Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Модели данных и системы управления базами данных»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент кафедры Информатики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. С. Плиска |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«База данных для торговой площадки»**

БГУИР КП 1-40 04 01 010 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 053505  Слуцкий Никита Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc153023984)

[1 Описание программного продукта 4](#_Toc153023985)

[2 Тестирование приложения 8](#_Toc153023986)

[2.1 Общие сведения 8](#_Toc153023987)

[2.2 Тестирование базы данных 8](#_Toc153023988)

[2.3 Тестирование некоторых некоторых функций 9](#_Toc153023989)

[2.4 Запуск тестов при загрузке на удалённый репозиторий 10](#_Toc153023990)

[3 Развёртывание приложения 11](#_Toc153023991)

[3.1 Диаграмма развёртывания 11](#_Toc153023992)

[3.2 Развёртывание приложения и другие примечания 11](#_Toc153023993)

[Заключение 13](#_Toc153023994)

[Приложение А (обязательное) Листинг кода 14](#_Toc153023995)

# Введение

В настоящее время электронная коммерция стала неотъемлемой частью современного бизнеса, и интернет-магазины становятся все более важным каналом продаж для предприятий различных масштабов. С ростом популярности онлайн-торговли важно иметь эффективную и хорошо спроектированную базу данных, которая обеспечивает хранение, управление и доступ к информации о товарах, клиентах, заказах и других аспектах торговли в электронной среде.

Данный курсовой проект направлен на создание и исследование базы данных упрощённого интернет-магазина, которая превращает концепцию онлайн-торговли в управляемую и интуитивно понятную систему. В основу проектирования базы данных положены принципы гибкости, расширяемости, связанные с хранением и обработкой информации о продуктах, клиентах, заказах, оплате, доставке и других аспектах.

В качестве контекста в будущем может быть использовано предполагаемое приложение, которое будет реализовано на основе созданной базы данных, обеспечивая удобство для покупателей и эффективное управление информацией для администраторов и владельцев магазина.

Целями данного курсового проекта ставятся:

– разработать базу данных для упрощённого интернет-магазина товаров с использованием выбранных технологий;

– использовать не менее 25 сущностей в разрабатываемой базе данных;

– нормализовать базу данных до третьей нормальной формы;

– создать набор используемых триггеров, процедур и индексов;

– создать набор основных запросов для работы с базой данных;

– оформить пояснительную записку к курсовому проекту в соответствии со стандартом предприятия университета.

# 1 Обзор существующих аналогов

## 1.1 Каталог «Онлайнер»

Onliner.by – это белорусский сайт, который включает в себя СМИ, маркетплейс товаров и услуг, а также форум.

Каталог – это раздел сайта, где представлены различные товары и услуги. Здесь можно найти информацию о ценах, скидках, а также воспользоваться услугой оплаты частями. Это удобный инструмент для выбора и покупки различных товаров с доставкой по Беларуси.

На рисунке 1.1 приведено изображение главной страницы каталога.

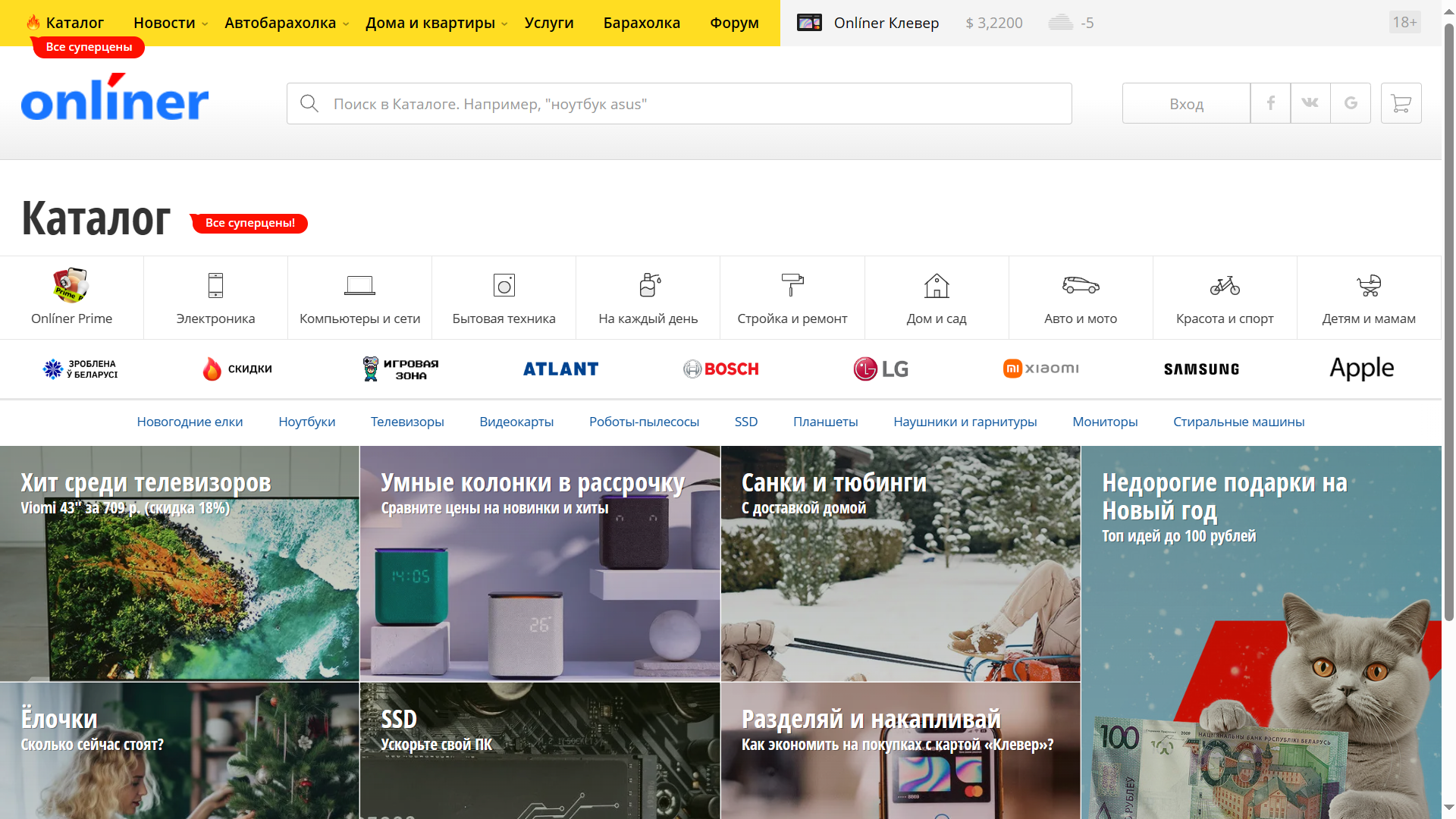


Рисунок 1.1 – Главная страница приложения каталога

Каталог предлагает пользователям искать товары в разных категориях, сравнивать товары по характеристикам и ценам от разных магазинов, а также производить оплату и заказ непосредственно через сайт. В одном из разделов сайта можно размещать объявления по продаже единиц товара, бывших в употреблении.

Магазинам данный каталог предоставляет возможность размещать свои предложения на площадке, а также покупать места на показ рекламы.

Таким образом это полноценное и большое решение по решению задач продаж очень широкого спектра товаров через интернет. Сайт является одним из самых посещаемых ресурсов в Беларуси.

## 1.2 Торговая площадка «Яндекс Маркет»

«Яндекс Маркет» – электронная торговая площадка, сервис для покупки товаров. Пользователь «Маркета» может просматривать и покупать товары из различных категорий, сравнивать их характеристики и цены, читать и оставлять отзывы и обзоры на товары, задавать вопросы другим посетителям сайта, магазинам и производителям. Сервис берет на себя хранение товаров, обработку и доставку заказов и общение с покупателями. Приложение «Яндекс.Маркет» доступно для iOS и Android.

На рисунке 1.2 приведено изображение главной страницы описываемого сайта.

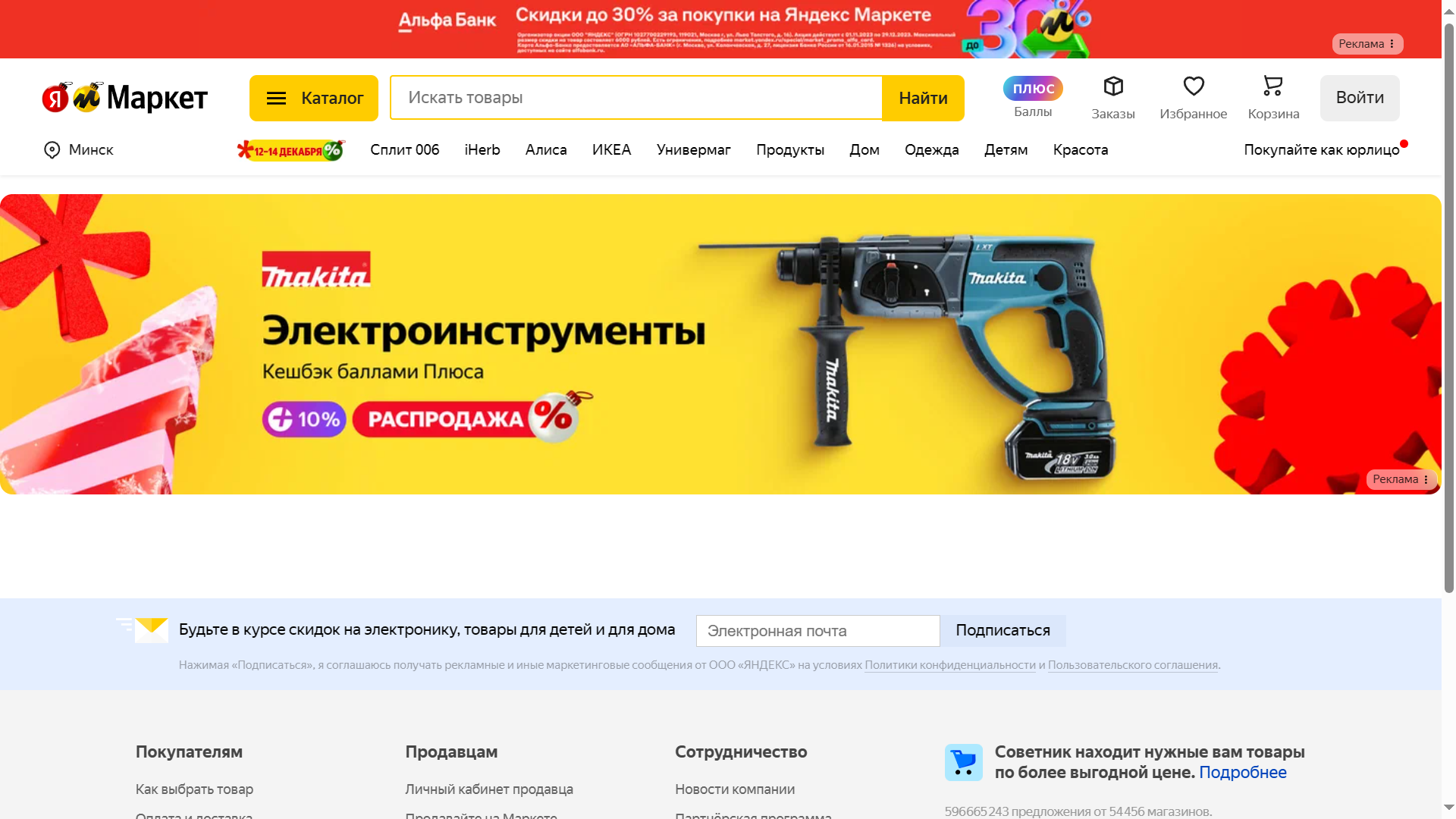


Рисунок 1.2 – Главная страница сервиса «Яндекс Маркет»

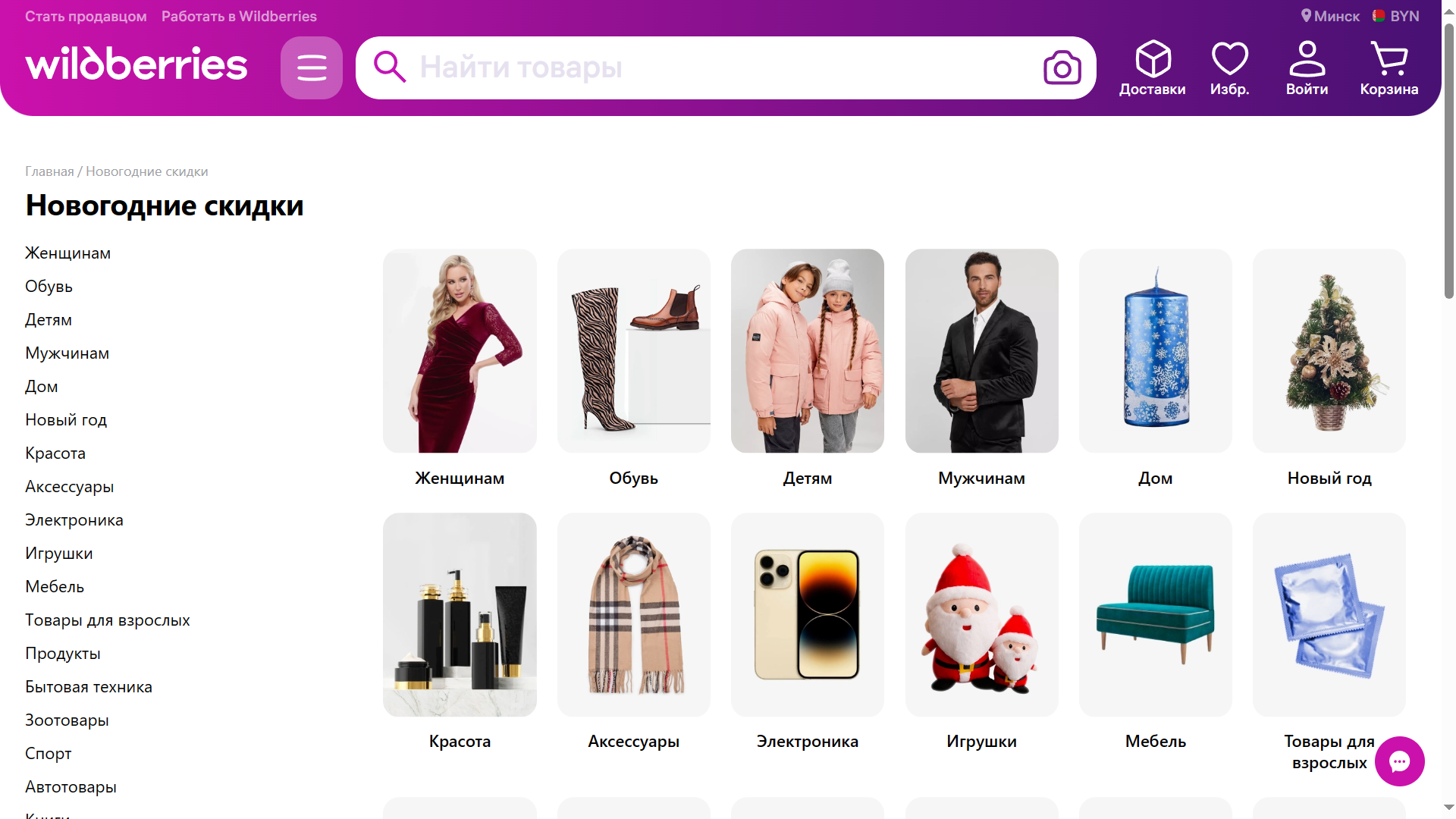
«Яндекс.Маркет» – единая торговая площадка, где продавцы размещают товары, а покупатели сравнивают и выбирают то, что им нужно – от электроники до одежды. Она доступна независимо от того, есть ли у магазина сайт.

## 1.3 Wildberries

Wildberries – российский международный интернет-магазин одежды, обуви, электроники, детских товаров, товаров для дома и других товаров.

География присутствия компании Беларусь, Россию и другие страны. Ежедневно на Wildberries оформляется 4 миллиона заказовОснован в 2003 году и управляется ООО «Вайлдберриз» со штаб-квартирой в Москве.

На рисунке 1.3 приведено изображение одной из страниц сайта.э



Бизнес-модель Wildberries характеризуют как онлайн-гипермаркет, магазин универсального формата или торговая площадка с товарами компаний-партнёров. Компания напрямую сотрудничает с производителями одежды и официальными поставщиками товаров. Они самостоятельно формируют ассортимент своих товаров в интернет-магазине и розничные цены, а Wildberries зарабатывает на комиссии по итогам продаж.

## 1.4 Формирование требований

Исходя из разбора существующих аналогов торговых площадок, к разрабатываемой базе данных упрощённой торговой площадке можно выдвинуть некоторые требования.

Площадка должна в разной степени поддерживать следующие возможности:

– регистрация магазинов;

– формирование цен на товар у разных магазинов;

– написание комментариев и отзывов;

– оформление заказов;

– разграничение прав доступа для пользователей;

– поддержка встраивания рекламы на сайт;

– наличие раздела с новостями и купонами.

При последующей возможной разработке приложения под разрабатываемую в рамках настоящего курсового проекта базу данных необходимо сделать упор также на:

– доступность интерфейса с разных платформ;

– валидацию данных пользователей при совершении платежей, оформлении заказа, регистрации.

– правильное разграничение доступа к разделам или страницам сайта;

– безопасность при работе с данными пользователей;

– удобные интерфейсы для взаимодействия системой не только для клиентов, а также других типов пользователей: администраторов, курьеров и представителей магазинов.

# 2 Формирование функциональных требований и выбор инструментов разработки

## 2.1 Формирование функциональных требований

Функциональное требование – это заявление о том, как должна вести себя система. Он определяет, что система должна делать, чтобы удовлетворить потребности или ожидания пользователя. Функциональные требования можно рассматривать как функции, которые обнаруживает пользователь. Они отличаются от нефункциональных требований, которые определяют, как система должна работать внутри. Ниже изложены основные функциональные требования к приложению, которое можно будет разработать на основе проектируемой в рамках настоящего курсового проекта базы данных.

1 Просмотр списка товаров. Пользователь должен иметь возможность просматривать товары в разных категориях и подкатегориях. Товары можно фильтровать по производителю и линейке, а также сортировать, например, по средней цене от магазинов.

2 Регистрация и вход в приложение. Пользователю должно быть предоставлено несколько вариантов входа в приложения для улучшения опыта использования.

3 Оформление заказа. У пользователя должна быть возможность составить заказ, возможно, состоящий из нескольких позиций и оформить его. Каждый магазин определяет свои возможные способы оплаты и доставки. Заказ может объединить в себе лишь товары из одного магазина. При этом если есть необходимость заказать товары из разных магазинов, необходимо будет сделать несколько заказов.

4 Просмотр информации. Пункт включает в себя просмотр информации разного типа:

– описание товара;

– информация о поставщиках и линейках;

– сведения о магазинах, их способах оплаты и доставки, пунктах самовывоза.

– сведения об актуальных купонах и акциях;

– новости портала;

– отзывы к товарам;

– форумы и комментарии;

– часто задаваемые вопросы.

5 Журналирование действий. Основные виды действий должны заноситься в таблицу журналирования.

6 Права пользователей с ролью администратора. Они должны включать редактирование списка товаров без привязки к магазинам и конкретным ценам, добавление новых рекламных объектов в реестре отображаемой рекламы, изменение списка купонов, а также регистрацию новых магазинов и их представителей.

7 Права представителей магазинов. Представители магазинов при пользовании приложением относятся к пользователям с отдельной соответствующей ролью. Они могут редактировать товары, выставленные их магазином на разрабатываемой торговой площадке.

8 Права пользователей с ролью курьеров. В будущем планируется расширение базы данных и приложения с возможностью отслеживания истории доставок, самовывозов, а также с введением системы рейтинга для курьеров и обычных клиентов.

## 2.2 Выбор инструментов разработки

Исходя из описанных выше функциональных требований системы и имеющегося опыта разработки, выбор сделан в пользу проектирования с использованием реляционных систем управления базой данных. В настоящее время одними из самых используемых реляционных систем управления базой данных являются PostgreSQL и Oracle Database. Также набирают популярность комплексы, включающие внутри себя базу данных и многие вспомогательные инструменты и сервисы, такие, как, например, Supabase. Упомянутые технологии описаны ниже.

**2.2.1** Система управления базами данных PostgreSQL. PostgreSQL – свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL. Имеет открытый исходный код и является альтернативой коммерческим базам данных.

Данная система управления базами данных позволяет гибко управлять базами данных (БД). С ее помощью можно создавать, модифицировать или удалять записи, отправлять транзакцию – набор из нескольких последовательных запросов на особом языке запросов SQL.

Система управления базами данных PostgreSQL нужна для:

– гибкого доступа к базам данных, их организации и хранения;

– управления записями в базах данных: создания, редактирования и удаления, обновления версий и так далее;

– просмотра нужной информации из базы по запросу, например для ее отправки на сайт или в интерфейс приложения;

– отправки транзакций, последовательных запросов, собранных в подобие скрипта;

– настройки и контроля доступа к той или иной информации, группировки пользователей по уровню прав;

– контроля версий и организации одновременного доступа к базе из разных источников так, чтобы предотвратить сбои;

– защиты информации от возможных утечек и потерь.

– контроля состояния базы в целом.

Иногда PostgreSQL называют бесплатным аналогом Oracle Database. Обе системы адаптированы под большие проекты и высокую нагрузку, но по разному хранят данные.

Среди преимуществ PostgreSQL разработчики выделяют следующие пункты:

1 Объектно-реляционная модель. Традиционно популярные системы управления базами данных – реляционные. Это значит, что данные, которые в них хранятся, представляются в виде записей, связанных друг с другом отношениями. Получаются связанные списки, которые могут иметь между собой те или иные отношения, – так и образуется таблица.

2 Поддержка множества типов данных. Еще одна особенность PostgreSQL – поддержка большого количества типов записи информации. Это не только стандартные целочисленные значения, числа с плавающей точкой, строки и логические значения, но и денежный, геометрический, перечисляемый, бинарный и другие типы. PostgreSQL поддерживает битовые строки и сетевые адреса, массивы данных, в том числе многомерные, композитные типы и другие сложные структуры. В ней есть поддержка XML, JSON и NoSQL-баз.

3 Работа с большими объемами. В большинстве СУБД, рассчитанных на средние и небольшие проекты, есть ограничения по объему базы и количеству записей в ней. В PostgreSQL ограничений нет.

4 Поддержка сложных запросов. PostgreSQL работает со сложными, составными запросами. Система справляется с задачами разбора и выполнения трудоемких операций, которые подразумевают и чтение, и запись, и валидацию одновременно. Она медленнее аналогов, если речь заходит только о чтении, но в других аспектах превосходит конкурентов.

5 Написание функций на нескольких языках. В PostgreSQL можно писать собственные функции – пользовательские блоки кода, которые выполняют те или иные действия. Эта возможность есть практически в любых СУБД, но PostgreSQL поддерживает больше языков, чем аналоги. Кроме стандартного SQL, в PostgreSQL можно писать на C и C++, Java, Python, PHP, Lua и Ruby. Он поддерживает V8 – один из движков JavaScript, поэтому JS тоже можно использовать.

6 Одновременная модификация базы. Важная особенность PostgreSQL –  возможность одновременного доступа к базе с нескольких устройств. В СУБД реализована клиент-серверная архитектура, когда база данных хранится на сервере, а доступ к ней осуществляется с клиентских компьютеров. Так, например, реализуются разнообразные сайты. Одна из возможных сложностей — ситуация, когда несколько человек одновременно модифицируют базу и нужно избежать конфликтов.

7 Высокая мощность и широкая функциональность. PostgreSQL – возможно, единственная бесплатная СУБД с открытым исходным кодом, которая рассчитана на работу с объемными и сложными проектами. Она мощная, производительная, способна эффективно работать с большими массивами данных.

**2.2.2** Oracle Database – это объектно-реляционная система управления базами данных от компании Oracle. Она используется для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации.

Данная система объединяет в себе две модели хранения информации: объектно-ориентированную и реляционную. Реляционная модель представляется как набор отношений между записями. Одни данные связаны с другими – так формируется база. Визуально ее можно представить как двумерную таблицу; математически – как модель, построенную на отношениях. Объектно-ориентированная модель воспринимает данные как объект. У объекта есть атрибуты, которые описывают его свойства, и методы – они нужны для взаимодействия с другими объектами. Каждый объект принадлежит к классу – это понятие можно представить как «схему» объекта. Объектно-ориентированный подход используется во многих языках программирования и упоминается во многих наших статьях. А в базах данных он нужен для работы с данными, у которых сложная структура.

Oracle Database работает и с объектно-ориентированной, и с реляционной моделью.

Система работает по принципу «клиент – сервер». Это означает, что ее основная часть размещается на сервере, там же, где и база данных. Человек работает с интерфейсом приложения-клиента. Клиентская часть управляет только пересылкой и получением информации от сервиса.

Достоинства такого подхода – в высоком уровне безопасности и легком доступе для клиентов. Клиент-серверная организация разгружает сеть и снимает вычислительную нагрузку с клиентских компьютеров. А вот сервер для такой системы управления базами данных должен быть мощным.

Информация в системе хранится в отдельных базах – экземплярах базы данных. Это не физические, а логические понятия, которые состоят из процессов и оперативной памяти. Все содержимое одного экземпляра имеет единую системную глобальную область – часть оперативной памяти, с которой работает.

Внутри экземпляров расположены логические пространства, которые называются табличными – tablespaces. Табличные пространства содержат компоненты данных – как файлы в папках.

Данная система управления базами данных состоит из одного или нескольких инстансов и программного обеспечения, которое ими управляет. Система поддерживает работу с независимыми базами в рамках одного инстанса. Она может работать и с мультиарендной архитектурой, где множественными клиентами управляет один экземпляр приложения. В Oracle поддерживаются кластеризация и секционирование – физическое разделение элементов баз данных без потери доступа.

**2.2.3** Supabase – это реляционная база данных на основе тех же технологий, что лежат в PostgreSQL – одной из самых популярных и надежных баз данных в мире.

Supabase – это бесплатный аналог Firebase, полифункциональная платформа, объединяющая в себе несколько важных программных решений и упрощающая их реализацию до предельно примитивного уровня, чтобы разработчики могли добавлять в свои приложения или сайты такие функции, как:

– авторизация;

– хранилище файлов;

– обновление контента на сайте в реальном времени.

Разработчики данного сервиса требуют оплату использования только при достижении количества запросов определенных значений. То есть на этапе разработки оплачивать БД не придется, все возможности Supabase можно будет опробовать самостоятельно, а не опираться на обзоры и демо.

Supabase идет в комплекте с собственным хранилищем файлов, которое можно подключать к базе данных. Например, можно добавить в Supabase файлы изображений, еще на этапе их загрузки в базу данных создавать специальные ссылки и закреплять их за статьями, комментариями, профилями в таблицах Supabase. Таким образом, можно привязывать файлы из хранилища к записям из базы данных, создавая бесшовную систему. Разработчики создали достаточный набор готовых команд для управления файлами.

Важное преимущество Supabase – встроенная функция авторизации, реализованная практически на надёжном уровне.

Supabase –универсальный продукт, включающий в себя много компонентов, необходимых для создания полноценных приложений.

**2.2.4** Обоснование выбора. Была выбрана реляционная модель баз данных. Реляционная модель предлагает следующие преимущества:

1 Структурированность данных и декларативный язык запросов: реляционные базы данных хранят данные в виде таблиц, что обеспечивает структурированность и позволяет устанавливать отношения между различными таблицами.

2 Целостность данных и связей: реляционные базы данных обеспечивают соблюдение целостности данных через использование ограничений целостности и связей между таблицами.

3 Гибкость в запросах и аналитике: Благодаря SQL, реляционные базы данных предоставляют богатые возможности для запросов и аналитики данных. Это включает в себя операции объединения, агрегации, фильтрации и многие другие, что позволяет эффективно извлекать и обрабатывать информацию.

4 Нормализация и избыточность: реляционные базы данных поддерживают нормализацию, что помогает избежать избыточности данных и минимизировать возможность ошибок в обновлении и модификации данных.

5 Масштабируемость: реляционные базы данных обладают хорошей масштабируемостью, позволяя увеличивать количество записей, пользователей и производительность по мере роста приложения.

6 Многозадачность и параллелизм: реляционные системы управления базами данных предоставляют возможность эффективно обрабатывать множество запросов параллельно, что обеспечивает производительность и отзывчивость системы.

7 Поддержка транзакций: реляционные базы данных обеспечивают ACID-транзакции для обеспечения целостности и надежности данных.

В ходе разработки приложения целесообразно использовать PostgreSQL, потому что мощности Oracle Database избыточны. Supabase представляет собой обёртку над PostgreSQL с дополнительными удобными возможностями для разработки, например, механизмы аутентификации и поддержания сессии. Поэтому выбор сделан в пользу Supabase.

# 3 Проектирование базы данных

## 2.2 Тестирование базы данных

Тесты запускаются на копии реальной базы данных. Это всё такой же сервер Supabase с запущенным экземпляром PostgreSQL внутри. Перед началом тестов конструкцией beforeAll создаётся клиент Supabase и регистрируются два тестовых пользователя. На каждом шаге проверяется статус операции.

При регистрации пользователь автоматически заносится во встроенную таблицу Auth.Users, однако где-то необходимо хранить и не только стандартные данные, например, имя, фамилию, номер паспорта. Оптимальным решением для создания новой записи в таблице   
Profiles является создание триггера, который после вставки записи во встроенную таблицу Users вытягивает метаданные из запроса и создаёт запись в Profiles.

Первый тест покрывает логику регистрации и работы созданного триггера на таблицу Profiles. Тест проверяет возможность авторизоваться в приложении а также, что триггер отработал – и кастомные данные пользователя также сохранились.

Следующий набор тестов проверяет корректность настройки политик доступа к таблице счетов в базе данных. Проверяется защита доступа неавторизированного пользователя от создания новой записи в этой таблице. Далее последовательно происходит вход в два созданных аккаунта и создание счёта от их имени. Причём также проверяется, что после создания этих счетов у их создателей доступ предоставляется только непосредственно к их собственным счетам.

Одной из важных частей приложения и базы данных является хранимая процедура совершения платежей. В некорректных ситуациях процедура выбрасывает исключение и прерывает транзакцию. Необходимо протестировать функцию как на некорректных, так и на корректных данных. Проверяется факт наличия ошибки при таких случаях, как:

– отрицательная сумма перевода;

– недостаточное количество средств на выбранном счёте отправителя.

После этого совершается запрос с корректными данными и производится проверка на уменьшение средств на счёте-отправителе и увеличение средств на счёте-получателе.

Необходимо проверить механизм входа в приложение. Поэтому несколькими наборами тестируется функция входа. Это могут быть комбинации верного адреса электронной почты, но неверного пароля, или наоборот.

После прохождения тестов необходимо отдельно созданным клиентом Supabase с правами администратора удалить созданные аккаунты клиентов. Это происходит в конструкции afterAll.

## 2.3 Тестирование некоторых некоторых функций

Ввиду наличия системы валидаторов в приложении целесообразным является протестировать эту систему. Тесты покрывают такие функции-валидаторы, как:

– валидатор номера паспорта;

– валидатор адреса электронной почты;

– валидатор номера телефона.

Также тестируется общая функция для валидации целого набора данных как на корректном наборе, так и на невалидном.

Функция, рассчитывающая ежемесячный платёж по кредиту, также покрыта базовым тестом с сайта, с которого была взята формула расчёта кредита.

## 2.4 Запуск тестов при загрузке на удалённый репозиторий

Одним из условий лабораторной работы является выкладывание исходного кода в удалённый репозиторий GitHub. GitHub предоставляет возможность автоматизировать некоторые процессы после каждого сохранения кода в репозитории. Можно разрабатывать сценарии, при которых приложение собирается, тестируется различными средствами и в случае успеха деплоится на сервер.

В случае с разрабатываемым решением с помощью GitHub Actions CI был автоматизирован запуск тестов в репозитории. Таким образом посетитель репозитория может наблюдать, прошла ли текущая выложенная версия приложения тесты или нет. На рисунке 2.1 приведен скриншот результата тестирования внутри GitHub Actions.

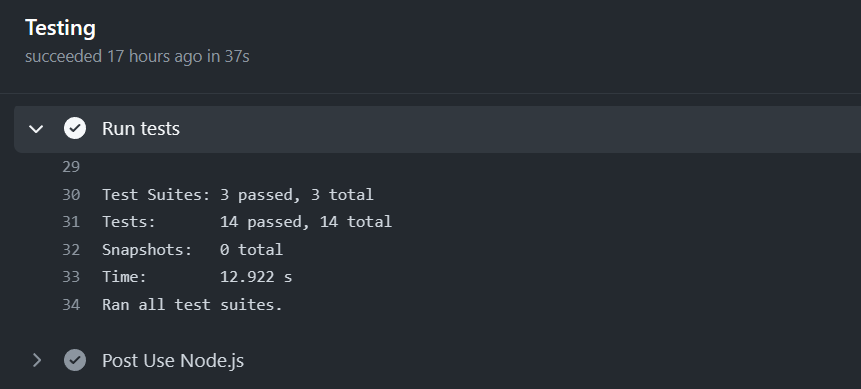


Рисунок 2.1 – Результат удалённого автоматического тестирования

Для написания такого сценария в корне локального репозитория в специальной папке создаётся файлик в формате YAML. В нём описывается используемое окружение и последовательность команд. Для подключения к тестовой базе данных внутри тестов используются секретные ключи. Они должны быть записаны в переменные окружения во время запуска тестов. Локально они хранятся в файлике, который игнорируется системой контроля версий GIT и не попадает на удалённый репозиторий. В репозитории созданы так называемые секретные переменные. Таким образом никакой из ключей доступа не доступен случайному посетителю репозитория.

# 3 Развёртывание приложения

## 3.1 Диаграмма развёртывания

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их. Включает в себя следующие элементы: узлы, артефакты, соединения, устройства.

На рисунке 3.1 приведена диаграмма развёртывания для разработанного приложения.

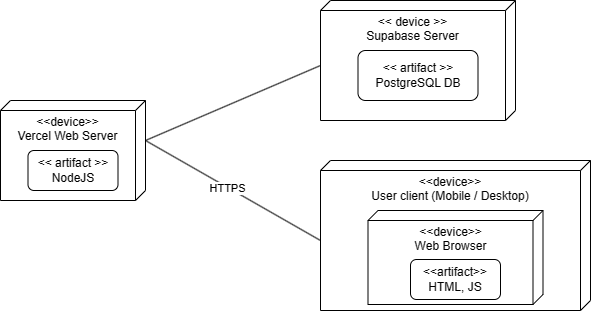


Рисунок 3.1 – Диаграмма развёртывания

Диаграммы развертывания помогают моделировать аппаратную топологию системы по сравнению с другими типами UML-диаграмм, которые в основном описывают логические компоненты системы.

## 3.2 Развёртывание приложения и другие примечания

Приложение разработано на технологии NextJS. Разрабатывающая этот фреймворк компания предлагает удобный способ сборки и автоматического развёртывания – с использованием Vercel CLI.

Vercel – это облачная платформа, специализирующаяся на развертывании статических сайтов и серверных функций. Она позволяет размещать сайты и приложения, которые мгновенно развертываются, автоматически масштабируются и не требуют никаких трудозатрат для обслуживания. Vercel предлагает бесплатный уровень, позволяющий начать работу без внесения каких-либо платежей и допускающий развертывание бессерверных функций для реализации бэкенда

Таким образом приложение может быть выложено в открытый доступ всего за несколько шагов:

– создание аккаунта Vercel;

– установка Vercel CLI;

– авторизация на локальной машине;

– запуск команд vercel и vercel --prod.

Можно настроить GitHub Actions CI на автоматический деплой приложения при успешном прохождении тестов, а можно деплоить вручную. Для этого в файле package.json созданы команды deploy:dev, deploy:prod и   
deploy.

Исходный код приложения можно исследовать в удалённом репозитории GitHub для лабораторных работ по предмету. В репозитории созданы Readme.md файлы, описывающие основные важные моменты, касающиеся проекта, а также инструкции по запуску приложения и тестов.

# Заключение

В результате выполнения был разработан полноценный программный продукт упрощённого аналога банковской системы, а также написаны тесты для ключевой логики приложения. Тесты автоматически запускаются при сохранении кода в удалённом репозитории – и таким образом всегда можно наблюдать актуальное состояние статуса работоспособности ключевых частей программного продукта. В качестве дополнительных видов тестирования для веб-приложения также можно реализовывать:

– тестирование React-компонентов с помощью Testing Library;

– тестирование страниц и пользовательских сценариев с помощью   
Cypress;

– нагрузочное тестирование;

– тестирование безопасности.

Приложение развёрнуто в открытом доступе. А также разработана диаграмма развёртывания и составлен отчёт, оформленный в соответствии с общими требования стандарта предприятия БГУИР.

В приложении А приведена вырезка кода из тестирования базы данных, которая совершает действия перед началом тестирования и после окончания тестирования.

Цели лабораторной работы можно считать достигнутыми.

# Приложение А (обязательное) Листинг кода

beforeAll(async () => {

SupabaseAdmin = createClient(process.env.DB\_URL as string, process.env.DB\_KEY\_ADMIN as string);

Supabase = createClient(process.env.DB\_URL as string, process.env.DB\_KEY as string);

const signUpResponse1 = await Supabase.auth.signUp(user1);

const signUpResponse2 = await Supabase.auth.signUp(user2);

if (signUpResponse1.data.user) {

user1.id = signUpResponse1.data.user.id;

console.log('Signed up a new user: ', user1.id);

}

if (signUpResponse2.data.user) {

user2.id = signUpResponse2.data.user.id;

console.log('Signed up a new user: ', user2.id);

}

});

afterAll(async () => {

const deleteResult1 = await SupabaseAdmin.auth.admin.deleteUser(user1.id!);

expect(deleteResult1.error).toBe(null);

const deleteResult2 = await SupabaseAdmin.auth.admin.deleteUser(user2.id!);

expect(deleteResult2.error).toBe(null);

});