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](#_Toc132724577)

[1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ АВТОМАТИЗАЦИИ ЖИЛОГО ФОНДА СТУДЕНЧЕСКОГО ОБЩЕЖИТИЯ 4](#_Toc132724578)

[1.1 Обзор существующих систем автоматизации 4](#_Toc132724579)

[1.2 Анализ разработки и проектирования веб приложений 5](#_Toc132724580)

[1.3 Анализ используемых технологий для реализации поставленной задачи 6](#_Toc132724581)

[1.4 Анализ инструментальных средств автоматизации разработки и тестирования 10](#_Toc132724582)

[1.5 Техническое задание для клиент-серверного программного продукта «Программный комплекс автоматизации обслуживания жилого фонда студенческого общежития» 13](#_Toc132724583)

[2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И АЛГОРИТМЫ 14](#_Toc132724584)

[2.1 Анализ предметной области 14](#_Toc132724585)

[2.2 Функциональная модель программного комплекса 14](#_Toc132724586)

[2.3 Информационная модель программного комплекса 17](#_Toc132724587)

# **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.

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.

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.

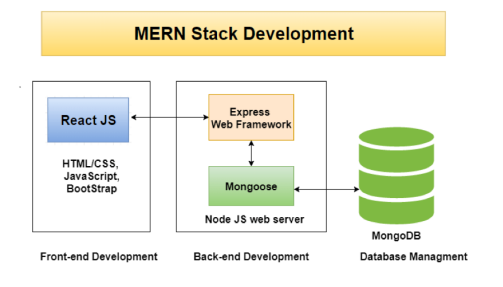
Кроме того, веб-приложения могут использовать клиент-серверную архитектуру, где клиентское приложение работает на стороне пользователя, а серверное приложение обрабатывает запросы и предоставляет данные. Например, веб-приложение может использовать AJAX для асинхронной передачи данных между клиентом и сервером, что улучшает пользовательский интерфейс и ускоряет процесс загрузки страниц.

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).

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.

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.

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.

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.

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.

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.

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.

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.

* 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 вести учет личного технического оборудования, а также количества свободных и занятых мест. Для заполнения списка студентов комендант сможет импортировать файл формата Excel в базу данных. Также у него будет возможность производить мониторинг оплаты за проживание.

Для студентов будет реализовано личное пространство, где они смогут следить за текущими новостями и событиями, а также общаться с воспитателем в созданном ей чате и создавать заявки на починку бытового оборудования.

Для воспитателей приложение предусматривает следующие функции: создание и ведение ленты новостей и 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; 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

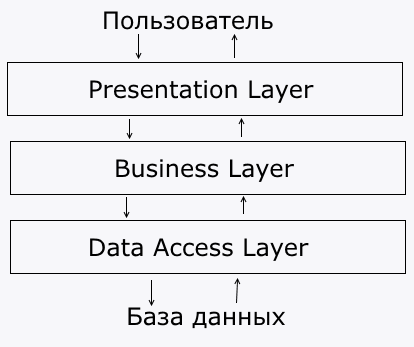
Анализ предметной области «Автоматизация обслуживания жилого фонда студенческого общежития» позволяет выделить следующие основные аспекты:

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, чтобы гости и студенты были в курсе .
4. Связь с управлением вуза. Необходимо иметь возможность связываться с управлением вуза для решения вопросов, связанных с проживанием студентов 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.

* 1. Функциональная модель программного комплекса

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.

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

Данная диаграмма 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 коллекция 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 для разрабатываемого продукта были выделены следующие сущности:

* коллекция «Студент» хранит в себе информацию о студентах;
* коллекция «Аккаунт»;
* коллекция «Сотрудники»;
* коллекция «Мероприятие»;
* коллекция «Новость»;
* коллекция «Чат».

Таблица 2.1 – Коллекция «Студент»

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

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 |

Таблица 2.3 – Коллекция «Аккаунты»

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

Таблица 2.4 – 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<Object> | Нет | Нет |

Таблица 2.5 – 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 |