МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Специализация 1-40 01 01-10 Программное обеспечение информационных технологий (программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

Web-приложение «Сервис предоставления услуг»

Выполнил студент Корнелюк Валентин Владимирович

(Ф.И.О.)

Руководитель проекта ст. преп. О. А. Нистюк

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. В. В. Смелов

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2025

Содержание

[ВВЕДЕНИЕ 3](#_Toc193855568)

[1 Постановка задачи и обхор аналогичных решений 4](#_Toc193855569)

[1.1 Обзор аналогичных решений 4](#_Toc193855570)

[1.2 Постановка задачи 6](#_Toc193855571)

[2 Проектирование web-приложения 8](#_Toc193855572)

[2.1 Архитектура приложения 8](#_Toc193855573)

[2.2 Проектирование структуры базы данных 9](#_Toc193855574)

[ПРИЛОЖЕНИЕ А 14](#_Toc193855575)

# ВВЕДЕНИЕ

Тема разработки приложения "Сервис предоставления услуг" является весьма актуальной и востребованной в наше время. С развитием технологий и увеличением числа предпочитающих онлайн-сервисы пользователей, эффективное управление записями на услуги и расписанием становится важным аспектом для многих профессионалов и клиентов в сфере здоровья, красоты и других услуг.

Анализ существующих приложений в данной области показывает, что существующие решения имеют свои преимущества, однако они часто не обеспечивают полного удовлетворения потребностей пользователей. С ростом конкуренции и требований клиентов появляется необходимость в инновационных и удобных в использовании решениях.

Целью данного проекта является создание и реализация приложения, которое позволит клиентам удобно записываться на услуги и мастерам эффективно управлять своим расписанием. Главной целью приложения будет обеспечение удобства, простоты использования и эффективности в организации записей и управлении временем.

В основе разработки приложения лежат несколько ключевых принципов, направленных на достижение поставленных целей. Во-первых, приложение должно быть удобным в использовании и иметь интуитивно понятный интерфейс как для клиентов, так и для мастеров. Во-вторых, оно должно обеспечивать простой и быстрый процесс записи на услуги и управления расписанием. В-третьих, приложение должно поддерживать эффективное взаимодействие между клиентами и мастерами, предоставляя удобные средства коммуникации и обмена информацией.

В результате разработки и внедрения приложения, соответствующего вышеуказанным принципам и требованиям, пользователи смогут более эффективно организовывать свое время и управлять услугами, что в конечном итоге приведет к повышению удовлетворенности клиентов.

# Постановка задачи и обхор аналогичных решений

## Обзор аналогичных решений

Одним из первых этапов в создании программного продукта является анализ прототипов и литературных источников.

При анализе было обращено внимание на отсутствие идентичных аналогов разрабатываемому приложению. Вместо этого, были обнаружены приложения, частично содержащие функционал, схожий с предполагаемым функционалом разрабатываемого приложения. В данном разделе будет проведено сравнение и анализ указанных приложений с целью изучения их функционала и возможностей в контексте нашего приложения. Это позволит выявить преимущества и недостатки существующих решений, а также определить уникальные особенности и добавленную ценность разрабатываемого приложения.

Первый аналог – «fokanail.by**»**. На рисунке 1.1 изображён интерфейс сайта:

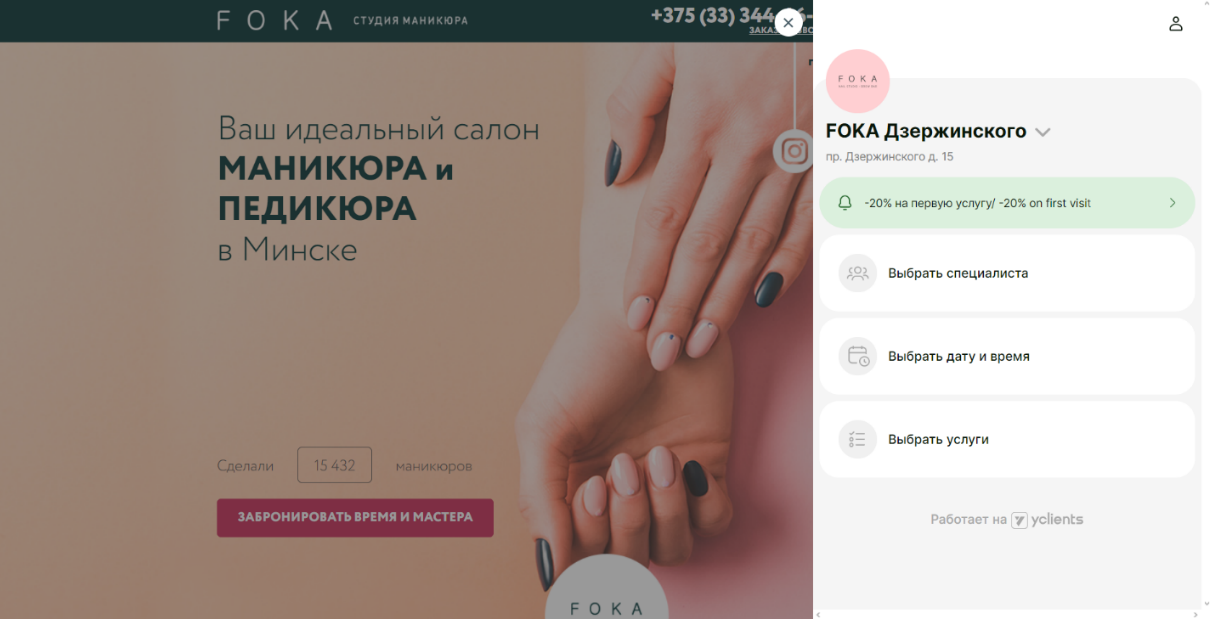
****

Рисунок 1.1 – Интерфейс приложения «foknail»

На сайте было обращено внимание на несколько удобных и функциональных аспектов организации записи на услуги:

* Интуитивный интерфейс для записи на услуги: Сайт предлагает интуитивно понятный интерфейс для записи на услуги, что делает процесс выбора и бронирования услуг быстрым и удобным для клиентов. Четкое разделение услуг и простая навигация облегчают процесс выбора;
* Удобный календарь для выбора даты и времени: Календарь, предоставляемый на сайте, позволяет клиентам легко выбирать удобные даты и время для записи на услуги. Интерактивный календарь с возможностью выбора дат и времени привлекает внимание и облегчает процесс выбора оптимального времени;
* Понятный процесс выбора услуг: Структура сайта делает процесс выбора услуг простым и прозрачным. Ясное представление доступных услуг и их описаний позволяет клиентам быстро найти нужную услугу и ознакомиться с ее описанием перед бронированием;
* Гибкие опции для персонализации записи: Возможность выбора дополнительных опций или параметров при записи на услуги (например, выбор мастера или дополнительных услуг) позволяет клиентам настроить бронирование под свои индивидуальные предпочтения и потребности.

В целом, приложение fokanail.by студии FOKA предоставляет клиентам удобный и интуитивно понятный интерфейс для записи на услуги, где процесс выбора услуг, даты и времени осуществляется легко и эффективно. Гибкие опции и информативное представление услуг обеспечивают удовлетворение потребностей клиентов и делают процесс записи на услуги приятным и беззаботным.

В качестве второго аналога рассмотрим приложение «DIKIDI». На рисунке 1.2 представлен интерфейс приложения:

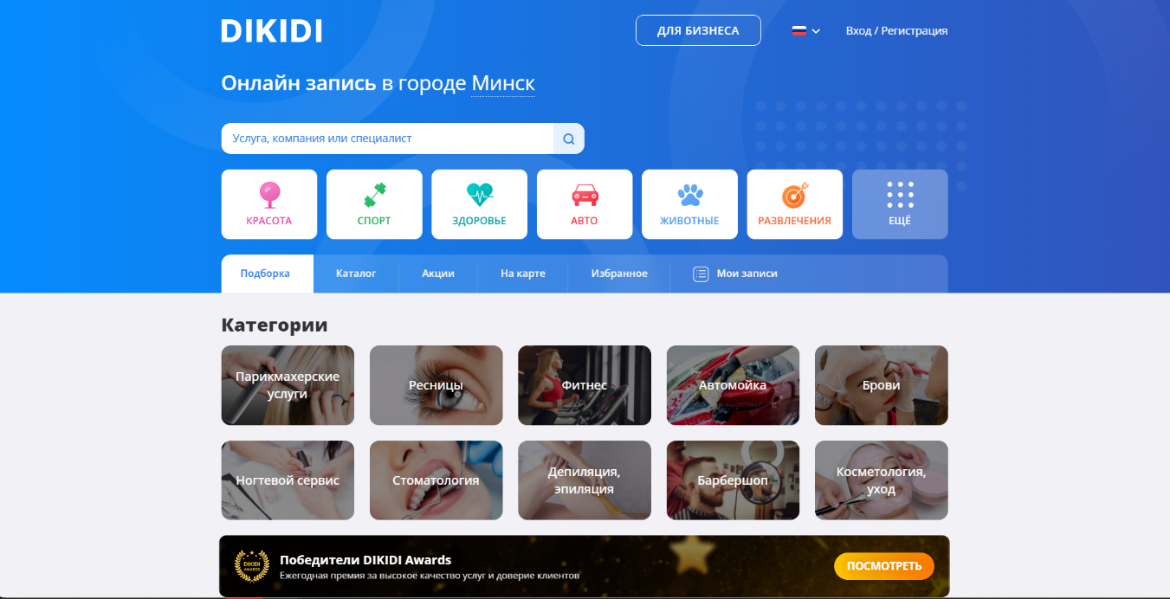


Рисунок 1.2 – Интерфейс приложения «DIKIDI»

При рассмотрении сайта были отмечены следующие плюсы:

* Простой процесс выбора услуг по подборкам: структура сайта обеспечивает легкость и ясность при выборе услуг. Понятное представление доступных услуг и их описаний позволяет клиентам быстро найти нужную услугу и ознакомиться с ее деталями;
* Гибкие возможности персонализации записи: Клиентам предоставляется возможность выбора дополнительных опций или параметров при записи на услуги, таких как выбор мастера или добавление дополнительных услуг. Это позволяет настраивать бронирование в соответствии с индивидуальными предпочтениями и потребностями клиентов.

В целом, приложение предоставляет клиентам простой и интуитивно понятный интерфейс для записи на услуги. Здесь процесс выбора услуг, даты и времени осуществляется легко и эффективно. Гибкие опции и информативное представление услуг удовлетворяют потребности клиентов, делая процесс записи приятным и беззаботным.

В качестве третьего аналога рассмотрим приложение **«**Instagram». Функционал приложения представлен на рисунке 1.3. В данном приложении есть две интересующие нас функции, это механизм подписки и функция создания публикаций.

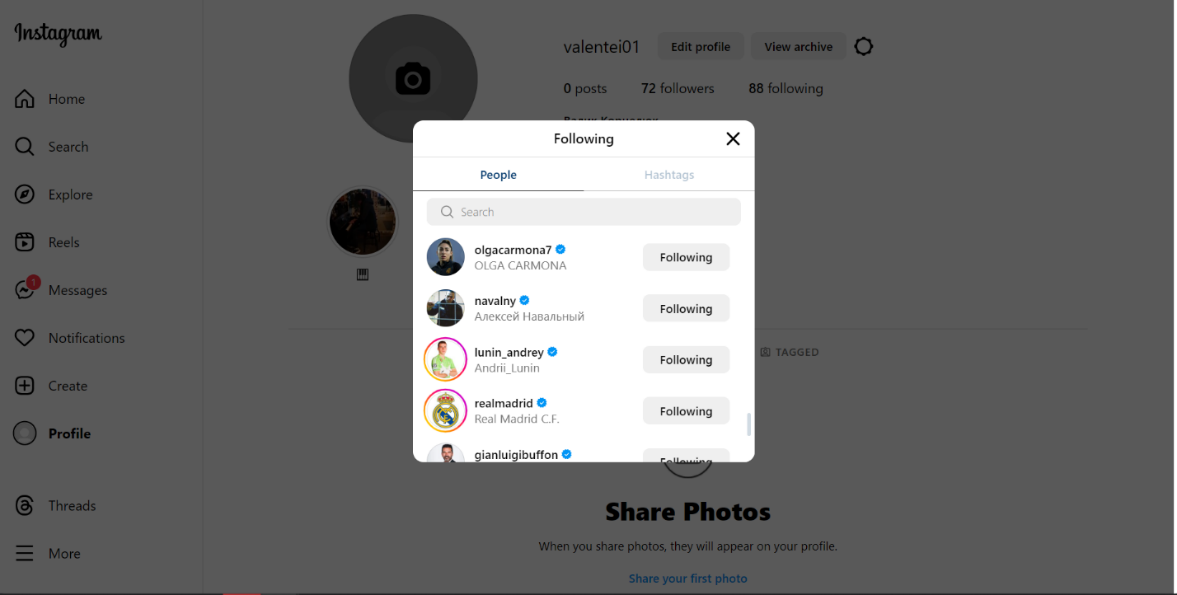


Рисунок 1.3 – Интерфейс приложения «Instagram»

При анализе этого приложения было обращено внимание на следующие аспекты, которые могут быть применены к организации аккаунтов пользователей и механизму подписки в разрабатываемом приложении:

* Механизм подписки на другие аккаунты: в Instagram пользователи могут подписываться на аккаунты других пользователей, чтобы получать обновления и контент. Эта функция может быть реализована в разрабатываемом приложении для возможности пользователям подписываться на профили мастеров, чтобы держать наиболее часто посещаемых мастеров в быстром доступе;
* Возможность публикации постов: в Instagram пользователи имеют возможность публиковать посты, включающие в себя фотографии, видео или текстовые материалы. Эта функция позволяет пользователям делиться своим творчеством, работами или полезной информацией с аудиторией. Аналогичная функциональность может быть реализована в разрабатываемом приложении для мастеров, позволяя им публиковать фотографии своих работ, видеоуроки или советы по уходу. Это поможет мастерам продвигать свои услуги, привлекать новых клиентов и установить более тесные отношения с существующей аудиторией.

Instagram предоставляет эффективный механизм для создания и управления пользовательскими аккаунтами, а также возможность подписки на другие аккаунты для получения актуальной информации и контента. Эти функциональные возможности могут быть адаптированы и внедрены в разрабатываемое приложение для улучшения взаимодействия пользователей и повышения их удовлетворенности.

## Постановка задачи

Основываясь на анализе аналогичных решений из пункта 1.1, проектируемое программное средство должно включать следующие характеристики и функциональные возможности.

Интуитивный интерфейс: приложение должно иметь простой и интуитивно понятный интерфейс, который позволит пользователям легко и быстро записываться на услуги. Это включает в себя понятную навигацию, минимальное количество шагов для записи и удобное расположение основных элементов.

Личный кабинет пользователя: каждому клиенту и мастеру должен быть доступен личный кабинет, где они могут просматривать свои записи, изменять их, просматривать историю предыдущих записей и т.д.

Возможность подписки на мастера: пользователь должен иметь возможность подписаться на мастера, чтобы иметь в быстром доступе аккаунты наиболее часто посещаемых мастеров. Кроме того, пользователь должен иметь возможность управлять своим списком подписок, добавлять или удалять мастеров.

Управление мастерами: администратору приложения должны быть доступна функциональность добавления новых мастеров.

Уведомления и напоминания: приложение должно отправлять уведомления и напоминания пользователям о предстоящих записях. Это помогает пользователям не забывать о своих записях и быть в курсе актуальной информации.

Безопасность и защита данных: приложение должно обеспечивать высокий уровень безопасности и защиты данных пользователей. Это включает шифрование личной информации и защиту от несанкционированного.

О выполнении задач, чтобы помочь в управлении проектами и принятии информированных решений.

Суммируя, проектируемое программное средство должно обладать удобным интерфейсом, возможностью создания и управления профилем, возможностью записи на услуги, подписки на мастеров, управления текущими записями. Учитывая эти характеристики, новое программное средство поможет клиента автоматизировать процесс записи на различные услуги, а мастерам эффективнее организовать рабочий процесс.

2 Проектирование web-приложения

2.1 Архитектура приложения

Серверная часть веб-приложения реализована с использованием фреймворка Express.js, который предоставляет REST API для взаимодействия между клиентом и сервером. REST API (Representational State Transfer) представляет собой архитектурный стиль, который определяет принципы построения веб-сервисов с использованием стандартных методов HTTP.

Клиентская часть проекта использует React.js, который представляет собой библиотеку JavaScript для разработки динамичных пользовательских интерфейсов. React.js позволяет создавать переиспользуемые компоненты, которые эффективно обновляются и реагируют на изменения данных. Это достигается за счет использования виртуального DOM (Virtual DOM), который позволяет React обновлять только необходимые элементы интерфейса без перерисовки всего дерева компонентов. Такой подход улучшает производительность взаимодействия с динамическими интерфейсами и значительно упрощает процесс разработки.

Взаимодействие между клиентской и серверной частями осуществляется посредством Web API. Сервер предоставляет API, определяющий доступные конечные точки и форматы данных для обмена информацией с клиентским приложением. Клиентское приложение, построенное на React.js, отправляет HTTP-запросы на сервер, используя соответствующие конечные точки, и принимает ответы, обычно в формате JSON. Это обеспечивает надежное взаимодействие между клиентом и сервером, позволяя передавать данные и обновления между ними и обеспечивая функциональность и интерактивность всего приложения.

Для аутентификации и авторизации пользователей в приложении используется сервер Keycloak. Он предоставляет централизованное управление доступом, поддержку OpenID Connect и OAuth 2.0, а также возможность настройки ролевой модели и политик безопасности. Keycloak упрощает управление пользователями и их правами доступа, обеспечивая надежную защиту данных.

Для хранения данных используется база данных PostgreSQL, которая является мощной объектно-реляционной СУБД. Она обеспечивает надежность, масштабируемость и расширяемость, поддерживает ACID-транзакции, имеет мощную систему индексации, механизмы репликации и авторизации, а также возможность создания собственных типов данных, операторов и функций. В приложении для работы с базой данных используется ORM Sequelize, что упрощает управление данными, позволяет выполнять операции CRUD и работать с моделями на уровне кода без необходимости написания сложных SQL-запросов.

Для хранения изображений в приложении используется облачное хранилище Dropbox. Это позволяет безопасно загружать, хранить и получать доступ к файлам с различных устройств, а также обеспечивает масштабируемость и надежность хранения данных.

Также в приложении используется протокол WebSockets. Использование WebSockets в web-приложении позволяет создать интерактивный и динамичный пользовательский опыт. WebSockets обеспечивают постоянное соединение между клиентом и сервером, что позволяет мгновенно передавать данные в обоих направлениях без необходимости постоянного обновления страницы.

Диаграмма развертывания представлена в приложении А.

2.2 Проектирование структуры базы данных

В данном разделе представлена логическая модель базы данных для проекта, а также описаны таблицы и связи, созданные в базе данных. Логическая схема базы данных представлена на рисунке 2.1.

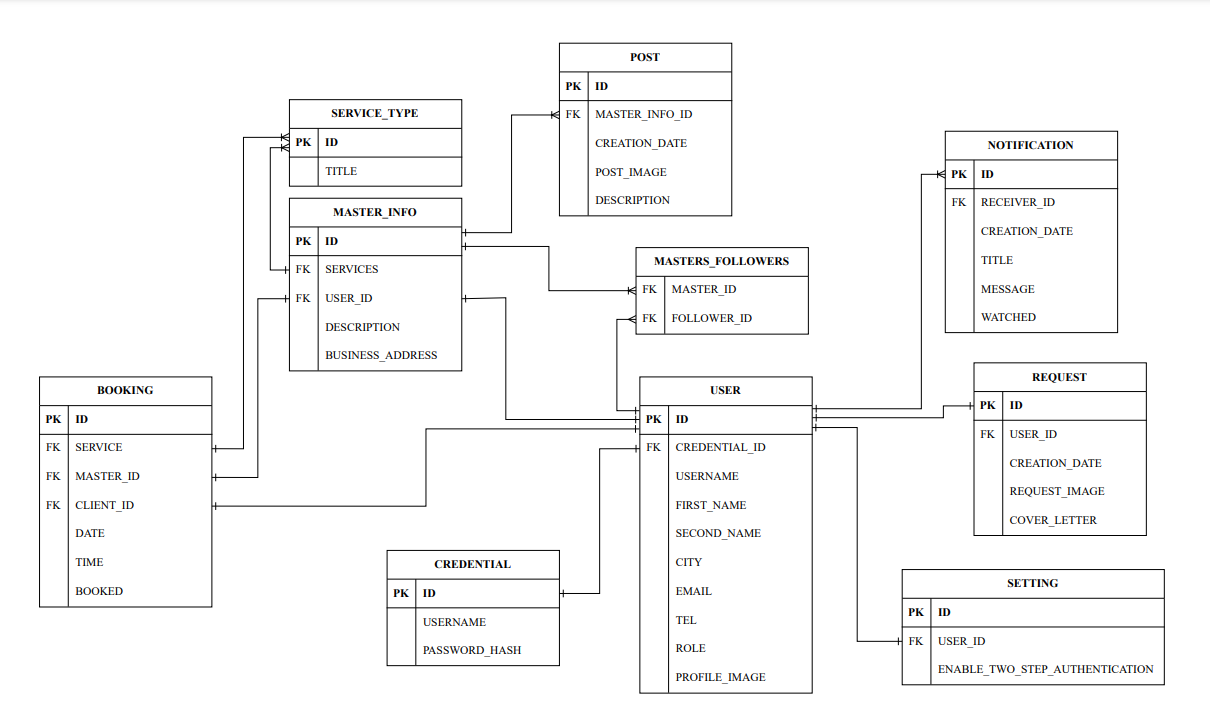


Рисунок 2.1 – Логическая схема базы данных

Проектирование базы данных играет ключевую роль в обеспечении эффективности и устойчивости работы любой системы. Разработка структуры таблиц, связей между ними и их характеристик должна учитывать, как текущие, так и потенциальные будущие требования к системе. Грамотно спроектированная база данных обеспечивает упрощение работы с данными и улучшает производительность запросов и повышает их читаемость.

В базе данных представлены 9 таблиц: USERS, CREDENTIAL, MASTER\_INFO, BOOKING, NOTIFICATION, POST, REQUEST, SETTING, MASTERS\_FOLLOWERS.

Таблица USERS хранит информацию о пользователях. Структура представлена в таблице 2.1:

Таблица 2.1 – Столбцы таблицы USERS

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Ограничение целостности |
| Id | NUMBER | PRIMARY KEY |
| Username | VARCHAR2(255) | NOT NULL, Unique |
| First\_name | VARCHAR2(255) | NOT NULL |
| Second\_name | VARCHAR2(255) | NOT NULL, UNIQUE |
| City | VARCHAR2(100) | NOT NULL |
| email | VARCHAR2(255) | DEFAULT |
| Tel | VARCHAR2(20) | NOT NULL |
| Role | VARCHAR2(50) | NOT NULL |
| Profile\_image | BLOB |  |
| Credential\_id | NUMBER | NOT NULL |
| Master\_info\_id | NUMBER |  |

Таблица MASTER\_INFO содержит дополнительную информацию о пользователе. Структура представлена в таблице 2.2

Таблица 2.2 – Столбцы таблицы MASTER\_INFO

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Ограничение целостности |
| Id | NUMBER | PRIMARY KEY |
| description | VARCHAR2(255) | NOT NULL |
| business\_address | VARCHAR2(255) | NOT NULL |

Таблица CREDENTIAL содержит информацию об учетных данных пользователя. Структура представлена в таблице 2.3

Таблица 2.3 – Столбцы таблицы CREDENTIAL

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Ограничение целостности |
| Id | NUMBER | PRIMARY KEY |
| username | VARCHAR2(255) | NOT NULL |
| password\_hash | VARCHAR2(255) | NOT NULL |

Таблица POST хранит информацию о постах мастеров. Структура представлена в таблице 2.4:

Таблица 2.4 – Столбцы таблицы POST

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Ограничение целостности |
| Id | NUMBER | PRIMARY KEY |
| creation\_date | DATE | NOT NULL |
| post\_image | BLOB | NOT NULL |
| description | VARCHAR2(255) |  |
| master\_info\_id | NUMBER | NOT NULL |

Таблица BOOKING хранит информацию о временных ячейках для записей на услуги. Она позволяет организовать удобное управление расписанием мастеров и клиентов. Данные таблицы используются для бронирования записей и анализа загруженности. Структура представлена в таблице 2.5:

Таблица 2.5 – Столбцы таблицы BOOKING

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Ограничение целостности |
| Id | NUMBER | PRIMARY KEY |
| client\_id | NUMBER |  |
| master\_id | NUMBER | NOT NULL |
| service | VARCHAR2(255) |  |
| booking\_date | DATE | NOT NULL |
| time | TIMESTAMP | NOT NULL |
| booked | NUMBER(1) | CHECK (booked IN (0, 1) |

Таблица NOTIFICATION хранит информацию об уведомлениях пользователей. Данные из этой таблицы используются для отправки актуальных уведомлений и ведения их истории для каждого пользователя. Структура представлена в таблице 2.6:

Таблица 2.6 – Столбцы таблицы NOTIFICATION

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Ограничение целостности |
| Id | NUMBER | PRIMARY KEY |
| receiver\_id | NUMBER | NOT NULL |
| creation\_date | DATE | NOT NULL |
| message | VARCHAR2(255) | NOT NULL |
| title | VARCHAR2(255) | NOT NULL |
| time | TIMESTAMP | NOT NULL |
| watched | NUMBER(1) | CHECK (watched IN (0, 1) |

Таблица REQUEST хранит информацию об запросах пользователей на то чтобы стать мастером. Эти данные позволяют администратору эффективно обрабатывать заявки и управлять назначением новых мастеров. Структура представлена в таблице 2.7:

Таблица 2.7 – Столбцы таблицы REQUEST

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Ограничение целостности |
| Id | NUMBER | PRIMARY KEY |
| user\_id | NUMBER | NOT NULL |
| creation\_date | DATE | NOT NULL |
| request\_image | BLOB | NOT NULL |
| cover\_letter | VARCHAR2(255) | NOT NULL |

Таблица SETTING хранит информацию о настройках приложения для пользователя. Данная таблица позволяет индивидуально настраивать параметры работы приложения для каждого пользователя, обеспечивая персонализированный опыт использования. Благодаря этому повышается удобство взаимодействия с системой и удовлетворение потребностей различных категорий пользователей. Структура представлена в таблице 2.8:

Таблица 2.8 – Столбцы таблицы SETTING

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Ограничение целостности |
| Id | NUMBER | PRIMARY KEY |
| account\_id | NUMBER | NOT NULL |
| enable\_two\_step\_authentication | NUMBER(1) | CHECK IN (0, 1) |

Таблица MASTERS\_FOLLOSERS хранит информацию о пользователеях и мастеров, на которых они подписаны. Эта таблица обеспечивает возможность отслеживания подписок пользователей на мастеров, что позволяет организовать удобное взаимодействие пользователей. Структура представлена в таблице 2.9:

Таблица 2.9 – Столбцы таблицы MASTERS\_FOLLOWERS

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Ограничение целостности |
| client\_id | NUMBER | NOT NULL |
| master\_id | NUMBER | NOT NULL |

Представленная структура базы данных обеспечивает эффективное и надежное хранение информации, необходимой для функционирования системы. Предусмотренные ограничения целостности и связи между таблицами гарантируют целостность данных и согласованность информации в системе.

Далее в таблице 2.10 будут представлено описание связей между таблицами базы данных.

Таблица 2.10 – Описание связей таблиц

|  |  |  |
| --- | --- | --- |
| Наименование столбца первичного ключа | Наименование столбца внешнего ключа | Описание связи |
| BOOKING.CLIENT\_ID | USERS.ID | Связь многие к одному для связи клиента и его записей |
| BOOKING.MASTER\_ID | MASTER\_INFO.ID | Связь многие к одному для связи мастера и его записей |
| POST.MASTER\_INFO\_ID | MASTER\_INFO.ID | Связь многие к одному для связи мастера и его постов |
| USERS.MASTER\_INFO\_ID | MASTER\_INFO.ID | Связь один к одному для связи общей информации пользователя и деталей информации о мастере |
| USERS.CREDENTIAL\_ID | CREDENTIAL.ID | Связь один к одному для связи информации пользователя с его учетными данными |
| SETTING.ACCOUNT\_ID | USERS.ID | Связь один к одному для связи информации пользователя и настроек приложения для него |
| REQUEST.USER\_ID | USERS.ID | Связь один к одному для связи запроса с пользователем от для которого он создан |
| NOTIFICATION.RECEIVER\_ID | USERS.ID | Связь многие к одному для связи пользователя с его уведомлениями |
| MASTERS\_FOLLOWERS.CLIENT\_ID | USERS.ID | Связь многие ко многим чтобы связать пользователя и его подписки |
| MASTERS\_FOLLOWERS.MASTER\_ID | USERS.ID | Связь многие ко многим чтобы связать мастера и его подписчиков |

Определение и реализация связей между таблицами обеспечивает целостность данных и эффективное выполнение запросов в базе данных. Такие связи помогают поддерживать логическую структуру системы, улучшая производительность и гарантируя согласованность информации.

В данной главе была спроектирована архитектура приложения, включая серверную и клиентскую часть, а также базу данных.

ПРИЛОЖЕНИЕ А

