МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Мобильное приложение “магазин строительных инструментов Мастак”»

Выполнил студент Макрица Фёдор Николаевич

(Ф.И.О.)

Руководитель проекта ст. преподаватель Блинова Е.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В . (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2023

**Содержание**

ВВЕДЕНИЕ 3

1. Постановка задачи и аналитический обзор литературы 4

1.1 Приложение «Wildberries» 4

1.2 Приложение «Ozon» 5

1.3 Выводы по главе «Обзор аналогов приложений» 7

2. Проектирование информационной системы 9

2.1 Локальная и глобальная база данных 9

2.2 Проектирование мобильного приложения 16

2.3 Проектирование сервера 19

2.4 Вывод по главе «Проектирование информационной системы» 20

3. Разработка информационной системы 21

3.1 Разработка базы данных 23

3.2 Разработка мобильного приложения 23

3.3 Разработка веб-приложения 26

3.2 Вывод по главе «Разработка информационной системы» 28

4. Обеспечение безопасности информационной системы 29

4.1 Вывод по главе «Обеспечение безопасности информационной системы» 30

Заключение 31

Список использованных источников 32

Приложение А 33

# ВВЕДЕНИЕ

С каждым годом интернет-технологии занимают все более важное место в повседневной жизни людей. Сейчас подавляющее большинство магазинов различных направлений имеют свои онлайн-версии. Это позволяет наиболее эффективным образом продавать и доставлять товары, а также удобно хранить данные о продажах. В сети можно купить практически все, но остается множество магазинов, которые только начинают свой онлайн-путь. В этом контексте можно отметить такие ниши рынка, как продажа профессионального строительного инструмента и оборудования. Мобильное приложение "Мастак" представляет собой магазин, специализирующийся на продаже строительных инструментов.

Мобильно приложение будет сделано с помощью Android Studio на языке Java. Так же будет использован Node.js для создания сервера на котором будет храниться база данных.

Будет возможность работать при отключении интернета. Приложение будет держать всю информацию, а после подключения интернета информация будет перенесена в базу данных и сохранена.

Таким образом, данное приложение очень полезно для многих людей, которым не обходимо купить строительные инструменты.

# Постановка задачи и аналитический обзор литературы

## 1.1 Приложение «Wildberries»

Важным этапом создания различных проектов является ознакомление с разных литературных источников и аналитический разбор.

Я проанализировал несколько разных программных средств связанных с моей тематикой.

Приложение Wildberries позволяет совершать интернет-покупки одежды, обуви и аксессуаров. У магазина Wildberries есть и свое приложение для смартфонов, которое очень удобно использовать. Самый большой плюс этого приложения – это простой и понятный интерфейс. В плане навигации и поиска товаров, приложение Wildberries оставляет только положительные эмоции. Большой ассортимент товаров, простая и понятная фильтрация, наличие акций и скидок – все это делает приложение Wildberries одним из лучших в виртуальном мире.

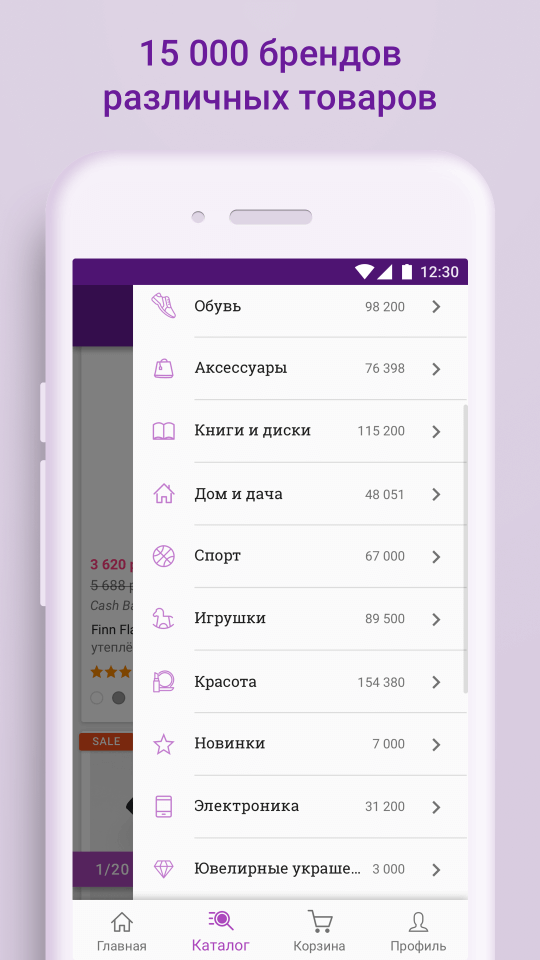


Рис. 1.1 – Мобильное приложение.

Данное приложение так же имеет и десктопнаю версию, которая показана на рисунке ниже (рисунок 1.2).

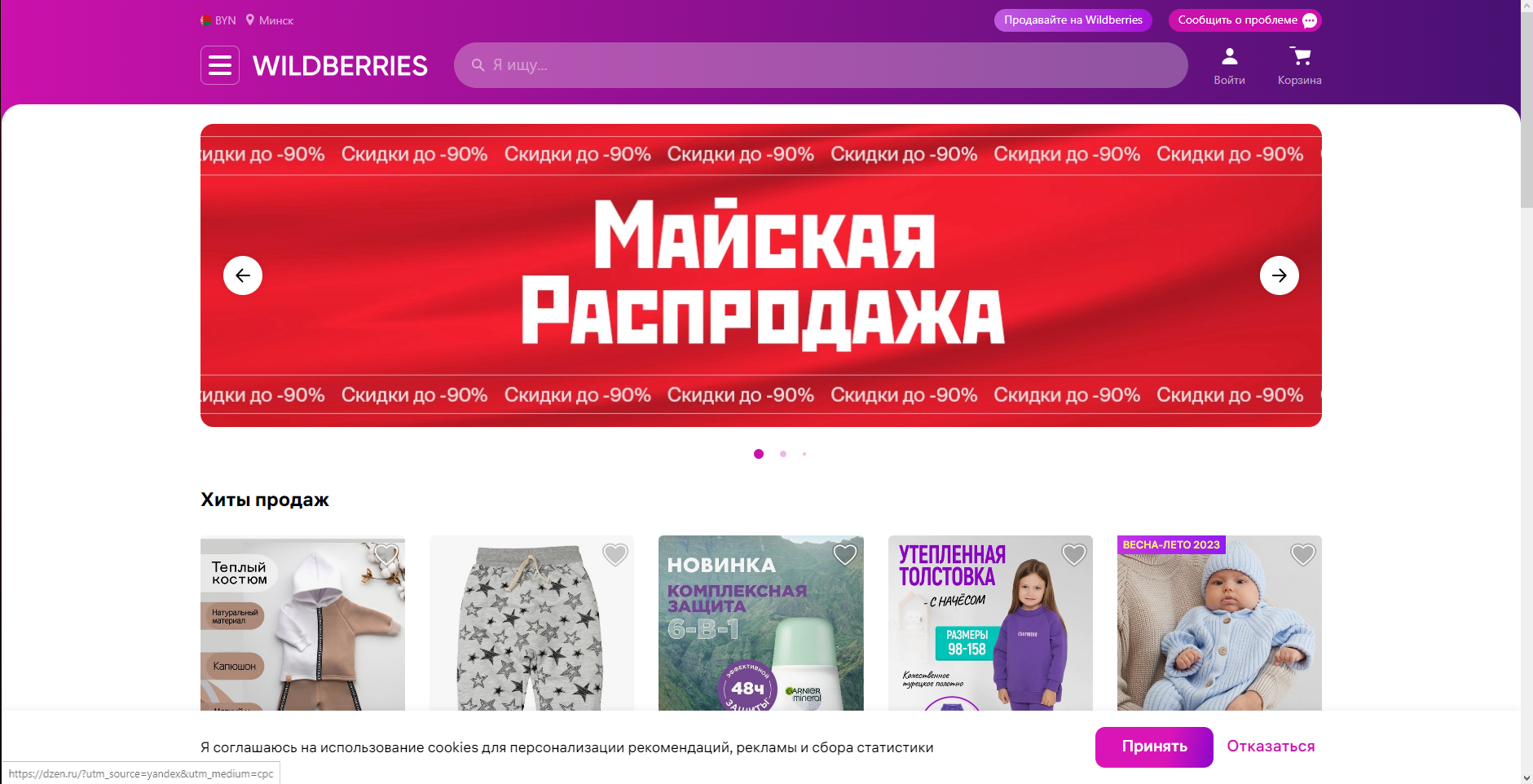


Рис. 1.2 – Версия для компьютеров.

## 1.2 Приложение «Ozon»

Есть еще более удобное приложение под название «Ozon».

Это многофункциональный интернет-магазин, который предлагает широкий ассортимент товаров, начиная от бытовой техники, продуктов питания, книг и заканчивая парфюмерией и косметикой. Один из недостатков данный магазин – это не слишком удобный интерфейс в приложении. Несмотря на это, если вы знаете, что хотите купить, на Ozon можно быстро найти то, что вам нужно, настроив при этом фильтры так, как вы считаете удобным. Также на сайте и в приложении Ozon есть множество акций и скидок, которые могут сохранить ваш бюджет. Удобной функцией является возможность оплаты заказа при получении товара и приложение может работать на разных телефонах с разными разрешениями экрана.

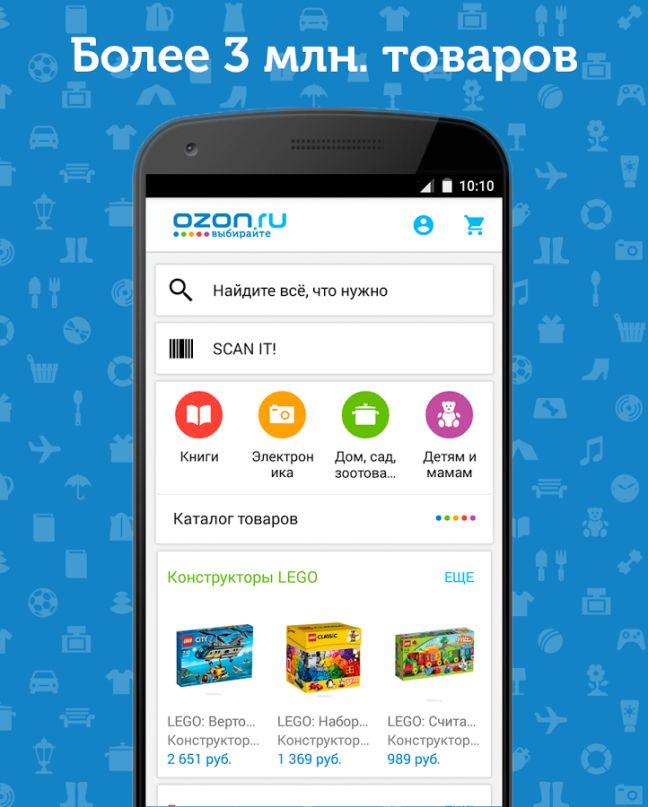


Рис. 1.3 – Ozon – дизайн под телефон.

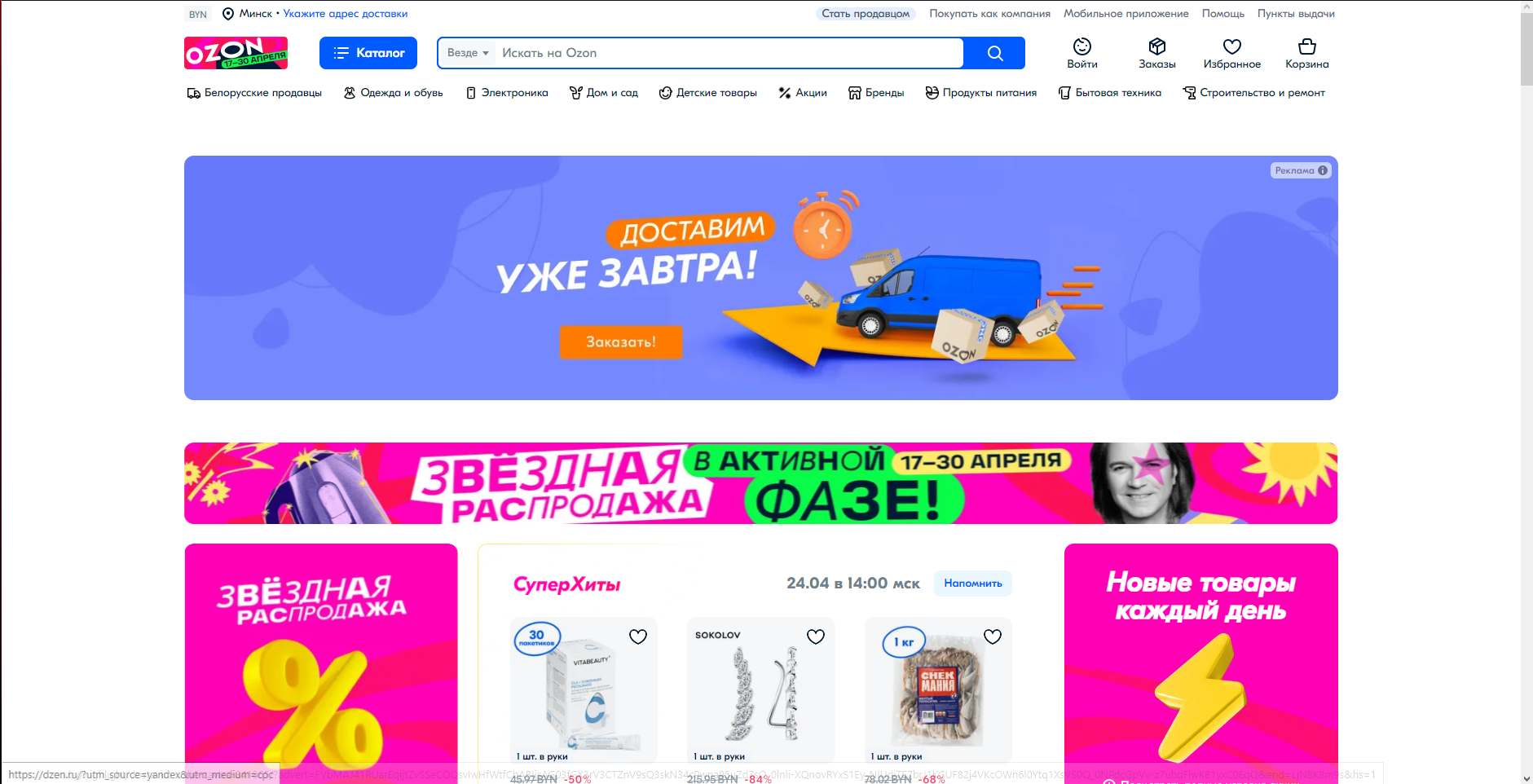


Рис. 1.4 – Интерфейс пользователя и администратора.

В моей программе будут сохранены товары в избранное, которые можно будут удалять. Это позволит не тратить время на поиск. Пользователи могут создать и авторизоваться в свою учетную запись.

## Выводы по главе «Обзор аналогов приложений»

В данной главе были показаны два популярных приложения, которые являются аналогами приложения «Мастак»: «Wildberries» и «Ozon». В результате подытожим сравнения.

Плюсы Wildberries:

- Простой интерфейс приложения, позволяющий быстро находить нужные товары;

- Возможность найти выгодные предложения и скидки на товары;

- Можно легко отслеживать состояние заказа и делать возврат товара.

- Подробный каталог товаров со свойствами, фото и отзывами покупателей;

- Возможность оставлять отзывы на товары, читать отзывы других покупателей;

- Различные способы оплаты, в том числе банковской картой, электронными деньгами, наличными;

- Удобный поиск товаров по различным параметрам.

Минусы Wildberries:

- Может быть сложно найти то, что вы ищете, из-за несоответствия категорий или трудности в поиске;

- Возможны сбои в системе начисления бонусов и их использования.

Плюсы Ozon:

- Простой интерфейс, который легко использовать и поисковую систему, которая работает быстро;

- Удобный поиск товаров по параметрам;

- Возможность выбирать из разных ценовых категорий и находить выгодные предложения;

- Покупки можно оплатить различными способами, в том числе банковской картой, электронными деньгами или наличными при получении.

Минусы Ozon:

- Некоторые пользователи жалуются на проблемы с поддержкой клиентов, особенно на перегруженность в период пиковых продаж.

- Длительная обработка возвратов и возможные сложности при их оформлении;

- Иногда цены на товары на сайте не соответствуют ценам в реальности, что вызывает разочарование

Таким образом, сравнение этих двух магазинов показало, что у каждого из них есть свои достоинства и недостатки. Wildberries позволяет с легкостью найти и заказать желаемый товар, обеспечивая при этом легкую навигацию, множество скидок и акций. Ozon же направлен на усовершенствование своего сервиса и расширение ассортимента товаров. Несмотря на особенности каждого магазина, оба из них предлагают высокое качество обслуживания. Выбор между ними зависит только от потребностей и предпочтений покупателя.

В моем приложении реализовано: возможность отслеживания состояние заказа, быстрый поиск, простой интерфейс, возможность оставлять отзывы на товары, читать отзывы других покупателей;

# Проектирование информационной системы

## 2.1 Локальная и глобальная база данных

Задачей курсового проекта создать приложение, которое будет иметь заданный функционал:

* Просмотр товара
* Заказ товара
* Управление заказанных товаров в корзине клиента
* Авторизация и регистрация аккаунта

В базе данных данного приложения созданы для использования 3 таблицы. Скрипт создания БД и таблиц находится в Приложении А. Ниже представлена их характеристика.

Схема базы данных представлена в таблице 2.1 – 2.9.

**Таблица 2.1- схема таблицы *Users.***

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Типы данных** | **Назначение** |
| id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| name | String | Имя пользователя |
| password | String | Пароль пользователя в хешированном виде |
| isAdmin | Bool | Определяет права пользователя |
| image | String | Изображение пользователя |

В таблице «Users» содержится информация для авторизации пользователей. Столбцы данной таблицы:

* id – уникальный идентификатор, который придумывает сам пользователь, является первичным ключом;
* name – строковый тип данных, максимальная длина 20;
* password – строковый тип данных, максимальной длины 12, хешируется на сервере и на мобильном приложении с помощью bcrypt
* isAdmin – булевый тип данных, определяющий права доступа
* Image – строковый тип данных, определяющий изображение пользователя(Приложение В);

**Таблица 2.2 – схема таблицы *Favorite*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| Favorite\_id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| User\_id | Integer | Идентификатор пользователя |
| Product\_id | Integer | Идентификатор мероприятия |

В таблице «***Favorite»*** содержатся пользователи и товары, на которые они хотят приобрести. Столбцы данной таблицы:

* id – целочисленный тип данных, первичный ключ, имеется AUTOINCREMENT позволяющий автоматически ставит id;
* user\_id – целочисленный тип данных, Внешний ключ, ссылается на первичный ключ таблицы Users, служит для связи этих таблиц;
* event\_id – целочисленный тип данных, Внешний ключ, ссылается на первичный ключ таблицы Product, служит для связи этих таблиц;

**Таблица 2.3 – схема таблицы *Product*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| product\_name | String | Название товара |
| price | Double | Цена товара |
| quantity | Integer | Количество товара |
| supplier | String | Поставщик товара |
| image | String | Изображение товара |
| category | String | Категория товара |

В таблице «***Product***» содержится информация о товаров. Столбцы данной таблицы:

* id – целочисленный тип данных, первичный ключ, имеется AUTOINCREMENT позволяющий автоматически ставит id товара;
* product\_name – строковый тип данных, не допускается null;
* price – строковый тип данных;
* quantity – целочисленный тип данных;
* supplier – содержит дату проведения мероприятия, тип данных дата, неуникальное значение, не допускается null, удаляется автоматически после истечения времени;
* image – строковый тип данных, содержит информацию о изображении;
* category – строковый тип данных;

**Таблица 2.4 – схема таблицы *Ordering*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| ordering\_id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| order\_number | String | Номер заказа |
| order\_date | Double | Дата заказа |
| status | Integer | Статус заказа |
| name\_on\_card | String | Имя карточки (имя на карточке) |
| card\_number | String | Номер карты |
| expiration\_date | Date | дата истечения срока действия |
| user\_id | Integer | Идентификатор пользователя |
| product\_id | Integer | Идентификатор мероприятия |

**Таблица 2.5 – схема таблицы *Cart*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| user\_id | Integer | Внешний ключ. Определяет уникальный идентификатор cart с user и ordering. |
| product\_id | Integer | Внешний ключ. Определяет уникальный идентификатор cart с product и ordering. |

**Таблица 2.6 – схема таблицы *History*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| history\_id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| user\_id | Integer | Внешний ключ. Определяет уникальный идентификатор cart с user и ordering. |
| product\_id | Integer | Внешний ключ. Определяет уникальный идентификатор cart с user и ordering. |

**Таблица 2.7 – схема таблицы *Shipping*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| shipping\_id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| address | String | Адрес пользователя |
| city | String | Город пользователя |
| country | String | Страна пользователя |
| zip | String | Почтовый индекс |
| phone | String | Номер телефона |
| user\_id | Integer | Внешний ключ. Определяет уникальный идентификатор cart с user и ordering. |
| product\_id | Integer | Внешний ключ. Определяет уникальный идентификатор cart с user и ordering. |

**Таблица 2.8 – схема таблицы *poster*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| poster\_id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| image | String | Имя картинки для poster |

**Таблица 2.9 – схема таблицы *review*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| review\_id | Integer | Первичный ключ. Определяет уникальный идентификатор записи. |
| user\_id | Integer | Внешний ключ. Определяет уникальный идентификатор cart с user и ordering. |
| product\_id | Integer | Внешний ключ. Определяет уникальный идентификатор cart с user и ordering. |
| feedback | Varchar (150) | обратная связь |
| rate | Float (11,1) | Оценка |
| review\_date | Date | Дата Оценки |

Схема глобальной базы данных представлена на рисунке 2.1

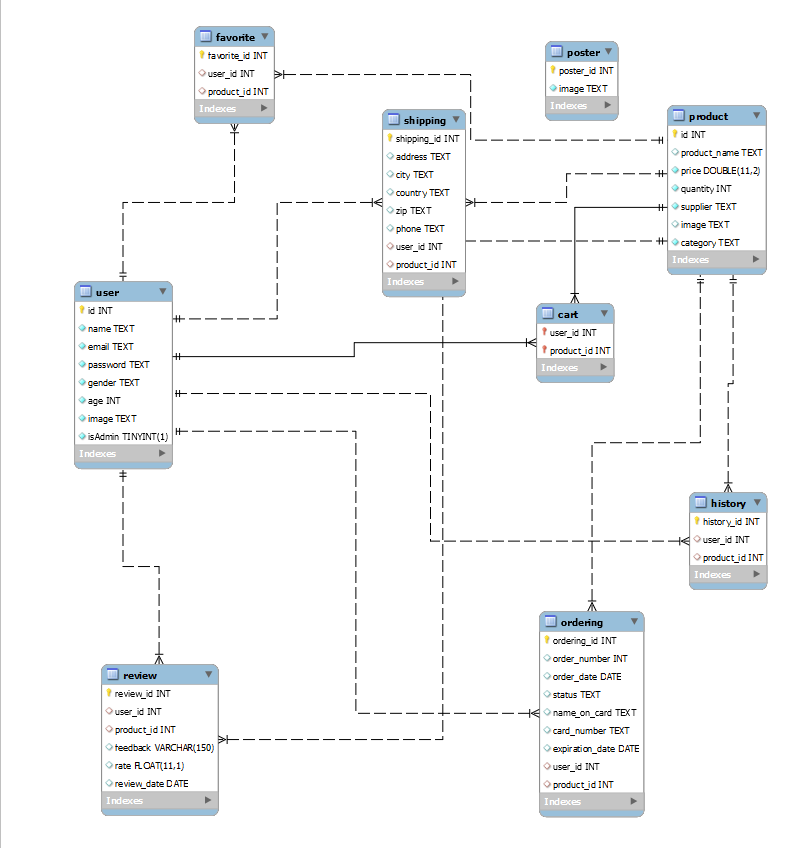
****

Рис. 2.1 – структура глобальной базы данных

Таблицы имеют внешние ключи, которые связывают таблицы друг с другом. Например, таблица "review" хранит информацию о пользовательских отзывах на продукты. Она содержит поля для хранения идентификатора отзыва, идентификатора пользователя, идентификатора продукта, текста отзыва, оценки продукта и даты отзыва. Внешние ключи на таблицы "user" и "product" связывают отзывы с соответствующими пользователями и продуктами.

Таблица "shipping" хранит информацию о доставке. Она содержит поля для хранения идентификатора доставки, адреса доставки, номера телефона, идентификатора пользователя и идентификатора продукта. Внешние ключи на таблицы "user" и "product" связывают доставку с соответствующими пользователями и продуктами.

Таким образом, эти таблицы обеспечивают функциональность приложения "Мастак", которое позволяет пользователям заказывать товары, оставлять отзывы, добавлять продукты в избранное, просматривать историю заказов и информацию о доставке.

Скрипт создания БД и таблиц находится в Приложении A.

Схема локальной базы данных представлена на рисунке 2.2

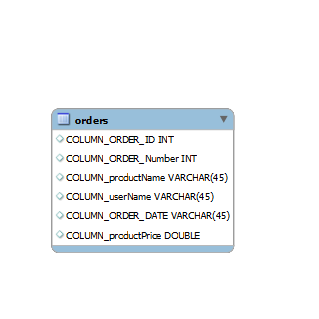


Рис. 2.2 – структура локальной базы данных

Как видно из рисунка 2.5 эта локальная база данных создается для хранения информации о заказах пользователей, если у приложения отсутствует соединение с интернетом. Она включает в себя a таблицу, которая имеет следующие столбцы:

- COLUMN\_ORDER\_ID — это уникальный идентификатор заказа.

- COLUMN\_ORDER\_Number — это номер заказа.

- COLUMN\_productName — это наименование продукта, который заказал пользователь.

- COLUMN\_userName — это имя пользователя, который заказал продукт.

- COLUMN\_ORDER\_DATE — это дата заказа.

- COLUMN\_productPrice — это стоимость продукта.

- COLUMN\_shippingAddress — это адрес доставки заказа.

- COLUMN\_ORDER\_STATUS — это статус заказа.

Эта локальная база данных позволяет приложению сохранять информацию о заказах пользователей, даже когда у приложения отсутствует соединение с интернетом. Это позволяет приложению сохранять информацию о заказах пользователя, которая может использоваться в будущем для целей аналитики и улучшения сервиса.

## 2.2 Проектирование мобильного приложения

В данной курсовой работе необходимо было разработать приложение, позволяющее пользователям совершать онлайн покупки инструментов.

Если пользователь запускает приложение на устройстве в первый раз, то открывается страница с именем магазина рисунок 2.3.

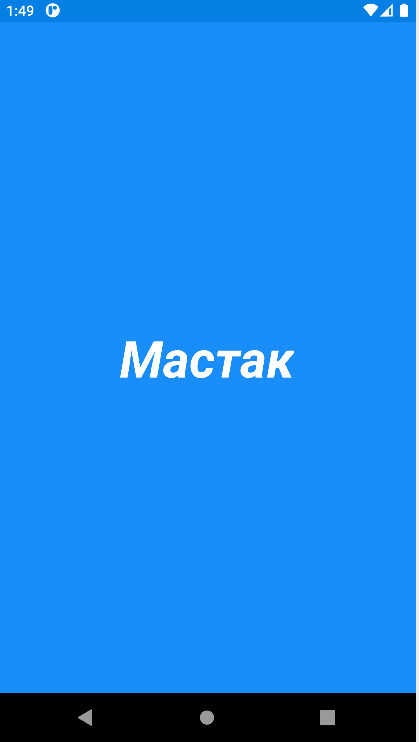


Рис. 2.3 – Пользователь запускает приложение

А затем идет переход к экрану регистрации, где у нас расположены поля ввода данных для таких данных как электронная почта и пароль, также на поле ввода пароля пользователь может нажать на кнопку с иконкой глаза чтобы увидеть пароль, который тот ввел. Внизу находиться кнопка регистрации, внизу которой ссылка на экран для создания учетной записи на рисунке 2.4 показан скриншот приложения.

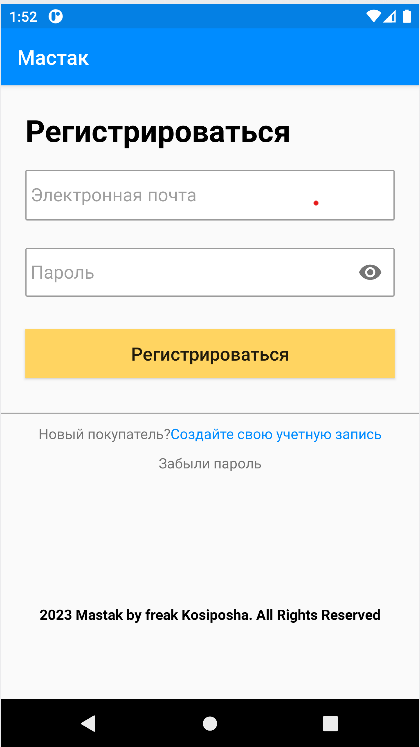
****

Рис. 2.4 – Страница регистрации

На экране регистрации в приложении доступна функция создания новой учетной записи. При регистрации нового пользователя, ему предоставляется возможность ввести свою электронную почту и пароль, который должен быть достаточной длины (не менее 8 символов). Если пользователь уже зарегистрирован в базе данных, то при попытке ввода его электронной почты на экране регистрации выскакивает соответствующее сообщение о том, что данный пользователь уже существует.

По умолчанию у всех пользователей приложения отсутствуют права на добавление товаров, такие права выдает администратор. Для получения прав на добавление товаров пользователь должен обратиться к администратору. Приложение позволяет создать новую учетную запись, чтобы пользователь мог легко и удобно получать доступ к различным функциям приложения и оставаться в курсе всех предложений и обновлений рисунок 2.5.

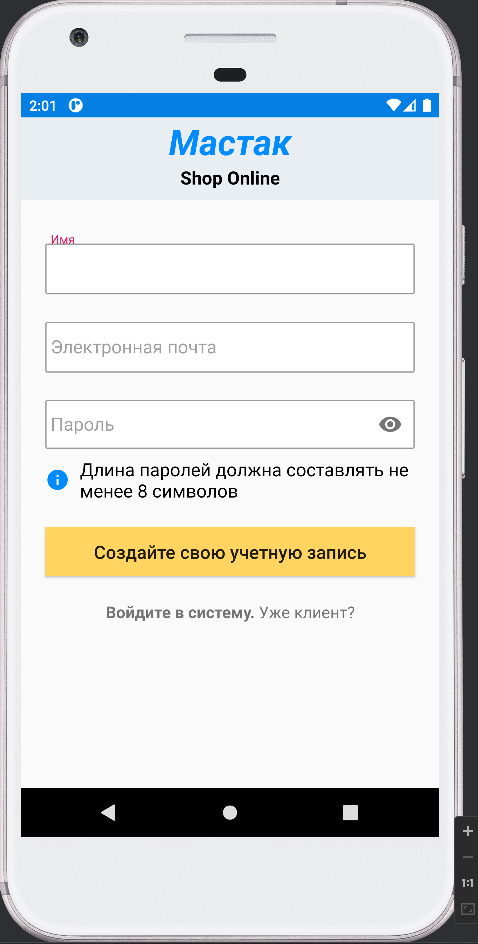


Рис 2.5 – Страница создания аккаунта

После успешной пройденной регистрации пользователь переходит на страницу со всеми товарами рисунок 2.6.

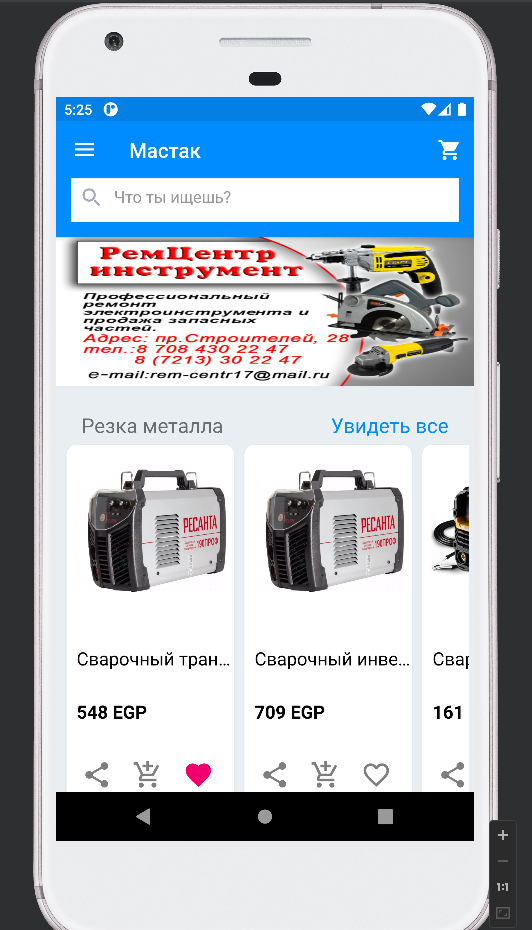


Рис 2.6 – Страница со всеми товарами

Пользователь, нажав на иконку, перейдет на станицу товара. На странице товара пользователь может добавить товар в корзину, купить, написать отзыв и посмотреть все отзывы. Нажав на иконку, тележка в верхнем правом углу, мы можем посмотреть содержимое корзины. Также нажав на иконку, три полоски в левом углу, откроется меню, тут пользователь может выбрать категорию товара, посмотреть свой заказ, настроить аккаунт, список желаний и ленту новостей.

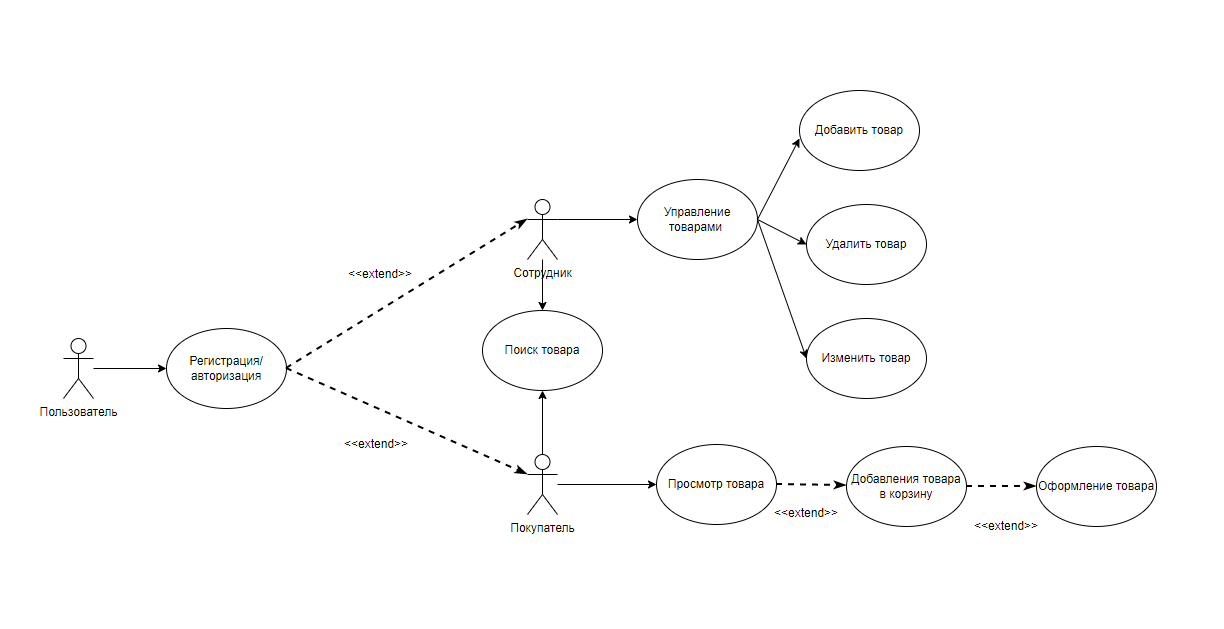


Рис. 2.7 – диаграмма использования мобильного приложения для пользователей

## 2.3 Проектирование сервера

Проектирование сервера — это процесс создания архитектуры и настройки серверной инфраструктуры, которая обеспечивает функционирование приложения и обработку запросов от пользователей.

Для создания механизма синхронизации данных между несколькими клиентами необходимо разработать серверное приложение, которое будет использоваться для взаимодействия компонентов распределенного приложения в сети. В качестве формата передачи данных между клиентом и сервером будет использоваться JSON. Для реализации сервера предлагается использовать архитектурный стиль REST, который позволяет использовать протокол HTTP для передачи данных без сохранения состояния между запросами. Для обеспечения взаимодействия клиентов и сервера предусмотрены HTTP методы, такие как GET, POST, PUT и DELETE, которые позволяют получать, отправлять и изменять данные на сервере. При проектировании сервера необходимо следить за корректностью кода и избегать избыточности и связности между составляющими частями, а также использовать подходящие паттерны проектирования и грамотно распределять логику сервера по разным уровням.

На основе приведенного выше контекста, рекомендуется разбить серверную логику на три уровня: API, Service и Repository. На уровне API находятся классы-контроллеры, которые описывают URL-адреса и их характеристики, преобразовывают данные в формат, подходящий для сохранения в БД и работы с ними как с бизнес-объектами. На уровне Service находятся классы, отвечающие за бизнес-логику. На уровне Repository находятся классы-репозитории, которые используются для операций с базой данных.

Важным аспектом проектирования серверной части приложения является вход пользователей в систему и аутентификация запросов. Рекомендуется использовать стандартные подходы, такие как аутентификация с помощью логина и пароля, используя собственную БД для хранения данных, либо сторонние сервисы, предоставляющие API для входа. При этом следует обратить внимание на типы приложений-клиентов, в которых будет использоваться система. В случае приложения под Android и веб-клиента было принято решение использовать JWT-токен для авторизации. Для поддержания статуса пользователя и выполнения запросов к API следует использовать JWT-токен, который поддерживает требуемый уровень безопасности.

## Вывод по главе «Проектирование информационной системы»

В данной главе было рассмотрено проектирование каждого отдельного компонента приложения.

По части мобильного приложения был определен основной функционал приложения, также были разработаны диаграмма вариантов использования приложения и описаны варианты использования программного средства.

Сбор и анализ требований — это важнейший этап проектирования, который помогает понять, что нужно создать, для кого и каким образом это будет использоваться. На этом этапе определяются функциональные и нефункциональные требования к продукту, его возможности и ограничения.

Была также спроектирована база данных, где были определены необходимые таблицы, охарактеризованы поля каждой из них, определены типы данных и ограничения целостности. Также были установлены связи между таблицами, заключающиеся в создании первичных и внешних ключей.

# Разработка информационной системы

Для разработки данного курсового проекта я использовала следующие программное обеспечение:

* Android Studio - это интегрированная среда разработки (IDE) для создания мобильных приложений под операционную систему Android. Эта IDE является официальным инструментом разработки от компании Google и основана на популярной среде разработки IntelliJ IDEA. Android Studio обладает широким набором инструментов для разработки мобильных приложений, включая редактор кода, отладчик, эмулятор устройства, инструменты для разработки пользовательского интерфейса и т.д. Эта IDE также обеспечивает интеграцию с другими инструментами разработки, такими как системы контроля версий Git и SVN, а также библиотеки и фреймворки, такие как Gradle. Android Studio также обладает большим сообществом разработчиков, которые предоставляют бесплатные плагины и инструменты, которые помогают ускорить процесс разработки. Благодаря всем этим инструментам и возможностям, Android Studio является одним из самых популярных инструментов для разработки мобильных приложений под Android.
* Visual Studio Code (VS Code) - это бесплатный и открытый исходный код редактор кода, который разработан Microsoft. Он поддерживает множество языков программирования и позволяет расширять функциональность с помощью плагинов и расширений. VS Code имеет множество функций, которые помогают упростить процесс разработки приложений. Некоторые из них включают: подсветка синтаксиса для многих языков программирования, автодополнение кода, интеграция с системой контроля версий Git, отладчик, который позволяет отслеживать выполнение кода и исправлять ошибки, интеграция с различными фреймворками и библиотеками.
* MySQL Workbench - это интегрированная среда разработки (IDE) для работы с базами данных MySQL. Она предоставляет пользователю графический интерфейс для создания и управления базами данных, а также для проектирования таблиц и запросов. MySQL Workbench имеет множество полезных функций, таких как возможность создавать, изменять и удалять таблицы, индексы, процедуры и триггеры; редактировать и выполнять запросы на языке SQL; создавать модели данных с помощью инструментов ER-моделирования; создавать резервные копии и восстанавливать данные; анализировать запросы на производительность и многое другое. MySQL Workbench также имеет удобный пользовательский интерфейс, который облегчает работу с базами данных. Он позволяет легко переключаться между различными базами данных, открывать несколько вкладок с различными запросами, визуально представлять структуру таблиц и многое другое. Кроме того, MySQL Workbench предоставляет мощные средства для работы с базами данных на удаленных серверах. Он поддерживает соединение с базами данных MySQL на удаленных серверах и предоставляет возможность удаленного управления базами данных. Для создания мобильного приложения я использовала язык программирования Java.

Java - это объектно-ориентированный язык программирования, который был разработан в 1995 году компанией Sun Microsystems. Он имеет богатую стандартную библиотеку, которая предоставляет множество готовых классов и методов для решения различных задач, таких как работа с сетью, файловой системой, графикой и т.д. Кроме того, для Java существует огромное количество сторонних библиотек и фреймворков, которые позволяют упростить и ускорить разработку приложений.

Одной из особенностей Java является ее строгая типизация и проверка на этапе компиляции, что позволяет выявлять многие ошибки еще до запуска программы. Кроме того, Java имеет сборщик мусора, который автоматически удаляет неиспользуемые объекты из памяти, что позволяет уменьшить количество ошибок, связанных с управлением памятью.

Java также имеет множество инструментов для отладки, профилирования и тестирования приложений, что делает процесс разработки более эффективным и быстрым.

Для создания веб-приложения я использовала язык разметки HTML и язык таблиц стилей CSS.

HTML и CSS являются основными технологиями для создания веб-страниц и веб-приложений. HTML (Hypertext Markup Language) - это язык разметки, который используется для определения структуры веб-страниц. CSS (Cascading Style Sheets) - это язык таблиц стилей, который используется для определения внешнего вида веб-страницы.

HTML используется для создания разметки веб-страниц, которая определяет структуру содержимого веб-страницы, такую как заголовки, параграфы, списки, ссылки и т.д. С помощью HTML вы можете создавать элементы, определять их свойства и располагать их на странице.

CSS используется для определения внешнего вида веб-страницы, такого как цвета, шрифты, размеры и расположение элементов на странице. Он позволяет создавать стили, которые могут быть использованы на нескольких страницах, а также использовать анимации и эффекты для создания интерактивных веб-страниц.

Совместное использование HTML и CSS позволяет создавать привлекательные и функциональные веб-страницы, которые могут отображаться на различных устройствах и браузерах.

Для создания сервера я использовала node.js c фреймворком Express.js. Node.js - это среда выполнения JavaScript, построенная на движке V8 Chrome. Она позволяет запускать JavaScript на стороне сервера и обеспечивает высокую производительность за счет асинхронной обработки запросов и событийного цикла.

Node.js широко используется для создания серверных приложений, веб-приложений и микросервисов. Он также позволяет работать с базами данных, создавать сетевые приложения и многое другое.

## Разработка базы данных

После того, как мы спроектировали нашу базу данных, ее необходимо подключить к серверу.

В листинге 1 видна реализация подключения базы данных к серверу в файле config.js.

const mysql = require('mysql2');

const connectionPool = mysql.createPool({

connectionLimit: 10,

host: '192.168.100.4',

user: 'root',

password: '1234567890',

database: 'bd',

})

module.exports = connectionPool;

Листинг 1 – Подключение базы данных к серверу

Первая строка создает объект mysql для подключения к базе данных. Затем объект connectionPool создается с использованием метода mysql.createPool(), который создает пул подключений к базе данных с ограничением соединений, указанным в свойстве connectionLimit.

Далее, в объекте connectionPool указываются параметры подключения к базе данных: host - адрес хоста, где запущена база данных, user - имя пользователя базы данных, password - пароль пользователя, database - имя базы данных.

Наконец, экспортируется объект connectionPool для использования в других модулях Node.js для выполнения запросов к базе данных.

## 3.2 Разработка мобильного приложения

Мобильное приложение было реализовано в среде разработки – Android Studio. Для написания приложения был использован язык программирования Java. Рассмотрим подробнее реализацию REST API для связи с сервером и получения данных на примере функции добавления блюда в избранное.

В папке net находится интерфейс Api - этот интерфейс предназначен для работы с REST API на стороне клиента в мобильном приложении, который использует библиотеку Retrofit2 в языке Java. В листинге 2 представлена реализация.

public interface Api {

@POST("favorites/add")

Call<ResponseBody> addFavorite(@Body Favorite favorite);

}

Листинг 2 – Реализация интерфейс Api

В папке net находится класс RetrofitClient в листинге 3 представлена реализация.

public class RetrofitClient {

private static final String BASE\_URL = LOCALHOST;

private static RetrofitClient mInstance;

private final Retrofit retrofit;

private RetrofitClient() {

retrofit = new Retrofit.Builder()

.baseUrl(BASE\_URL)

.addConverterFactory(GsonConverterFactory.create())

.build();

}

public static synchronized RetrofitClient getInstance() {

if (mInstance == null) {

mInstance = new RetrofitClient();

}

return mInstance;

}

public Api getApi() {

return retrofit.create(Api.class);}

}

Листинг 3 – Реализация класса RetrofitClient

Здесь создается клиент Retrofit для обращения к REST API сервера.

RetrofitClient - это класс, который содержит Retrofit клиент и метод для получения API, которое будет использоваться для вызова методов REST API.

Retrofit - это библиотека для Android, которая предоставляет удобный способ для работы с REST API на основе интерфейсов Java.

BASE\_URL - это базовый URL-адрес, который будет использоваться при обращении к REST API.

GsonConverterFactory - это фабрика конвертера, которая будет использоваться для преобразования JSON-данных в объекты Java.

GetApi() - это метод, который возвращает API, созданное с помощью Retrofit.

Также в классе определен статический метод getInstance(), который возвращает единственный экземпляр RetrofitClient. Этот метод использует паттерн Singleton для того, чтобы в приложении существовал только один экземпляр RetrofitClient.

В папке repository хранится класс AddFavoriteRepository реализация представлена в листинге 4.

public class AddFavoriteRepository {

private static final String TAG = AddFavoriteRepository.class.getSimpleName();

public LiveData<ResponseBody> addFavorite(Favorite favorite, RequestCallback callback) {

final MutableLiveData<ResponseBody> mutableLiveData = new MutableLiveData<>();

RetrofitClient.getInstance().getApi().addFavorite(favorite).enqueue(new Callback<ResponseBody>() {

@Override

public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {

Log.d(TAG, "onResponse" + response.code());

ResponseBody responseBody = response.body();

if(response.code() == 200){

callback.onCallBack();

}

if (response.body() != null) {

mutableLiveData.setValue(responseBody);

}}

@Override

public void onFailure(Call<ResponseBody> call, Throwable t) {

Log.d(TAG, "onFailure" + t.getMessage());

}

});

return mutableLiveData;}}

Листинг 4 – Реализация класса AddFavoriteRepository

В этом классе описана реализация добавления элемента в избранное. Он использует Retrofit для отправки POST запроса на сервер с данными, полученными из объекта Favorite. Класс возвращает LiveData<ResponseBody>, который можно использовать для отслеживания результата запроса.

Для уведомления об успешном добавлении элемента в избранное используется интерфейс RequestCallback. Если запрос успешен и сервер возвращает код 200, метод onCallBack интерфейса будет вызван.

В случае ошибки во время запроса будет вызван метод onFailure и будет выведено сообщение об ошибке.

В папке viewmodel хранится класс AddFavoriteViewModel , реализация которого представлена в листинге 5.

public class AddFavoriteViewModel extends ViewModel {

private final AddFavoriteRepository addFavoriteRepository;

public AddFavoriteViewModel() {

addFavoriteRepository = new AddFavoriteRepository();

}

public LiveData<ResponseBody> addFavorite(Favorite favorite, RequestCallback callback) {

return addFavoriteRepository.addFavorite(favorite,callback);

}

}

Листинг 5 – Реализация класса AddFavoriteViewModel

В этом классе определен конструктор, который создает экземпляр AddFavoriteRepository - класса, который обеспечивает доступ к данным для добавления избранных товаров.

В этом классе также определен метод addFavorite(), который принимает объект типа Favorite и RequestCallback в качестве параметров и возвращает объект LiveData<ResponseBody>.

Метод использует экземпляр AddFavoriteRepository, чтобы добавить избранный товар, вызывая метод addFavorite(), который делает запрос к API и возвращает объект LiveData<ResponseBody>.

При успешном добавлении метод callback.onCallBack() вызывается в onResponse().

Таким образом в результате нажатию на кнопку сердечка в мобильном приложении будет формировать и отправляться запрос на серверную часть.

## 3.3 Разработка веб-приложения

Веб-сервер разработан с использование языка программирование JavaScript и среды исполнения NodeJS. NodeJS — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Веб сервер разработан с использованием архитектуры REST. Каждая программная сущность определена в отдельный модуль. В общем случае каждая программная сущность описывается тремя компонентами: контроллер, сервис и роутер. Такое разделение позволяет независимо разрабатывать каждый компонент модуля по отдельности, а также позволяет относительно легко применить юнит тестирование к программному модулю.

Разработка сервера - это сложный процесс, требующий глубоких знаний в области программирования и сетевых технологий, а также внимательного планирования и тестирования.

Сервер состоит из следующих папок :

* api/routes: Эта папка содержит маршруты (routes) для API, которое предоставляет сервер. Здесь определены методы и логика, которые будут выполняться при получении запросов от клиентов.
* html: Эта папка содержит файлы HTML, которые будут отправляться клиентам при запросе страницы. Эти файлы содержат структуру и контент страниц, которые будут отображаться в веб-браузере.
* middleware: Эта папка содержит промежуточное ПО (middleware), которое будет выполняться перед выполнением методов API. В данном случае происходит проверка авторизации пользователя.
* node\_modules: Эта папка содержит все установленные зависимости Node.js. Здесь находятся все пакеты, которые используются на сервере.
* utils: Эта папка содержит вспомогательные функции и утилиты, которые могут быть использованы в различных частях сервера.
* config.js: Этот файл содержит конфигурационные параметры для сервера, такие как порт, на котором он должен запускаться, или параметры базы данных.
* nodemon.json: Этот файл содержит настройки для Nodemon, инструмента, который автоматически перезапускает сервер при изменении кода.
* package-lock.json: Этот файл содержит информацию об установленных версиях пакетов и их зависимостях, необходимую для корректной работы проекта.
* package.json: Этот файл содержит информацию о проекте, включая его имя, автора, зависимости и скрипты для запуска и сборки.
* server.js: Этот файл является главным файлом сервера, который будет запускаться. Он подключает и настраивает все необходимые компоненты, включая API, middleware, базу данных и т.д.

Продолжим рассмотрение добавления блюда в избранное только теперь со стороны сервера.

В листинге 6 представлен маршрут, который хранится в файле server.js. Для каждой из сущностей используется соответствующий роутер, который настраивается на определенный путь с помощью метода app.use().

const favoriteRouter = require('./api/routes/favorites')

app.use('/favorites',favoriteRouter)

Листинг 6 – Маршруты

В папке api/routes расположен файл favorites, который хранит в себе маршрут реализующий метод post.

Пример кода кода представлен в листинге 7 .

Когда клиент отправляет запрос на маршрут «favorites/add», сервер обрабатывает запрос и извлекает данные, отправленные в теле запроса (userId и productId). Затем он выполняет запрос на вставку в базу данных, используя параметризованный запрос для защиты от SQL-инъекций. Если вставка проходит успешно, сервер отправляет ответ с кодом состояния 200 и сообщением "Добавлено в избранное".

router.post("/add", (request, response) => {

const userId = request.body.userId

const productId = request.body.productId

const query = "INSERT INTO favorite(user\_Id, product\_Id) VALUES(?, ?)"

const args = [userId, productId]

database.query(query, args, (error, result) => {

if (error) {

if (error.code === 'ER\_DUP\_ENTRY') {

response.status(500).send("Дублирующаяся запись")

} else {

throw error;

}

} else {

response.status(200).send("Добавлено в избранное")

}

});

});

Листинг 7 – Добавление в избранное

Если происходит ошибка, сервер отправляет ответ с кодом состояния 500 и соответствующим сообщением об ошибке (например, "Дублирующаяся запись", если попытка добавить уже существующую запись).

## Вывод по главе «Разработка информационной системы»

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта.

Вторым этапом была разработка структуры базы данных на стороне сервера и клиента.

Третьим этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрены некоторые классы и их методы, в частности.

Четвертый этап – это разработка веб-приложения, где была представлена графическая структура веб-сервера и веб-клиента, рассмотрены некоторые классы и их методы.

# 4. Обеспечение безопасности информационной системы

Обеспечение безопасности информационной системы - одна из важнейших задач в разработке программных приложений. Пользовательская информация, такая как логины, пароли и данные карт, должна храниться в безопасном месте, чтобы не попасть в руки злоумышленников.

Для обеспечения безопасности данных приложения могут использовать несколько методов. Один из них - это хэширование паролей и карт пользователей в базе данных. Хэширование - это процесс преобразования данных в непредсказуемую строку фиксированной длины. При регистрации пользователей пароли и карты хэшируются и сохраняются в базе данных в зашифрованном виде. Когда пользователь пытается авторизоваться, введенный им пароль или номер карты хэшируются и сравниваются с хэшами, хранящимися в базе данных. Если хэши совпадают, то пользователь проходит авторизацию.

В node.js существует метод encrypt() - это функция, используемая для шифрования данных. Она может использоваться для шифрования конфиденциальной информации, такой как пароли, кредитные карты и другие чувствительные данные, перед их сохранением в базу данных.

Процесс шифрования с помощью encrypt() включает в себя преобразование исходных данных в байтовую строку и применение ключа шифрования для создания зашифрованных данных. Шифрованные данные могут быть сохранены в базе данных, а затем использованы для проверки подлинности пользователей при последующих запросах.

Одним из распространенных протоколов, использующих encrypt(), является протокол HTTPS. Этот протокол используется для обеспечения безопасной передачи данных через Интернет, используя шифрование данных во время передачи.

Еще один метод для обеспечения безопасности - это использование токенов для авторизации. Токен - это уникальный идентификатор, который используется для аутентификации и авторизации пользователей. При успешной авторизации сервер выдает токен, который клиент должен отправлять с каждым запросом, чтобы подтвердить свою личность. Токены могут быть временными или постоянными, а также могут иметь ограничения по времени жизни.

JWT (JSON Web Token) - это открытый стандарт (RFC 7519) для создания токенов авторизации, которые могут передаваться между клиентом и сервером в виде JSON-объектов. JWT может использоваться для аутентификации и авторизации пользователя в веб-приложениях.

JWT токен состоит из трех частей: заголовка (header), полезной нагрузки (payload) и подписи (signature).

Заголовок содержит информацию о том, как должен быть обработан токен. Он содержит тип токена и используемый алгоритм шифрования.

Полезная нагрузка содержит информацию, которая передается в токене. Это может быть информация об аутентификации пользователя, его роли в системе и т.д.

Подпись - это хеш-значение, созданное на основе заголовка, полезной нагрузки и секретного ключа. Он используется для проверки подлинности токена и защиты от подделки.

JWT токен может быть передан клиенту после успешной аутентификации пользователя. Клиент может сохранить токен в локальном хранилище (например, в localStorage или sessionStorage) и использовать его при каждом запросе к серверу. Сервер должен проверять подлинность токена при каждом запросе и возвращать ошибку, если токен недействителен.

Использование JWT токена вместо традиционных механизмов авторизации (например, cookie) имеет ряд преимуществ, таких как:

* Безопасность: JWT токен защищен от подделки и шифрованиям
* Масштабируемость: JWT токен может быть использован в микросервисах и распределенных приложениях
* Переносимость: JWT токен может быть передан между различными системами и языками программирования.

Однако, при использовании JWT токена необходимо учитывать следующие моменты:

* Надежность секретного ключа: если секретный ключ будет скомпрометирован, то все выданные токены также будут скомпрометированы
* Управление жизненным циклом токена: необходимо установить корректные значения срока действия токена и механизмы его обновления или отзыва
* Отсутствие контроля над токеном: клиент может сохранить токен в локальном хранилище и использовать его несанкционированно.

Если говорить о шифровании карт пользователей, то кроме хэширования, можно использовать алгоритмы симметричного и асимметричного шифрования. При симметричном шифровании используется один и тот же ключ для шифрования и дешифрования данных. При асимметричном шифровании используется пара ключей: публичный и приватный. Публичный ключ может использоваться для шифрования данных, а приватный - для их дешифрования.

# 4.1 Вывод по главе «Обеспечение безопасности информационной системы»

Есть множество различных способов обеспечение безопасности информационной системы. В своем курсовом проекте я использовала шифрование пароля и карты пользователя на сервере при помощи метода encrypt(). А также использовала JWT токен при авторизации пользователя.

# Заключение

Целью моего курсового проекта было предоставить пользователям возможность легко и удобно покупать товары.

Были решены следующие поставленные задачи:

* обзор аналогов;

В результате было рассмотрено 2 приложения и были выявлены их положительные и отрицательные стороны. Эта информация была использована для улучшения функционала приложения и пользовательского интерфейса.

* проектирование приложения;

Был спроектирован каждый отдельный компонент приложения. В мобильном и веб-приложении был определен основной функционал, также была разработана диаграмма вариантов использования. При проектировании базы данных, были определены необходимые таблицы и разработана схема базы данных. В серверном приложении был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передаваемых данных и HTTP методы для работы с серверной частью.

* реализация приложения;

Данная задача была реализована с помощью следующих технических средств: MySQL, сред разработки Android Studio и Visual Studio Code, языков программирования JavaScript, Java, библиотеки Retrofit 2.

Были реализованы базы данных на стороне сервера и клиента.

При решении данной задачи была достигнута поставленная цель, и был создан проект Мастак, который включает в себя android-клиент, веб-клиент и веб-сервер.

# Список использованных источников

1. REST-API [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/483202/>. – Дата доступа: 10.05.2023.
2. StudFiles [Электронный ресурс]. – Режим доступа: https://studfiles.net/preview/page:2/. – Дата доступа: 10.05.2023.
3. Трехуровневая архитектура — Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Трехуровневая\_архитектура. – Дата доступа: 16.05.2021.
4. Документация по Android [Электронный ресурс]. – Электронные данные. – Режим доступа: https://developers.google.com/android/ – Дата доступа: 10.05.2023;
5. Мартин, Р. Чистая архитектура и искусство разработки программного обеспечения / Р. Мартин. – Санкт-Петербург: Питер, 2019. – 352 с
6. Metanit Java [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/android/>. – Дата доступа: 16.05.2021
7. Голощапов А. Google Android: программирование для мобильных устройств / Голощапов А. — СПб.: БХВ-Петербург, 2010. —

# Приложение А

create database bd;

use bd;

CREATE TABLE `cart` (

`user\_id` int(11) NOT NULL,

`product\_id` int(11) NOT NULL,

FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `favorite` (

`favorite\_id` int(11) NOT NULL,

`user\_id` int(11) DEFAULT NULL,

`product\_id` int(11) DEFAULT NULL,

FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `history` (

`history\_id` int(11) NOT NULL,

`user\_id` int(11) DEFAULT NULL,

`product\_id` int(11) DEFAULT NULL,

FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `ordering` (

`ordering\_id` int(11) NOT NULL,

`order\_number` int(11) DEFAULT NULL,

`order\_date` date DEFAULT NULL,

`status` text,

`name\_on\_card` text,

`card\_number` text,

`expiration\_date` date DEFAULT NULL,

`user\_id` int(11) DEFAULT NULL,

`product\_id` int(11) DEFAULT NULL,

FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `poster` (

`poster\_id` int(11) NOT NULL,

`image` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `poster` (`poster\_id`, `image`) VALUES

(4, 'storage\_poster\\poster.jpg'),

(5, 'storage\_poster\\poster1.jpg'),

(6, 'storage\_poster\\poster2.jpg');

CREATE TABLE `product` (

`id` int(11) NOT NULL,

`product\_name` text,

`price` double(11,2) NOT NULL,

`quantity` int(11) NOT NULL,

`supplier` text NOT NULL,

`image` text,

`category` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `review` (

`review\_id` int(11) NOT NULL,

`user\_id` int(11) DEFAULT NULL ,

`product\_id` int(11) DEFAULT NULL,

`feedback` varchar(150) DEFAULT NULL,

`rate` float(11,1) DEFAULT NULL,

`review\_date` date DEFAULT NULL,

FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `shipping` (

`shipping\_id` int(11) NOT NULL,

`address` text,

`city` text,

`country` text,

`zip` text,

`phone` text,

`user\_id` int(11) DEFAULT NULL,

`product\_id` int(11) DEFAULT NULL,

FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `user` (

`id` int(11) NOT NULL,

`name` text NOT NULL,

`email` text NOT NULL,

`password` text NOT NULL,

`gender` text NOT NULL,

`age` int(11) NOT NULL,

`image` text NOT NULL,

`isAdmin` tinyint(1) NOT NULL DEFAULT '0'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `cart`

ADD PRIMARY KEY (`user\_id`,`product\_id`),

ADD UNIQUE KEY `cart\_constraint` (`user\_id`,`product\_id`),

ADD KEY `product\_id` (`product\_id`);

ALTER TABLE `favorite`

ADD PRIMARY KEY (`favorite\_id`),

ADD UNIQUE KEY `fav\_constraint` (`user\_id`,`product\_id`),

ADD KEY `product\_id` (`product\_id`);

ALTER TABLE `history`

ADD PRIMARY KEY (`history\_id`),

ADD UNIQUE KEY `history\_constraint` (`user\_id`,`product\_id`),

ADD KEY `product\_id` (`product\_id`);

ALTER TABLE `ordering`

ADD PRIMARY KEY (`ordering\_id`),

ADD KEY `user\_id` (`user\_id`),

ADD KEY `product\_id` (`product\_id`);

ALTER TABLE `poster`

ADD PRIMARY KEY (`poster\_id`);

ALTER TABLE `product`

ADD PRIMARY KEY (`id`);

ALTER TABLE `review`

ADD PRIMARY KEY (`review\_id`),

ADD UNIQUE KEY `review\_constraint` (`user\_id`,`product\_id`),

ADD KEY `product\_id` (`product\_id`);

ALTER TABLE `shipping`

ADD PRIMARY KEY (`shipping\_id`),

ADD KEY `user\_id` (`user\_id`),

ADD KEY `product\_id` (`product\_id`);

ALTER TABLE `user`

ADD PRIMARY KEY (`id`);

ALTER TABLE `favorite`

MODIFY `favorite\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=112;

ALTER TABLE `history`

MODIFY `history\_id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `ordering`

MODIFY `ordering\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=24;

ALTER TABLE `poster`

MODIFY `poster\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=7;

ALTER TABLE `product`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=61;

ALTER TABLE `review`

MODIFY `review\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=50;

ALTER TABLE `shipping`

MODIFY `shipping\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=23;

ALTER TABLE `user`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=26;

-- ALTER TABLE `cart`

-- ADD CONSTRAINT `cart\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

-- ADD CONSTRAINT `cart\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

-- ALTER TABLE `favorite`

-- ADD CONSTRAINT `favorite\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

-- ADD CONSTRAINT `favorite\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

-- ALTER TABLE `history`

-- ADD CONSTRAINT `history\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

-- ADD CONSTRAINT `history\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

-- ALTER TABLE `ordering`

-- ADD CONSTRAINT `ordering\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

-- ADD CONSTRAINT `ordering\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

-- ALTER TABLE `review`

-- ADD CONSTRAINT `review\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

-- ADD CONSTRAINT `review\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

-- ALTER TABLE `shipping`

-- ADD CONSTRAINT `shipping\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

-- ADD CONSTRAINT `shipping\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

INSERT INTO `product` (`id`, `product\_name`, `price`, `quantity`, `supplier`, `image`, `category`) VALUES

(19, 'Уровень строительный Total TMT215086M ', 91.92, 300, 'shop.by', 'storage\_product\\measuring4.jpg', 'measuring'),

(20, 'Уровень строительный Empire EM81.12M', 40.78, 300, 'shop.by', 'storage\_product\\measuring5.jpg', 'measuring'),

(21, 'Уровень строительный ADA Instruments Pro Digit RUMB A00481 ', 82.30, 300, 'shop.by', 'storage\_product\\measuring6.jpg', 'measuring'),

(22, 'Рулетка 3м 16 мм обрезинен.корп. блок GIANT 000050821712 ', 3.60, 700, 'shop.by ', 'storage\_product\\measuring7.jpg', 'measuring'),

(23, 'Сварочный трансформатор Telwin QUALITY 220 AC/DC 230-400V', 548.20, 700, 'shop.by', 'storage\_product\\metalCutting1.jpg', 'metalCutting'),

(24, 'Сварочный инвертор РЕСАНТА САИ-190ПРОФ', 709.00, 700, 'shop.by', 'storage\_product\\metalCutting1.jpg', 'metalCutting'),

(25, 'Сварочный инвертор Deko DKWM250A', 160.90, 676, 'shop.by', 'storage\_product\\metalCutting2.jpg', 'metalCutting'),

(26, 'Сварочный инвертор Spec MIG/MMA-250Pro', 805.00, 676, 'shop.by', 'storage\_product\\metalCutting3.jpg', 'metalCutting'),

(27, 'Сварочный аппарат Ставр САИ-180БТЛ', 238.40, 943, 'shop.by', 'storage\_product\\metalCutting4.jpg', 'metalCutting'),

(28, 'Сварочный инвертор Mikkele COMBIMIG-201G', 450.75, 340, 'shop.by', 'storage\_product\\metalCutting5.jpg', 'metalCutting'),

(29, 'Инверторный сварочный полуавтомат Aurora PRO SPEEDWAY 175', 1590.00, 600, 'shop.by', 'storage\_product\\metalCutting6.jpg', 'metalCutting'),

(30, 'Сварочный инвертор Deko DKWM250A', 160.00, 845, 'shop.by ', 'storage\_product\\metalCutting7.jpg', 'metalCutting'),

(31, 'Сварочный инвертор Solaris MULTIMIG-227', 1098.00, 755, 'shop.by', 'storage\_product\\metalCutting8.jpg', 'metalCutting'),

(32, 'Подающий механизм для Aurora SKYWAY 350 DUAL PULSE', 3237.00, 70, 'shop.by', 'storage\_product\\metalCutting9.jpg', 'woodworking'),

(33, 'Бензопила Champion 241-16', 299.00, 245, 'shop.by', 'storage\_product\\woodworking.jpg', 'woodworking'),

(34, 'Бензопила Shtenli 550', 490.00, 450, 'shop.by', 'storage\_product\\woodworking1.jpg', 'woodworking'),

(35, 'Бензопила Champion 251-18', 320.00, 300, 'shop.by', 'storage\_product\\woodworking2.jpg', 'woodworking'),

(36, 'Бензопила ECO CSP-235', 199.89, 250, 'shop.by', 'storage\_product\\woodworking3.jpg', 'woodworking'),

(37, 'Бензопила Patriot PT 2512', 249.00, 100, 'shop.by', 'storage\_product\\woodworking4.jpg', 'woodworking'),

(38, 'Бензопила Oleo-Mac GSH 51', 629.00, 150, 'shop.by', 'storage\_product\\woodworking5.jpg', 'woodworking'),

(39, 'Бензопила DGM GS-232', 173.51, 150, 'shop.by', 'storage\_product\\woodworking6.jpg', 'woodworking'),

(40, 'Бензопила Oasis GS-5218', 192.00, 660, 'shop.by', 'storage\_product\\woodworking7.jpg', 'woodworking'),

(41, 'Бензопила цепная Husqvarna 450 II', 1300.00, 660, 'shop.by', 'storage\_product\\woodworking8.jpg', 'woodworking'),

(42, 'Бензопила EFCO MTH 560', 658.50, 30, 'shop.by', 'storage\_product\\woodworking9.jpg', 'household'),

(43, 'Газонокосилка бензиновая Sturm PL5114SV', 767.70, 45, 'shop.by', 'storage\_product\\household.jpg', 'household'),

(44, 'Газонокосилка бензиновая Hyundai L5362S', 1397.00, 12, 'shop.by', 'storage\_product\\household1.jpg', 'household'),

(45, 'Газонокосилка электрическая Bosch ARM 37', 339.15, 300, 'shop.by', 'storage\_product\\household2.jpg', 'household'),

(46, 'Газонокосилка бензиновая Champion LM5347', 1163.10, 23, 'shop.by', 'storage\_product\\household3.jpg', 'household'),

(47, 'Газонокосилка электрическая Makita ELM3720', 499.00, 50, 'shop.by', 'storage\_product\\household4.jpg', 'household'),

(48, 'Газонокосилка бензиновая ECO LG-435', 518.40, 39, 'shop.by', 'storage\_product\\household5.jpg', 'household'),

(49, 'Газонокосилка бензиновая Champion LMH5640', 1219.00, 57, ' shop.by', 'storage\_product\\household6.jpg', 'household'),

(50, 'Газонокосилка бензиновая Skiper GW461S', 690.00, 60, 'shop.by', 'storage\_product\\household7.jpg', 'household'),

(51, 'Газонокосилка электрическая Makita ELM 3711', 392.60, 34, 'shop.by', 'storage\_product\\household8.jpg', 'household'),

(52, 'Лазерная рулетка ADA COSMO 120 Video', 296.42, 300, 'shop.by', 'storage\_product\\measuring.jpg', 'measuring'),

(53, 'Лазерный дальномер-рулетка Wortex LR 4004', 119.50, 400, 'shop.by', 'storage\_product\\measuring1.jpg', 'measuring'),

(54, 'Лазерный дальномер-рулетка Smart 20', 101.30, 450, 'shop.by ', 'storage\_product\\measuring2.jpg', 'measuring'),

(55, 'Лазерный дальномер-рулетка Mileseey DT10', 160.00, 450, 'shop.by ', 'storage\_product\\measuring3.jpg', 'measuring');