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:

«Web-приложение по поддержке процессов бытового обслуживания населения»

|  |  |
| --- | --- |
| Исполнитель: | студент гр. ИП–42 |
|  | Кирпиченко Д.Д. |
| Руководитель от предприятия: | гл. программист |
|  | Макаревич В.Л. |
| Руководитель: | ст. преподаватель |
|  | Шибеко В.Н. |

Дата проверки:

Дата допуска к защите:

Дата защиты:

Оценка работы:

Подписи членов комиссии

Гомель 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc132837358)

[1 Аналитический обзор существующих методов и средств решения поставленной задачи на основе изучения литературных источников по теме дипломной работы. 4](#_Toc132837359)

[1.1 Обзор существующих систем автоматизации 4](#_Toc132837360)

[1.2 Анализ современного языка программирования JavaScript 5](#_Toc132837361)

[1.3 Анализ используемых технологий для реализации поставленной задачи 7](#_Toc132837362)

[1.4 Анализ инструментальных средств автоматизации разработки и тестирования 10](#_Toc132837363)

[1.5 Техническое задание для программного продукта «Web-приложение по поддержке процессов бытового обслуживания населения» 15](#_Toc132837364)

[2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И АЛГОРИТМЫ 16](#_Toc132837365)

[2.1 Анализ предметной области 16](#_Toc132837366)

[2.2 Функциональная модель программного комплекса 16](#_Toc132837367)

[2.3 Информационная модель программного комплекса 18](#_Toc132837368)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 27](#_Toc132837369)

# ВВЕДЕНИЕ

В настоящее время все большее количество людей обращается за помощью к профессионалам, которые могут оказать бытовые услуги: уборка квартиры, ремонт, доставка продуктов, услуги мастера на дом и т.д. Однако, часто найти подходящего исполнителя для этих услуг может быть непросто. Именно поэтому существует растущий спрос на приложения, которые позволяют быстро и удобно заказать необходимую услугу на дом.

Цель данной дипломной работы - разработка приложения для оказания бытовых услуг населению, которое позволит пользователям легко и быстро находить подходящих исполнителей для выполнения работ. Приложение будет иметь функционал для поиска исполнителя по различным критериям, оценки качества выполненных работ, расчета стоимости услуг и возможности оставлять отзывы и рейтинги.

Основная задача состоит в создании удобного и надежного сервиса, который позволит пользователю быстро и безопасно заказать необходимую услугу на дом. В рамках дипломной работы будет произведен анализ рынка услуг и существующих приложений, а также проведена разработка и тестирование программного обеспечения. Ожидаемый результат – готовое приложение, способное удовлетворить потребности населения в бытовых услугах.

1. Аналитический обзор существующих методов и средств решения поставленной задачи на основе изучения литературных источников по теме дипломной работы.

## Обзор существующих систем автоматизации

Создание приложения по оказанию бытовых услуг населению может иметь ряд преимуществ перед традиционными методами предоставления услуг. Так, например, использование современных технологий позволяет быстро и удобно заказывать услуги через мобильное приложение или сайт. Приложение может включать в себя функции, такие как выбор услуги, выбор времени и даты, оплата онлайн, отслеживание выполнения заказа и т.д. Такой подход удобен для клиентов, так как позволяет сэкономить время и избежать необходимости посещать офисы или ожидать по телефону.

Одним из примеров таких приложений является MasterOk, это белорусский сайт, который предлагает широкий спектр услуг для населения.

Это приложение предоставляет возможность выбрать нужную категорию, а затем – услугу. После чего подробно прочитать про эту услугу и посмотреть места, где данную услугу могут выполнить. Данное приложение имеет существенный минус для получения услуги надо лично явится в один из сервисов.

Таким образом, MasterOkявляется больше информационным сайтом, чем приложением по оказанию услуг.

Другим примером таких приложений является Kufar. Kufar ­­­– это белорусское приложение, которое позволяет пользователям покупать и продавать товары и услуги.

Данное приложение имеет ряд плюсов:

* удобный фильтр поиска;
* много предложений со стороны исполнителей;
* чат на прямую с исполнителем;
* кроссплатформенность.

Но при этом Kufarимеет ряд минусов:

* отсутствует проверка соответствия категории, что может привести к появлению лишних предложений, не относящихся к запрошенной услуге;
* отсутствует регулировка цен, что может быть несправедливо для пользователей, которые могут переплатить за услугу;
* отсутствует проверка выполнения услуги, что может повлиять на качество услуги;
* рабочие, предлагающие услуги на сайте, не связаны с ним никакими контрактами или договоренностями, кроме прохождения регистрации.

В целом, этот сайт предоставляет удобный фильтр поиска и много предложений, но также имеет ряд недостатков, которые могут повлиять на его использование.

В заключение, хочется отметить, что существует множество аналогов приложений по оказанию бытовых услуг, и каждое из них имеет свои недостатки и преимущества.

## Анализ современного языка программирования JavaScript

JavaScript является одним из наиболее популярных языков программирования на сегодняшний день. Он используется для создания интерактивных веб-сайтов, приложений и игр. JavaScript также может быть использован для создания программного обеспечения на стороне сервера, с помощью платформы Node.js [1].

JavaScript был создан Бренданом Эйхом в 1995 году. С тех пор язык претерпел множество изменений и доработок, что позволило ему стать гибким и мощным инструментом для разработки приложений. В настоящее время JavaScript является основным языком программирования для веб–разработки, и это не удивительно, учитывая его преимущества и возможности.

Одной из основных причин популярности JavaScript является его возможность работать вместе с другими языками и технологиями, такими как HTML и CSS. Это позволяет разработчикам создавать динамические и интерактивные веб–сайты с помощью одного языка программирования.

JavaScript также имеет множество библиотек и фреймворков, которые значительно упрощают разработку приложений. Некоторые из наиболее популярных библиотек и фреймворков, которые используются в настоящее время, включают React, Angular и Vue.js.

Одним из ключевых преимуществ JavaScript является его возможность асинхронного программирования. Это означает, что код может выполняться параллельно, что повышает производительность и ускоряет работу приложений. Это особенно полезно для приложений, которые работают с большими объемами данных или с сетевыми запросами.

JavaScript также имеет низкий порог вхождения для новых разработчиков благодаря доступности обучающих материалов и огромному сообществу разработчиков. Это делает язык доступным для всех, кто хочет научиться программированию и создавать свои собственные приложения.

Однако, JavaScript также имеет свои недостатки. Например, язык может быть склонен к ошибкам из–за сложности синтаксиса и особенностей работы с объектами. Это может привести к проблемам с производительностью и безопасностью приложений.

JavaScript также имеет ряд особенностей, которые могут быть сложными для понимания и использования. Например, в языке есть механизмы области видимости и подъема переменных, которые могут быть запутывающими для новичков. Кроме того, JavaScript имеет динамическую типизацию, что может привести к ошибкам, если не учитывать типы данных при написании кода.

Одним из способов улучшения безопасности приложений, написанных на JavaScript, является использование строгого режима ("use strict"). Этот режим предотвращает некоторые ошибки, которые могут привести к уязвимостям в приложениях, и улучшает производительность.

Несмотря на некоторые недостатки, JavaScript остается одним из самых востребованных языков программирования в мире. Кроме того, его популярность продолжает расти, поскольку язык постоянно развивается и улучшается. Новые версии языка, такие как ECMAScript 2015 (также известная как ES6) и ECMAScript 2020, добавляют новые функции и возможности, что делает язык еще более мощным и гибким.

Несмотря на то, что JavaScript является одним из самых распространенных языков программирования, он все еще имеет ряд вызовов, с которыми сталкиваются разработчики при создании сложных приложений. Один из главных вызовов состоит в том, что JavaScript является языком программирования, который работает в браузере, что означает, что он выполняется на клиентской стороне, а не на сервере. Это означает, что веб–разработчики должны быть особенно внимательными при написании кода, чтобы убедиться, что он будет работать в различных браузерах и операционных системах [2].

Кроме того, JavaScript может столкнуться с проблемами производительности при работе с большими объемами данных, особенно при работе в браузере. Одним из способов решения этой проблемы является использование библиотек и фреймворков, таких как React или Angular, которые помогают оптимизировать производительность и улучшить пользовательский опыт.

Также важно учитывать безопасность при создании приложений на JavaScript. JavaScript может быть подвержен атакам, таким как инъекция кода или межсайтовая подделка запросов (CSRF). Разработчики должны принимать меры для защиты своих приложений от этих угроз, используя соответствующие практики безопасности, такие как валидация ввода, фильтрация и защита от XSS-атак.

В целом, JavaScript является мощным языком программирования, который имеет широкий диапазон применений. Он может быть использован для создания интерактивных пользовательских интерфейсов, веб-приложений, игр, мобильных приложений и много другого. Кроме того, благодаря библиотекам и фреймворкам, таким как React, Angular и Vue.js, JavaScript становится еще более мощным инструментом для разработки сложных приложений.

## Анализ используемых технологий для реализации поставленной задачи

Для разработки курсового приложения был выбран стек MERN.

Стек – это сочетание технологий, используемых для создания веб–приложения. Любое веб-приложение будет создано с использованием нескольких технологий (фреймворки, библиотеки, базы данных и т. д.).

Стек MERN – это набор технологий, который используется для создания современных веб–приложений. MERN состоит из четырех основных компонентов: MongoDB, Express, React и Node.js. Эти технологии обеспечивают высокую производительность, расширяемость и масштабируемость приложений [3].

MongoDB – это NoSQL база данных, которая использует документоориентированный подход. Это означает, что данные хранятся в виде документов, которые могут содержать различные поля и структуры. MongoDB обеспечивает гибкость при работе с данными и позволяет легко масштабировать базу данных при необходимости. Кроме того, MongoDB имеет мощный язык запросов, который облегчает работу с данными.

Express – это фреймворк для создания веб-приложений на Node.js. Express предоставляет множество функций для обработки запросов и ответов, обеспечивая тем самым быстрое и надежное выполнение задач. Express также имеет обширную библиотеку плагинов, которые могут использоваться для улучшения функциональности приложений.

React – это JavaScript-библиотека для создания пользовательских интерфейсов. React облегчает разработку динамических веб–приложений, предоставляя различные компоненты, которые могут быть использованы для создания комплексных интерфейсов. React также имеет широкую экосистему, которая включает в себя множество плагинов и библиотек, что делает его еще более мощным инструментом для создания веб–приложений.

Node.js – это серверная среда, которая позволяет запускать JavaScript на стороне сервера. Node.js обеспечивает быстрое и надежное выполнение задач, а также позволяет легко масштабировать приложения при необходимости. Node.js также имеет обширную библиотеку модулей, которые могут использоваться для улучшения функциональности приложений.

Совместно, эти четыре компонента образуют мощный стек технологий для создания веб-приложений. В рамках стека MERN, MongoDB используется в качестве базы данных, Express обеспечивает веб-серверную логику, React используется для создания пользовательского интерфейса, а Node.js используется для запуска приложения на стороне сервера.

Одним из преимуществ использования стека MERN является то, что все компоненты стека используют JavaScript. Это обеспечивает легкую интеграцию между компонентами, а также упрощает разработку приложений для разработчиков, которые уже знакомы с JavaScript. Более того, стек MERN обеспечивает гибкость в разработке приложений, позволяя комбинировать различные инструменты и технологии в зависимости от требований проекта.

Создание приложения на стеке MERN начинается с создания базы данных в MongoDB и определения ее структуры. Затем веб-серверная логика определяется в Express, включая маршруты и контроллеры. После этого создаются React-компоненты для создания пользовательского интерфейса и обработки пользовательских действий.

Node.js используется для запуска приложения на стороне сервера. Для этого используются специальные модули, которые обеспечивают взаимодействие между компонентами стека MERN. Взаимодействие между компонентами осуществляется посредством API-запросов, которые обрабатываются в Express [4].

После создания приложения на стеке MERN, его можно развернуть на хостинге или облачной платформе. Для этого обычно используются специальные инструменты, такие как Heroku или AWS Elastic Beanstalk.

Стек MERN имеет множество преимуществ. Он позволяет разработчикам создавать современные веб–приложения быстро и эффективно. Кроме того, стек MERN обеспечивает высокую производительность приложений и легко масштабируется при необходимости. Более того, использование JavaScript для всех компонентов стека делает его легко доступным и удобным для разработчиков, которые уже знакомы с этим языком программирования.

Однако, использование стека MERN имеет и свои недостатки. Например, MongoDB не поддерживает транзакции в традиционном понимании, что может быть проблемой для некоторых проектов. Кроме того, стек MERN может быть сложным для новичков, которые не знакомы со всеми компонентами. Однако, справившись с этими проблемами, разработчики могут создавать высокопроизводительные и масштабируемость.

Более подробно рассмотрим каждый из компонентов стека MERN:

* MongoDB: документоориентированная база данных, которая хранит данные в формате BSON (Binary JSON), что позволяет эффективно обрабатывать данные в сравнении с традиционными реляционными базами данных. Она также обладает гибким схематическим подходом, который позволяет хранить данные различных форматов, а также добавлять новые поля в документы без изменения всей структуры базы данных. Это позволяет разработчикам быстро адаптировать базу данных к изменяющимся требованиям проекта;
* Express: минималистичный фреймворк для создания серверных приложений на Node.js. Он предоставляет широкий спектр инструментов и возможностей для обработки HTTP-запросов, обработки маршрутов, управления сеансами пользователей и многое другое. Он также легко расширяемый, позволяя разработчикам добавлять новые модули и функциональность для своих приложений;
* React: библиотека JavaScript для создания пользовательских интерфейсов. Он позволяет разработчикам создавать динамические и быстрые пользовательские интерфейсы с помощью компонентов. React также использует виртуальный DOM, что ускоряет процесс обновления пользовательского интерфейса и улучшает производительность приложения. Благодаря своей популярности, в React есть огромное сообщество, которое предоставляет множество инструментов и ресурсов для разработчиков;
* Node.js: среда выполнения JavaScript на стороне сервера. Он позволяет использовать JavaScript для разработки серверных приложений, что обеспечивает легкую интеграцию между серверной и клиентской логикой. Node.js также обладает высокой производительностью благодаря своему асинхронному и неблокирующему подходу, что позволяет обрабатывать большие объемы данных и запросов с минимальной задержкой.

Стек MERN предоставляет разработчикам все необходимые инструменты для создания современных веб-приложений. Он позволяет создавать приложения быстро и эффективно, обеспечивая высокую производительность и гибкость при разработке. Кроме того, он имеет большое сообщество разработчиков, что обеспечивает доступ к множеству инструментов и ресурсов для разработки.

Преимущества стека MERN:

* Высокая производительность: благодаря использованию MongoDB и Node.js, стек MERN обеспечивает высокую производительность и масштабируемость при обработке больших объемов данных и запросов;
* Гибкость: MongoDB предоставляет гибкую схематическую модель, что позволяет легко адаптировать базу данных к изменяющимся требованиям проекта. Express также предоставляет гибкий подход для обработки HTTP-запросов и управления маршрутами [5];
* Легкая интеграция: Node.js позволяет использовать JavaScript для разработки серверной логики, что обеспечивает легкую интеграцию между серверной и клиентской логикой;
* Компонентный подход: React использует компонентный подход для создания пользовательских интерфейсов, что обеспечивает легкость и переиспользуемость кода [6];
* Огромное сообщество: благодаря своей популярности, стек MERN имеет большое сообщество разработчиков, что обеспечивает доступ к множеству инструментов и ресурсов для разработки;
* Переносимость: приложения, созданные с помощью стека MERN, могут работать на различных операционных системах и устройствах, благодаря использованию JavaScript для разработки как клиентской, так и серверной логики.

В целом, стек MERN является мощным инструментом для создания современных веб-приложений. Он позволяет создавать приложения быстро и эффективно, обеспечивая высокую производительность и гибкость при разработке. Если вы хотите создать масштабируемое и гибкое веб-приложение, то стек MERN может быть отличным выбором.

## Анализ инструментальных средств автоматизации разработки и тестирования

StarUML – это инструмент для моделирования, который предоставляет графическую среду для создания различных типов диаграмм, таких как диаграммы классов, диаграммы последовательностей и диаграммы состояний.

StarUML позволяет пользователям создавать модели, которые могут использоваться для описания различных аспектов системы, таких как ее структура, поведение и взаимодействие с другими системами. С помощью StarUML можно создавать как малые, так и большие проекты, и он может быть использован во многих областях, включая программирование, разработку приложений и проектирование баз данных [7].

Одной из особенностей StarUML является то, что он поддерживает множество различных языков моделирования, включая UML (Unified Modeling Language), BPMN (Business Process Model and Notation), ERD (Entity-Relationship Diagram) и DFD (Data Flow Diagram).

В StarUML доступно несколько типов диаграмм, каждая из которых предназначена для определенных целей:

* Диаграммы классов: используются для описания структуры объектно-ориентированных систем, включая классы, атрибуты, методы и отношения между классами;
* Диаграммы последовательностей: используются для описания взаимодействия между объектами в системе во времени;
* Диаграммы состояний: используются для описания состояний объектов в системе и переходов между ними;
* Диаграммы деятельности: используются для описания бизнес-процессов и действий, не связанных непосредственно с объектами;
* Диаграммы компонентов: используются для описания компонентов системы и их взаимодействия;
* Диаграммы развертывания: используются для описания физической структуры системы и ее развертывания на аппаратном обеспечении.

Кроме того, StarUML предоставляет возможность экспортировать диаграммы в различные форматы, включая PNG, PDF, SVG и HTML.

В целом, StarUML – это мощный инструмент для моделирования, который позволяет пользователям создавать диаграммы, описывающие различные аспекты системы. Он поддерживает множество различных языков моделирования и может быть использован во многих областях, связанных с разработкой программного обеспечения.

Visual Studio Code (VS Code) – это бесплатный интегрированный среда разработки (IDE) от Microsoft, которая предоставляет удобный и гибкий интерфейс для создания, отладки и развертывания приложений. VS Code был выпущен в 2015 году и быстро стал одним из самых популярных инструментов для разработки программного обеспечения.

VS Code основан на платформе Electron и написан на языке программирования TypeScript. Он доступен для операционных систем Windows, macOS и Linux и поддерживает различные языки программирования, включая JavaScript, TypeScript, Python, Java, C++, C# и многие другие [8].

VS Code имеет множество функций и инструментов, которые делают его мощным инструментом для разработки. Некоторые из ключевых функций VS Code включают в себя:

* Редактор кода: VS Code предоставляет удобный и гибкий редактор кода, который поддерживает функции, такие как автодополнение, подсветку синтаксиса, форматирование и интеграцию с системами контроля версий;
* Отладка: VS Code позволяет пользователям отлаживать код, используя различные инструменты, такие как точки останова, просмотр переменных и консоль отладки;
* Интеграция с Git: VS Code имеет встроенную интеграцию с Git, что позволяет пользователям легко работать с системами контроля версий и управлять своими репозиториями;
* Расширения: VS Code поддерживает множество расширений, которые позволяют пользователям расширить функциональность IDE и добавить поддержку для различных языков программирования и инструментов;
* Интеграция с облачными сервисами: VS Code также имеет интеграцию с различными облачными сервисами, такими как Azure, AWS и Google Cloud Platform, что позволяет пользователям управлять своими облачными приложениями непосредственно из IDE;
* Терминал: VS Code имеет встроенный терминал, который позволяет пользователям работать с командной строкой и выполнить различные операции без выхода из IDE.

В целом, VS Code –это мощный инструмент для разработки программного обеспечения, который обеспечивает множество функций и инструментов для создания, отладки и развертывания приложений. Он поддерживает множество языков программирования, имеет множество расширений.

GitHub - это онлайн-платформа для хранения и управления версиями кода. Он был создан в 2008 году и быстро стал одним из самых популярных сервисов для хранения и совместной работы над проектами с открытым исходным кодом.

Кроме того, GitHub также является крупнейшей онлайн-библиотекой открытых исходных кодов, в которой разработчики могут находить и использовать уже существующие решения для своих проектов, а также делиться своими проектами с сообществом.

GitHub предоставляет бесплатные аккаунты с ограниченным количеством приватных репозиториев и возможностью бесплатно использовать открытые репозитории, что делает его доступным для разработчиков всех уровней. Также доступны платные тарифы для команд и компаний, которые позволяют получить дополнительные функции и возможности управления проектами [9].

Также GitHub предоставляет разработчикам множество инструментов для ыыуправления проектами и совместной работы, включая следующие функции:

* Репозитории: GitHub позволяет пользователям создавать и хранить репозитории для своих проектов. Каждый репозиторий содержит всю историю изменений, а также документацию и другие файлы проекта;
* Ветки: GitHub позволяет пользователям создавать отдельные ветки для разных версий проекта. Это позволяет разработчикам работать над различными функциями и исправлениями ошибок параллельно, без влияния на основную ветку;
* Pull-запросы: С помощью pull-запросов в GitHub пользователи могут предлагать изменения в проекте и предоставлять другим участникам возможность ознакомиться и обсудить эти изменения перед их включением в основную ветку;
* Проблемы и запросы на добавление функционала: GitHub предоставляет инструменты для создания и отслеживания проблем и запросов на добавление функционала, позволяя участникам сообщества легко сотрудничать и работать вместе над решением проблем и добавлении нового функционала;
* Интеграция с инструментами непрерывной интеграции и доставки: GitHub имеет множество интеграций с инструментами непрерывной интеграции и доставки, что позволяет разработчикам автоматизировать процессы сборки, тестирования и развертывания своих проектов;
* Инструменты для совместной работы над кодом: GitHub предоставляет различные инструменты для совместной работы над кодом, включая инструменты для рецензирования кода, инструменты для обнаружения конфликтов и инструменты для управления правами доступа;
* Расширения и интеграции: GitHub поддерживает множество расширений и интеграций, которые позволяют разработчикам расширить функциональность платформы и интегрировать ее с другими инструментами и сервисами.

В целом, GitHub – это мощная и удобная платформа для совместной работы над проектами, которая предоставляет разработчикам множество инструментов и функций для управления версиями кода, контроля доступа и совместной работы. Все это делает GitHub незаменимым инструментом для разработки программного обеспечения в команде или индивидуально.

MongoDB Compass – это инструмент для визуализации и управления данными в MongoDB, популярной NoSQL базе данных. Он предоставляет графический интерфейс для работы с данными, который упрощает взаимодействие с базой данных и увеличивает производительность работы.

MongoDB Compass имеет несколько основных функций, которые облегчают работу с данными. Одна из них – это возможность просматривать данные в удобном графическом интерфейсе. Это позволяет быстро находить нужные документы и просматривать их содержимое без необходимости использования сложных запросов.

Другая полезная функция – это возможность создания и редактирования документов прямо из MongoDB Compass. Это удобно, если вы хотите быстро создать новый документ или внести изменения в существующий без необходимости использования кода.

MongoDB Compass также имеет возможность создания и редактирования индексов, что позволяет оптимизировать производительность базы данных и улучшить ее скорость работы. Он также предоставляет возможность управления пользователями и настройки доступа к базе данных, что облегчает управление безопасностью и управлением доступом.

Наконец, MongoDB Compass имеет функцию анализа производительности, которая позволяет определить, какие запросы к базе данных занимают наибольшее время. Это помогает оптимизировать производительность базы данных и ускорить ее работу.

В целом, MongoDB Compass – это мощный инструмент для управления данными в MongoDB, который облегчает взаимодействие с базой данных и повышает производительность работы. Он предоставляет широкий набор функций для работы с данными, а также обладает простым и интуитивно понятным интерфейсом, что делает его доступным для использования как для опытных, так и для новичков в области баз данных.

Postman – это платформа для тестирования и разработки API, которая предоставляет множество инструментов для тестирования, отладки и документирования API. С его помощью разработчики могут упростить процесс разработки API, ускорить тестирование и улучшить качество их приложений.

Одной из главных функций Postman является создание запросов к API и их тестирование. Разработчики могут легко создавать запросы с помощью интуитивно понятного интерфейса и отправлять их на сервер для проверки. Postman поддерживает различные методы запросов, такие как GET, POST, PUT и DELETE, а также позволяет устанавливать заголовки и параметры запроса для более точного тестирования [10].

Postman также позволяет создавать коллекции запросов, что облегчает управление и тестирование большого количества запросов. Разработчики могут группировать запросы в коллекции и легко их запускать, что упрощает процесс тестирования и сокращает время, необходимое для тестирования.

Кроме того, Postman предоставляет возможность отслеживать результаты запросов и анализировать ответы сервера. Это позволяет быстро выявлять ошибки и улучшать качество приложений.

Postman также имеет функцию автоматической генерации документации для API, которая упрощает процесс создания документации для вашего API. Разработчики могут использовать Postman для создания автоматически генерируемой документации API, которая будет отображаться на вашем сайте или в вашем приложении.

В целом, Postman – это мощный инструмент для тестирования и разработки API, который позволяет разработчикам упростить и ускорить процесс разработки, тестирования и документирования API. Он обладает широким набором функций и интуитивно понятным интерфейсом, что делает его доступным для использования как для опытных, так и для новичков в области API.

## Техническое задание для программного продукта «Web-приложение по поддержке процессов бытового обслуживания населения»

Цель разработки: разработать веб-приложения, предназначенного для компании по оказанию услуг населению и их клиентов, которое позволит:

* создавать заказ;
* выбирать исполнителя этих заказов;
* общаться с исполнителем;
* контролировать оплату;
* выполнение заказа;
* оценивать результат выполнения.

Для исполнителей, приложения позволит:

* просматривать все заказы по их направлению;
* просматривать личные заказы;
* общаться с клиентом;
* ввести учёт выполнения;
* создание акта выполненных работ с возможностью его дополнения.

Также приложение будет иметь возможность получать следующие отчёты:

* доход за период;
* доходы по категориям;
* рейтинг мастеров.

Приложение должно иметь следующую структуру и функциональность:

* работать как веб-приложение, основанное на REST архитектуре;
* являться кроссплатформенным;
* быть доступным посредством веб-браузера;
* иметь пользовательский интерфейс.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И АЛГОРИТМЫ

## Анализ предметной области

Приложение по оказанию бытовых услуг населению должно предоставлять возможность заказа и оплаты услуг по дому, таких как уборка, глажка, ремонт и т.д. Для успешного функционирования приложения необходимо создать четкую структуру учета заказов и управления ими. Для этого требуется специализированное программное обеспечение, которое позволит автоматизировать основные процессы работы приложения и сделать их более прозрачными.

Одной из главных проблем, которые должно решать приложение, является обеспечение достоверности и доступности заказов. Это включает в себя информацию о доступных исполнителях услуг, адресе заказа, статусе выполнения заказа и т.д.

Для решения этих проблем необходимо создать жесткие нормативные требования к созданию муниципальных баз информационных ресурсов, которые будут автоматически собирать информацию о доступных исполнителях услуг, заказах, их выполнении и оплате. Также необходимо предоставить расчетно-сервисное обслуживание потребителей услуг по принципу «одного окна», чтобы обеспечить удобство и простоту использования приложения.

Для разработки приложения необходимо учитывать потребности и пожелания потенциальных пользователей. В настоящее время все большее количество людей предпочитает совершать покупки и заказы через мобильные приложения, поэтому важно обеспечить максимальную доступность и удобство использования приложения на различных мобильных платформах.

Кроме того, важно учитывать безопасность пользователей и конфиденциальность их данных. Для этого необходимо предусмотреть механизмы защиты персональной информации, а также создать надежную систему оплаты и обработки данных клиентов.

В целом, разработка приложения по оказанию бытовых услуг населению является актуальной задачей, которая позволяет существенно улучшить качество жизни людей и облегчить их бытовые заботы. Однако, для достижения успеха необходимо учитывать множество факторов, таких как удобство использования, безопасность и конфиденциальность, а также провести эффективную рекламную кампанию.

## Функциональная модель программного комплекса

Для обеспечения правильной работы всей системы программы доступ к её функционалу следует распределить по ролям. Таким образом явно выделяется 3 роли, которые будут взаимодействовать с программой:

* Клиент – это пользователи, которые заказывают услуги. Они должны иметь доступ к функциям заказа услуг, оплаты заказов, просмотра информации о доступных исполнителях услуг и их отзывов;
* Мастер – это исполнители услуг, которые получают заказы и выполняют их. Они должны иметь доступ к информации о новых заказах, их статусе, адресам и требованиям клиентов. Также они должны иметь возможность подтверждать или отклонять заказы и оставлять отзывы о выполненной работе;
* Менеджер – это пользователи, которые управляют системой и следят за её работой. Они должны иметь доступ ко всем функциям системы, включая управление клиентской базой данных, мастерами и заказами. Также они должны иметь возможность создавать отчеты и статистику по работе системы, чтобы улучшить её производительность и качество обслуживания клиентов.

Для создания работоспособной и качественной ИС, нужно понимать, что требуется для осуществления в данной предметной области. Необходимо изучить и осознать логические связи, а также объекты данной проектируемой области. Для достижения такого результата нам поможет UML-диаграмма.

Для создания модели UML диаграммы была использовано программное обеспечение StarUML.

UML диаграмма состоит из актеров и прецедентов, где актер – пользователь, которому доступны сервисы приложения, а прецедент – основные сервисы, доступные данному пользователю (актеру).

Подытожив всё вышеперечисленное, получим следующую UML-диаграмму (рисунок 2.1):

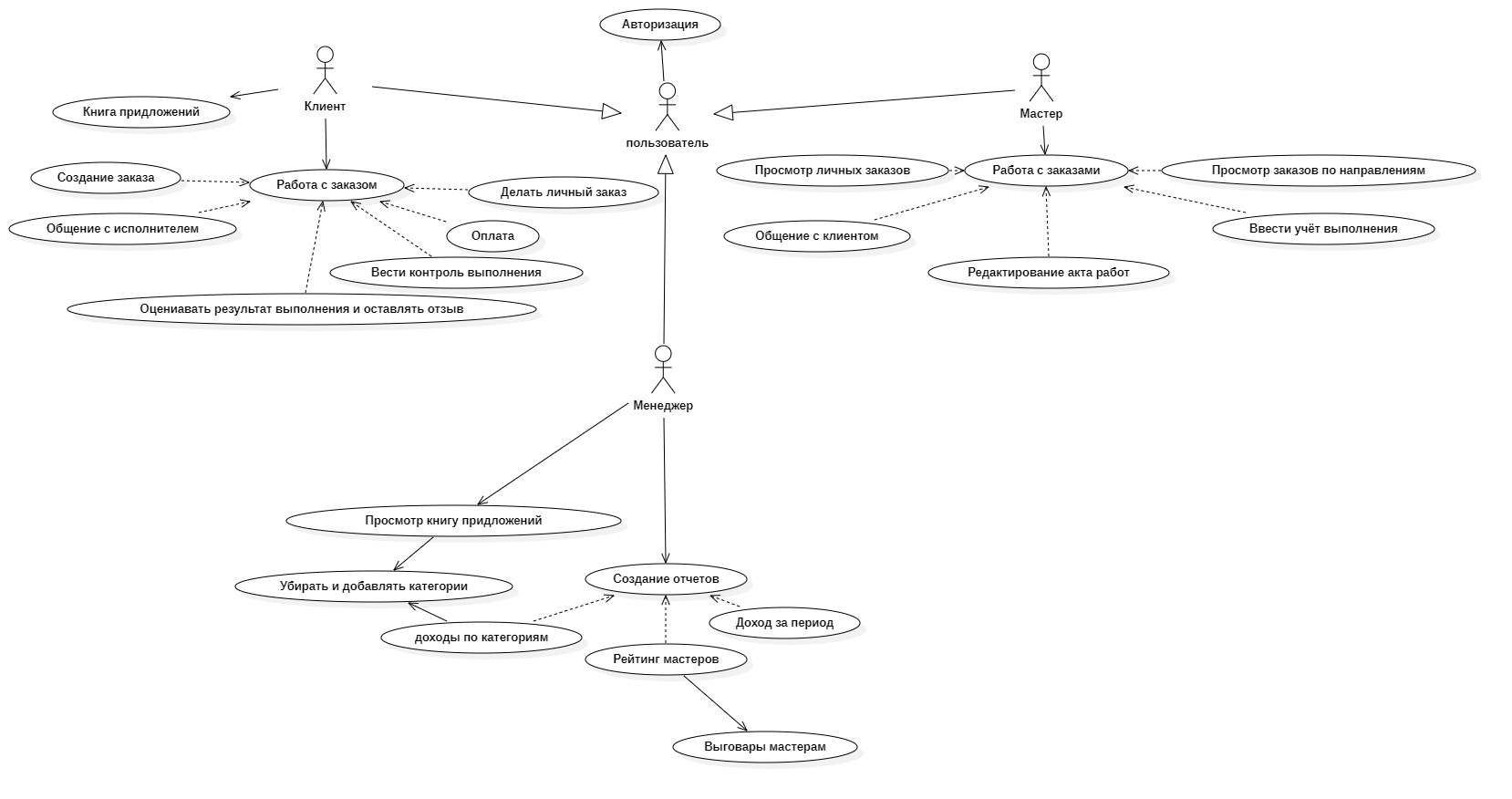


Рисунок 2.1 – Диаграмма прецедентов

Далее приведены преценденты для роли «Клиент».

Прецедент «Добавление записи в книгу предложений» должен реализовать функцию, для добавления пожелай пользователя для улучшения приложения.

Прецедент «Создание заказа» должен реализовать комплекс функций, для заполнения информации о заказе и его создание.

Прецедент «Общение с исполнителем» должен реализовать комплекс функций, для внутреннего общения между клиентом и исполнителем.

Прецедент «Создание личного заказа» должен реализовать комплекс функций, для создания заказа от клиента к конкретному исполнителю.

Прецедент «Контроль выполнения» должен реализовать комплекс функций, позволяющих клиенту вести контроль выполнения.

Прецедент «Оценивать» должен реализовать комплекс функций, для оценивания результата выполнения работы и оставления отзыва.

Прецедент «Оплата» должен реализовать комплекс функций, для оплаты клиентом заказа.

Далее приведены преценденты для роли «Мастер»

Прецедент «Просмотр личных заказов» должен реализовать комплекс функций, для просмотра личных заказов к мастеру от клиентов.

Прецедент «Просмотр заказов по направлению» должен реализовать комплекс функций, для просмотра заказов по направлениям данного мастера.

Прецедент «Общение с клиентом» должен реализовать комплекс функций, для внутреннего общения между мастером и клиентом.

Прецедент «Учёт выполнения» должен реализовать комплекс функций, для учёта выполненных работ мастером в заказе.

Прецедент «Редактировать акт работ» должен реализовать комплекс функций, для добавления или удаления выполненных работ в заказе.

Далее приведены преценденты для роли «Менеджер»

Прецедент «Просмотр книги предложений» должен реализовать комплекс функций, для просмотра менеджером всех записей и их сортировку в книге предложений.

Прецедент «Отчёты» должен реализовать комплекс функций, для просмотра и создания отчётом о работе приложения.

Прецедент «Работа с услугами» должен реализовать комплекс функций, для добавления или удаления услуг, предоставляемых приложением.

Важно понимать, что прецеденты отражают взгляд на систему со стороны, поэтому загромождать диаграмм UML является недопустимым, так как большое количество прецедентов может запутать читающего.

## Информационная модель программного комплекса

Изучив предметную область, были определены наборы сущностей и их свойств, а затем спроектирована база данных (рисунок 2.1).

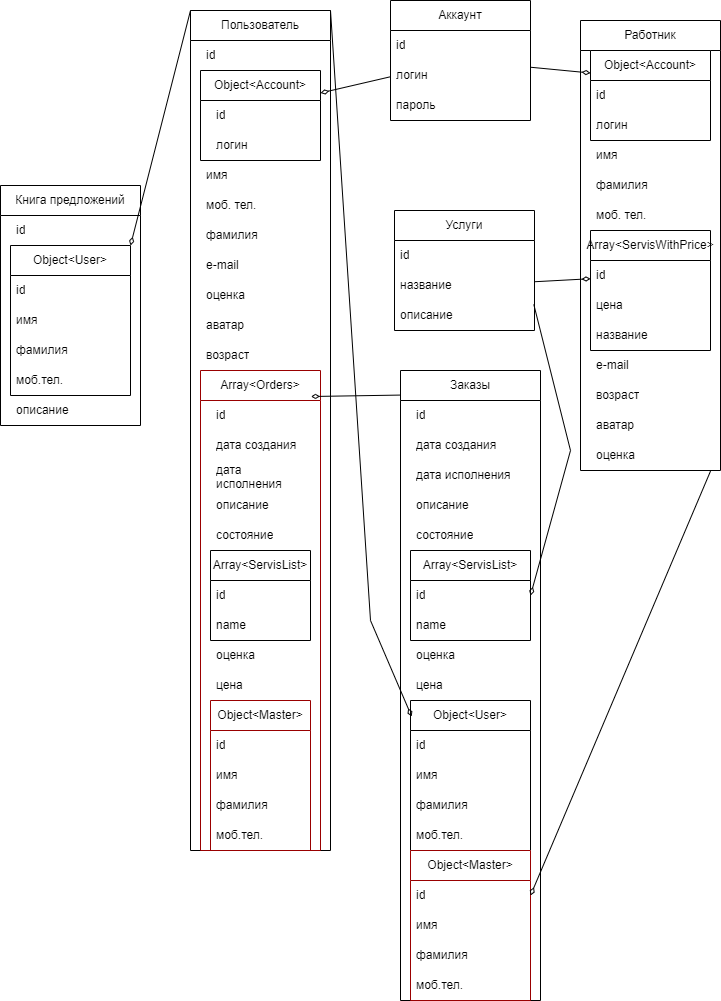


Рисунок 2.2 – Схема базы данных

Для представления информационной модели разрабатываемого продукта были выделены следующие сущности:

* сущность «Аккаунт» хранит в себе информацию о всех аккаунтах;
* сущность «Пользователь» хранить в себе всё информацию о клиентах и их заказов;
* сущность «Работник» хранит в себе информацию о всех мастерах и менеджеров;
* сущность «Услуги» хранит в себе все услуги, которые предоставляет приложение;
* сущность «Заказы» хранит в себе всю подробную информацию о заказах;
* сущность «Книга предложений» хранит в себе записи предложений по улучшению приложения о пользователей;

Физическая модель базы данных определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Построение физической модели базы данных производится на основе логической модели, которая и была представлена выше.

В таблице 2.1 описано соответствие сущностей логической модели и таблиц физической. Подробное описание каждой сущности приведено в таблицах 2.2 – 2.7.

Таблица 2.1 – Соответствие сущностей логической и физической моделей

|  |  |
| --- | --- |
| Сущность | Коллекция |
| Аккаунт | Account |
| Пользователь | Client |
| Работник | Employee |
| Услуги | Services |
| Заказы | Order |
| Книга предложений | Book |

Таблица 2.2 – Коллекция Account

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | ObjectId | false | true |

Продолжение таблицы 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| login | string | false | true |
| password | string | Fasle | false |

В таблице 2.2 поле login имеет уникальное значение, так как, для большей информативности и безопасности, пользователи будут работать с ним, вместо поля ID.

Таблица 2.3 – Коллекция Servis

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | ObjectId | false | true |
| название | string | false | true |
| описание | string | fasle | false |

В таблице 2.3 предоставлена коллекция Servis, которая хранит в себе все услуги, которые предоставляет приложение.

Таблица 2.4 – Коллекция Book

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | ObjectId | false | true |
| User | object | false | false |
| User.\_id | ObjectId | false | true |
| User.name | string | false | false |
| User.surname | string | false | false |

Продолжение таблицы 2.4

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| User.modile | string | false | true |
| описание | string | false | false |

В таблице 2.4 поле User имеет тип object, который предоставляет всё нужную информацию о пользователе.

Таблица 2.5 – Коллекция Employee

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | ObjectId | false | true |
| name | string | false | false |
| surname | string | fasle | false |
| mob. num. | string | false | true |
| e-mail | string | false | true |
| age | number | false | false |
| avatar | image | true | false |
| range | number | false | false |
| Account | object | false | true |
| Account.\_id | ObjectId | false | true |
| Account.login | string | false | true |
| ServisWithPrice | Array<object> | false | false |
| ServisWithPrice.\_id | ObjectId | false | true |

Продолжение таблицы 2.5

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| ServisWithPrice.name | string | false | false |
| ServisWithPrice.price | number | false | false |

В таблице 2.5 предоставлена коллекция Employee, которая хранит в себе всех сотрудников, в зависимости от наличия поля массива объектов ServisWithPrice, различаются мастера и менеджеры. Также коллекция хранит объект Account, который хранит нужную информацию об аккаунте работника.

Таблица 2.6 – Коллекция Client

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | ObjectId | false | true |
| name | string | false | false |
| surname | string | fasle | false |
| mob. num. | string | false | true |
| e-mail | string | false | true |
| age | number | false | false |
| avatar | image | true | false |
| range | number | false | false |
| Account | object | false | true |
| Account.\_id | ObjectId | false | true |
| Account.login | string | false | true |
| Orders | Array<object> | true | false |

Продолжение таблицы 2.6

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| Orders.\_id | ObjectId | false | true |
| Orders.start\_date | date | false | false |
| Orsers.end\_date | date | false | false |
| Orders.descripction | string | false | false |
| Orders.status | string | false | false |
| Orders.grade | number | true | false |
| Orders.price | number | false | false |
| Orders.ServisList | Array<object> | false | false |
| Orders.ServisList.\_id | ObjectId | false | true |
| Orders.ServisList.name | string | false | false |
| Orders.Master | object | true | false |
| Orders.Master.\_id | ObjectId | false | true |
| Orders.Master.name | string | false | false |
| Orders.Master.surname | string | false | false |
| Orders.Master.mob | string | false | false |

В таблице 2.6 предоставлена коллекция Client, которая хранит в себе всех клиентов. Коллекция хранит объект Account, который хранит нужную информацию об аккаунте работника, также она хранит объект Orders, который хранит информацию о заказах клиента.

Таблица 2.7 – Коллекция Order

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| id | ObjectId | false | true |
| start\_date | date | false | false |
| end\_date | date | false | false |
| descripction | string | false | false |
| status | union(“ожидает” | “принят” | “выполняется” | “ожидает оплаты” |  “выполнен” | “отменён”) | false | false |
| grade | number | true | false |
| price | number | false | false |
| ServisList | Array<object> | false | false |
| ServisList.\_id | ObjectId | false | true |
| ServisList.name | string | false | false |
| Master | object | true | false |
| Master.\_id | ObjectId | false | true |
| Master.name | string | false | false |
| Master.surname | string | false | false |
| Master.mob | string | false | false |
| Client | object | true | false |
| Client.\_id | ObjectId | false | true |

Продолжение таблицы 2.7

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| Client.name | string | false | false |
| Client.surname | string | false | false |
| Client.mob | string | false | false |

В таблице 2.6 предоставлена коллекция Orders, которая хранит в себе все заказы. Коллекция хранит объект Client, который хранит нужную информацию об клиенте, также она хранит объект Master, который хранит информацию о мастере этот объект не обязателен, т.к. в начале заказа может не быть мастера. Также хранится массив объектов ServisList, который хранит все услуги в заказе. Поле status имеет уникальный тип union из TypeScript, который позволят хранить в поле одно значение из списка.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Флэнаган, Д.: JavaScript. Подробное руководство/ Д. Флэнаган; под общ. ред. А.Галунов. – М.: Символ-Плюс, 2016. - 1072 с.
2. Закас, Н.: JavaScript для профессиональных веб-разработчиков/ Н. Закас; под общ. ред. Д.Загагулин- СПб.: Питер, 2013. - 768 с.
3. Гордон, А.: MERN: руководство по полной разработке веб-приложений/ А. Гордон; под общ. ред. Л.Мирнов –М.: БХВ-Петербург, 2020. – 624 с.
4. Сайт https://metanit.com. [Электронный ресурс] – Режим доступа: https://metanit.com/web/nodejs/.
5. Сайт https://itproger.com. [Электронный ресурс] – Режим доступа: https://itproger.com/course/mean.
6. Сайт https://ru.reactjs.org. [Электронный ресурс] – Режим доступа: https://ru.reactjs.org/docs/getting-started.html.
7. Сайт https://freeanalogs.ru. [Электронный ресурс] – Режим доступа: https://freeanalogs.ru/StarUML.
8. Сайт https://microsoft.com. [Электронный ресурс] – Режим доступа: https://visualstudio.microsoft.com/ru/.
9. Сайт https://habr.com/ru. [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/714538/.
10. Сайт https://sky.pro. [Электронный ресурс] – Режим доступа: https://sky.pro/media/postman-kak-polzovatsya/.