('/get-remarks') | number numberTest | Возвращает массив замечаний |
| router.post('/add-student') | string firstName, string string secondName, string middleName, string formEducation, number number floor, number block, number apartament, string faculty, string group | Добавляет нового студента в коллекцию Students |
| router.post('/pay-hostel') | string receipt, number numberTest, number payment | Добавляет новые данные об оплате общежития |
| router.post('/get-pay') | number numberTest | Возвращает массив оплаты за проживание в общежитии |
| router.post('/get-balls') | number numberTest | Возвращает массив баллов |

Таблица 3.3 – Обработчики маршрута mentor.route

|  |  |  |
| --- | --- | --- |
| Обработчик | Параметры | Описание |
| router.post('/info-mentor') | String login | Возвращает объект Mentor, найденный по логину |
| router.post('/update-info') | Number id, string login, string newPassword, string oldPassword, string phone, string email | Изменяет личные данные воспитателя и возвращает новый объект Mentor |
| router.post('/get-students-by-impact') | Object impact | Возвращает массив студентов в зависимости от того, на какие этажи распространяется влияние воспитателя |
| router.post('/update-balls') | Number numberTest, number num, string summary | Обновляет количество баллов у студента и возвращает обновленные данные |

Таблица 3.4 – Обработчики маршрута news.route

|  |  |  |
| --- | --- | --- |
| Обработчик | Параметры | Описание |
| router.post('/create-section') | String header, string description, number id | Создает новую секцию новости |
| router.post('/create-news') | Object mentor | Создает новость и возвращает id созданного объекта |
| router.get('/get-news') | - | Возвращает все новости из базы данных |
| router.get('/get-news-id') | Number id | Возвращает новость по id |

Таблица 3.5 – Обработчики маршрута event.route

|  |  |  |
| --- | --- | --- |
| Обработчик | Параметры | Описание |
| router.post('/create-event') | String header, string description, Date dateEvent, string placeEvent | Создает мероприятие и возвращает id созданного объекта |
| router.get('/get-events') | - | Возвращает все мероприятия из базы данных |

Таблица 3.6 – Обработчики маршрута chat.route

|  |  |  |
| --- | --- | --- |
| Обработчик | Параметры | Описание |
| router.post('/create-chat') | Array<string> arrayId, string name, string idMentor | Создает новый чат и возвращает id созданного объекта |
| router.post('/get-chats') | Array<string> idChats | Возвращает все чаты для переданного массива |
| router.post('/get-messages') | String id | Возвращает все сообщение для найденного по id чата |

Таблица 3.7 – Обработчики маршрута common.route

|  |  |  |
| --- | --- | --- |
| Обработчик | Параметры | Описание |
| router.get('/load') | file img | Возвращает картинку с название img |
| router.get('/get-cost-hostel') | - | Возвращает текущую стоимость проживания в общежитии |

Таблица 3.8 – Обработчики маршрута admin.route

|  |  |  |
| --- | --- | --- |
| Обработчик | Параметры | Описание |
| router.get('/get-admin-info') | number id | Возвращает объект Admin, найденный по id в базе данных |
| router.get('/get-employee') | - | Возвращает массив воспитателей |
| router.post('/create-employee') | String secondName, string string firstName, string middleName, number impactFrom, number impactTo, string login, string password | Добавляет нового воспитателя в коллекцию Mentors |
| router.get('/delete-student') | Number id | Удаляет данные студента из базы данных |
| router.get('/get-payments') | - | Возвращает массив Payments для всех студентов, у которых поле formEducation равно «платное» |
| router.get('/get-free-places') | - | Возвращает массив свободных мест в общежитии |
| router.get('/report-balls') | - | Возвращает сгенерированный Excel файл с отчетом по количеству баллов |
| router.get('/report-pays') | - | Возвращает сгенерированный Excel файл с отчетом по оплате |

Таблица 3.9 – Обработчики маршрута repair.route

|  |  |  |
| --- | --- | --- |
| Обработчик | Параметры | Описание |
| router.get('/get-repairs-all') | - | Возвращает все заявки на починку оборудования |
| router.post('/get-repairs-id') | Number id | Возвращает заявку на починку оборудования, найденного по id |
| router.post('/create') | Object user, string header, string description, Object room | Создает новый объект Repair и возвращает созданный объект |
| router.post('/change-status') | Number id | Изменяет статус заявки и возвращает измененный статус |

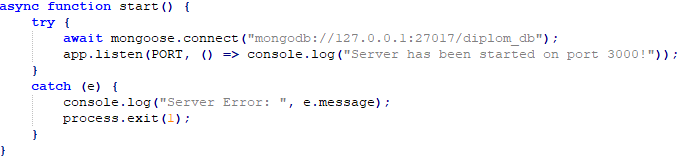
Каждый обработчик содержит дополнительный функционал для обработки ошибок. В случае возникновения ошибки, клиент получит соответствующее уведомление, а результат запроса завершится с кодом 400 или 500.

* 1. Уровень данных в приложении

Уровень данных в приложении обеспечивает связь между приложением и базой данных MongoDB. Этот уровень отвечает за работу с данными, их сохранение, извлечение, обновление и удаление. Он представлен в виде моделей. Модели представляют собой схемы и методы для работы с данными в базе данных MongoDB. Они названы в соответствии с названиями коллекций, схема которых приведена на рисунке 2.2, только в единственном числе.

Модель Mongoose позволяет определить структуру документов, которые будут храниться в коллекции MongoDB. Схема определяет поля, типы данных, валидацию, уникальные ограничения и другие атрибуты для каждого поля. Определение схемы облегчает работу с данными, позволяет контролировать их структуру и обеспечивает целостность данных.

Подключение к базе данных MongoDB с использованием Mongoose – это процесс установления соединения между приложением и базой данных MongoDB, а также настройка параметров для взаимодействия с базой данных. Для подключения была реализована функция start (рисунок 3.22), которая использует метод connect объекта mongoose со строкой подключения.



1. Функция для подключения к базе данных

После успешного подключения к базе данных запускается сервер на указанном порту. В случае ошибки приложение экстренно завершит работу с кодом 1.

1. ТЕСТИРОВАНИЕ, ВЕРИФИКАЦИЯ И ВАЛИДАЦИЯ ПРОГРАММНОГО КОМПЛЕКСА
   1. Тестирование пользовательских форм

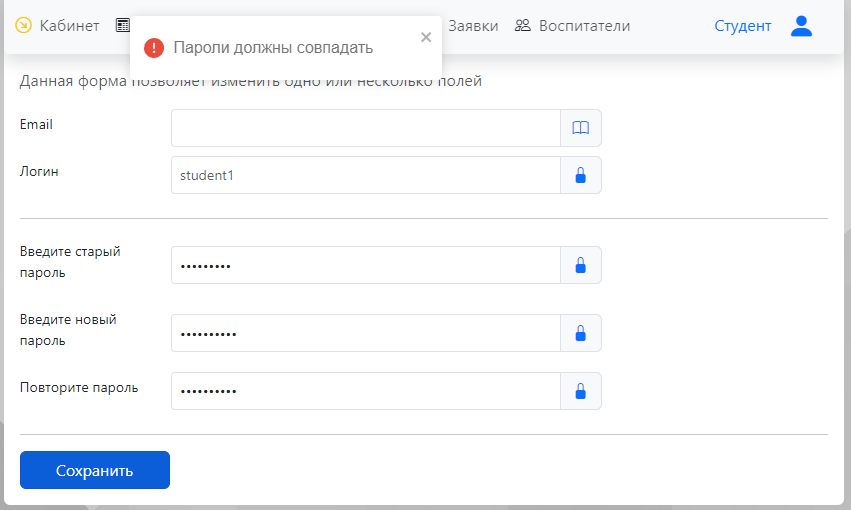
Тестирование пользовательских форм относится к проверке корректности работы и удобства использования форм в интерфейсе пользователя. Формы представляют собой элементы интерфейса, через которые пользователи вводят и отправляют данные, например, регистрационные данные, почтовые адреса, кредитные карты и т.д.

В таблице 4.1 проведены пошаговые тесты, которые в конечном итоге должны привести к ожидаемому результату.

Таблица 4.1 – Примеры проводимых тестов

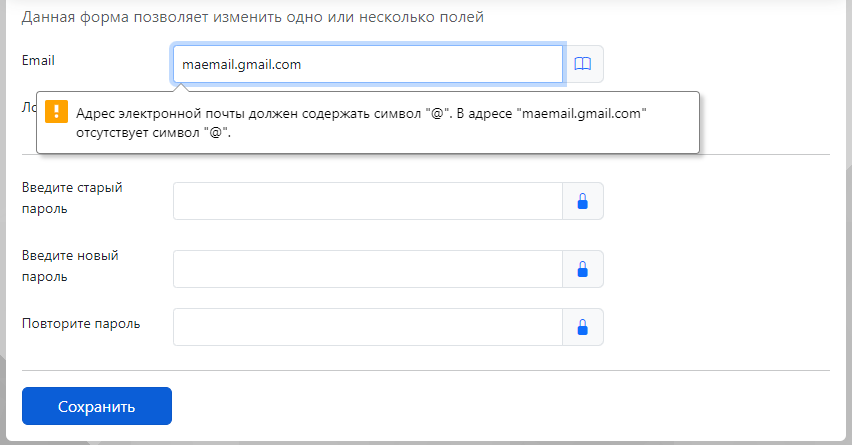
|  |  |  |  |
| --- | --- | --- | --- |
| Краткое описание | Шаги по воспроизведению | Ожидаемый результат | Полученный результат |
| Проверка совпадения паролей | Войти в личный кабинет воспитателя или студента;  Ввести старый пароль;  Ввести новый пароль «password12»;  Повторить пароль «password22»;  Нажать кнопку «Сохранить». | Приложение должно выводить следующее уведомление: «Пароли должны совпадать» | Приложение выводит уведомление «Пароли должны совпадать» |
| Проверка правильности email | Войти в личный кабинет воспитателя или студента;  Заполнить поле email строкой  myemail.gmail.com;  Нажать кнопку «Сохранить». | Приложение должно выводить следующее уведомление: «Адрес электронной почты должно содержать символ @» | Приложение выводит уведомление «Адрес электронной почты должно содержать символ @» |
| Проверка поля обязательного к заполнению | Открыть любую форму;  Нажать кнопку «Сохранить». | Приложение должно выводить следующее уведомление: «Заполните это поле» | Приложение выводит уведомление «Заполните это поле» |
| Проверка на количество символов | Войти в личный кабинет воспитателя;  Перейти на страницу создания нового события;  Заполнить все поля не более, чем 5 символами;  Нажать кнопку «Сохранить». | Приложение должно выводить следующее уведомление: «Введите данные в указанном формате» | Приложение выводит  уведомление: «Введите данные в указанном формате» |
| Проверка введенных данных на число | Перейти на форму, где требуется ввести число;  Попытаться ввести строку. | Приложение должно не позволить вводить символы, отличные от чисел | Приложение не позволило вводить символы, отличные от чисел |

На рисунке 4.1 приведено уведомление при попытке сохранить форму с неправильно повторенным паролем.



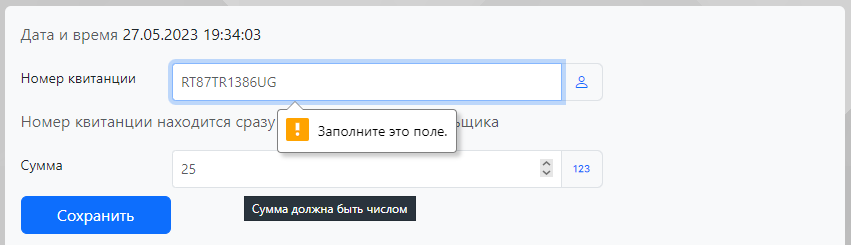
1. – Вид уведомления при неправильности повторения пароля

На рисунке 4.2 представлен результат проверки введенного email на правильность.



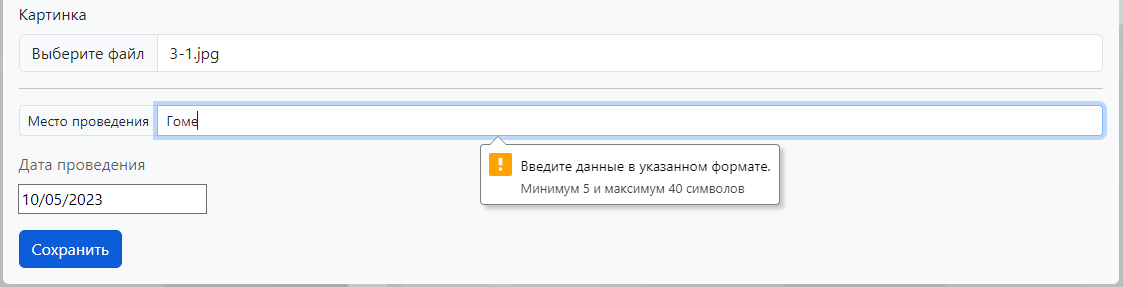
1. – Результат проверки email

На рисунке 4.3 представлен результат проверки на обязательное поле. Так как поле «Сумма» должно быть числом, при попытке ввести не число, программа не позволит это сделать и покажет подсказку «Сумма должна быть числом».



1. – Проверка на обязательное поле и на число

На рисунке 4.4 представлен результат проверки на разрешенное количество символов. Данная проверка ограничивает минимальное и максимальное разрешенное количество введенных символов.



1. – Результат проверки на количество символов

Тестирование пользовательских форм имеет важное значение, поскольку формы являются ключевым элементом взаимодействия между пользователем и приложением. Некорректная обработка данных в формах может привести к ошибкам или сбоям в приложении. Тестирование форм помогает выявить и исправить потенциальные проблемы, такие как отсутствие обработки специальных символов, неправильная валидация или некорректная обработка ошибок.

* 1. Тестирование бизнес-логики

Бизнес-логика является основным компонентом программного продукта, отвечающим за обработку данных, принятие решений и выполнение бизнес-правил. Тестирование бизнес-логики позволяет убедиться, что она работает правильно, соответствует ожидаемым результатам и выполняет необходимые проверки и валидации данных. Оно помогает выявить потенциальные ошибки, пропуски или некорректные реализации в бизнес-логике еще до того, как они повлияют на работу приложения в целом. Это позволяет оперативно исправить проблемы и предотвратить возможные сбои или неправильное поведение системы.

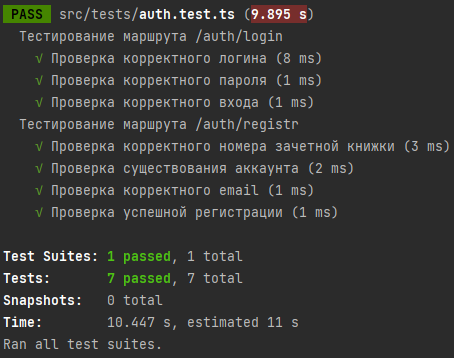
Тестирование бизнес-логики с помощью Jest позволяет повысить качество и надежность программного продукта, упростить разработку и поддержку кода, а также предоставить документацию и понимание ожидаемого поведения системы. Тесты бизнес-логики также могут служить своего рода документацией, позволяя разработчикам и другим заинтересованным сторонам лучше понять ожидаемое поведение и требования к функциональности. Тесты могут служить наглядным примером использования бизнес-логики и помогать при анализе и отладке проблемных ситуаций.

В таблице 4.2 приведен набор тестов для обработчиков auth.route, отвечающий за обработку запросов, связанных с авторизацией пользователей.

Таблица 4.2 – Набор тестов для auth.route

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название обработчика | Описание теста | Исходные данные | Ожидаемый результат | Результат теста |
| router.post  ('/auth/ login') | Проверка корректного логина | login: login123  password: pass123 | Сообщение «Такого логина не существует» и код 400 | Сервер возвращает сообщение «Такого логина не существует» и код 400 |
| router.post  ('/auth/ login') | Проверка корректного пароля | login: login123  password: pas | Сообщение «Неверный пароль» и код 400 | Сервер возвращает сообщение «Неверный логин» и код 400 |
| router.post  ('/auth/ login') | Проверка корректного входа | login: student1  password: student1 | Сообщение «Вход выполнен успешно» и код 200 | Сервер возвращает сообщение «Вход выполнен успешно» и код 200 |
| router.post  ('/auth /registr') | Проверка корректного номера зачетной книжки | numberTest: 123  email: email@gmail.com | Сообщение «Неверный номер зачетной книжки» и код 400 | Сервер возвращает сообщение «Неверный номер зачетной книжки» и код 400 |
| router.post  ('/auth /registr') | Проверка существования аккаунта | numberTest: 46753842  email:  email@gmail.com | Сообщение «У вас уже есть аккаунт» и код 400 | Сервер возвращает сообщение «У вас уже есть аккаунт» и код 400 |
| router.post  ('/auth /registr') | Проверка корректного email | numberTest: 46753842  email:  email.gmail.com | Сообщение «Неверный email» и код 400 | Сервер возвращает сообщение «Неверный email» |
| router.post('/auth /registr') | Проверка успешной регистрации | numberTest: 1583657  email:  pavel@gmail.com | Сообщение «Сообщение отправлено на почту» и код 200 | Сервер возвращает  cообщение «Сообщение отправлено на почту» и код 200 |

На картинке 4.5 приведены результаты исполнения набора тестов из таблицы 4.2.



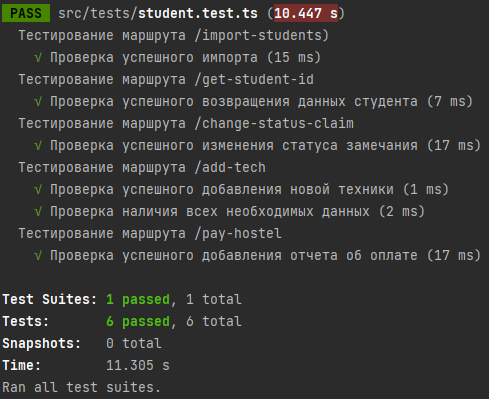
1. – Результаты тестирования auth.route

В таблице 4.3 приведен набор тестов для обработчиков student.route, отвечающий за обработку запросов, связанных со студентом.

Таблица 4.3 – Набор тестов для student.route

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название маршрута | Описание теста | Исходные данные | Ожидаемый результат | Результат теста |
| router.post('/import-students') | Проверка успешного импорта | Array  <Student> | Сообщение «Данные успешны выгружены» и код 200 | Сервер возвращает сообщение «Данные успешны выгружены» и код 200 |
| router.get('/get-student-id') | Проверка успешного возвращения данных студента | id: 2342w87er | Объект Student,  сообщение «Данные загружены» и код 200 | Сервер возвращает объект Student,  сообщение «Данные загружены» и код 200 |
| router.post('/change-status-claim') | Проверка успешного изменения статуса замечания | numberTest: 2478624,  ind: 1 | Объект Claim со статусом 1,  сообщение «Статус изменен» и код 200 | Сервер возвращает объект Claim со статусом 1,  сообщение «Статус изменен» и код 200 |
| router.post('/add-tech') | Проверка успешного добавления новой техники | numberTest: 2478624, model: AB-10, number: RT657, type: Холодильник | Сообщение «Новая техника добавлена» и код 200 | Сервер возвращает  сообщение «Данные добавлены» и код 200 |
| router.post('/add-tech') | Проверка наличия всех необходимых данных | numberTest: 2478624, number: RT657 | Сообщение «Модель и тип являются обязательными» и код 400 | Сервер возвращает  сообщение Сообщение «Модель и тип являются обязательными» и код 400 |
| router.post('/pay-hostel') | Проверка успешного добавления отчета об оплате | receipt: TE67UR32, numberTest: 2478624, payment: 20 | Сообщение «Оплата добавлена» и код 200 | Сервер возвращает Сообщение «Оплата добавлена» и код 200 |

На картинке 4.6 приведены результаты исполнения набора тестов из таблицы 4.3.



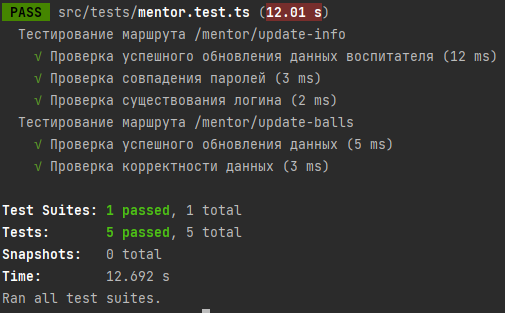
1. – Результаты тестирования student.route

В таблице 4.4 приведен набор тестов для обработчиков mentor.route, отвечающий за обработку запросов, связанных с воспитателем.

Таблица 4.4 – Набор тестов для mentor.route

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название маршрута | Описание теста | Исходные данные | Ожидаемый результат | Результат теста |
| router.post  ('/update-info') | Проверка успешного обновления данных воспитателя | Id: 2, login: mentor1, newPassword: newPass, oldPassword: oldPass, | Объект Mentor, сообщение «Данные обновлены» и код 200 | Сервер возвращает  объект Mentor, сообщение «Данные обновлены» и код 200 |
| router.post  ('/update-info') | Проверка совпадения паролей | Id: 2, newPassword: newPass, oldPassword: oldPass | Сообщение «Пароль должен совпадать со старым паролем» и код 400 | Сервер возвращает  сообщение «Пароль должен совпадать со старым паролем» и код 400 |
| router.post  ('/update-info') | Проверка существова-ния логина | id: 2  login:  mentor2 | Сообщение «Такой логин уже существует» и код 400 | Сервер возвращает  сообщение «Такой логин уже существует» и код 400 |
| router.post  ('/update-balls') | Проверка успешного обновления данных | numberTest: 4767246,  num: 30, summary: Участие с докладом на МНТК | Сообщение «Данные обновлены» и код 200 | Сервер возвращает  сообщение «Данные обновлены» и код 200 |
| router.post  ('/update-balls') | Проверка корректности данных | numberTest: 4767246, num: 3оа, summary: Участие с докладом на МНТК | Сообщение «Количество должно быть числом» и код 400 | Сервер возвращает  сообщение «Количество должно быть числом» и код 400 |

На картинке 4.7 приведены результаты исполнения набора тестов из таблицы 4.4.



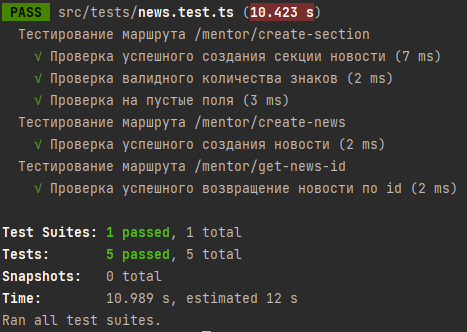
1. – Результаты тестирования mentor.route

В таблице 4.5 приведен набор тестов для обработчиков news.route, отвечающий за обработку запросов, связанных с новостями.

Таблица 4.5 – Набор тестов для news.route

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название маршрута | Описание теста | Исходные данные | Ожидаемый результат | Результат теста |
| router.post('/create-section') | Проверка успешного создания секции новости | Header: строка из 25 символов, description: строка из 200 символов, id: 3 | Сообщение «Новость создана» и код 200 | Сервер возвращает  сообщение «Новость создана» и код 200 |
| router.post('/create-section') | Проверка валидного количества знаков | Header: строка из 3 символов, description: строка из 1000 символов, id: 3 | Сообщение «Не верная длина заголовка или описания» и код 400 | Сервер возвращает сообщение «Не верная длина заголовка или описания» и код 400 |
| router.post('/create-section') | Проверка на пустые поля | Header: строка из 0 символов, description: строка из 0 символов, id: 3 | Сообщение «Поля обязательные к заполнению» и код 400 | Сервер возвращает сообщение «Поля обязательные к заполнению» и код 400 |
| router.post('/create-news') | Проверка успешного создания новости | Object Mentor | Сообщение «Новость создана», id новости и код 200 | Сервер возвращает сообщение «Новость создана», id новости и код 200 |
| router.get('/get-news-id') | Проверка успешного возвращение новости по id | id: 3 | Найденная новость по переданному id и код 200 | Сервер вернул новость по переданному id и код 200 |

На картинке 4.8 приведены результаты исполнения набора тестов из таблицы 4.5.



1. – Результаты тестирования news.route

Express основан на концепции маршрутизации, и правильное функционирование маршрутов является критически важным для веб-приложений. Тестирование маршрутов Express с помощью Jest позволяет удостовериться, что они работают согласно ожиданиям и возвращают правильные ответы на запросы. Тесты позволили нам проверить различные сценарии и условия работы приложения, включая различные HTTP-методы, передачу параметров, обработку ошибок и другие аспекты, а также обнаружить и исправить потенциальные уязвимости, неправильное поведение или сбои в работе приложения, повысив его стабильность и надежность.