Олимпиада школьников «Шаг в будущее»

Научно-образовательное соревнование «Шаг в будущее, Москва»

Регистрационный номер

ИУ «Информатика и системы управления»

название факультета

ИУ 7 «Программное обеспечение ЭВМ и информационные технологии»

название кафедры

Клиент-серверное приложение для продвижения товаров и услуг на отечественном рынке

название работы

Автор:

Янбухтин Даниил Эльдарович

Школа 57, 11 И

Научный руководитель:

Филиппов Михаил Владимирович

Преподаватель кафедры ИУ 7

Оглавление

Введение....1
1. Аналитический раздел....2
1.1.1 Требуемые функции....2
1.1.2 Обзор существующих решений....2
1.2.1 Вывод....3
2. Конструкторский раздел....3
2.1 Общая схема....3
2.2 Протокол взаимодействия....4
2.3 Архитектура серверной части....4
2.3.1 Выбор системы для первой СУБД....5
2.3.2 Проектирование первой СУБД....5
2.3.2 Проектирование второй СУБД....8
2.3.3 Выбор языка программирования....9
2.3.4 Основные библиотеки....10
2.4 Клиентская часть....10
3. Функциональный пример....10
Заключение....12
Список использованных источников....13

Введение

За прошедшие два года с российского потребительского рынка ушли многие зарубежные компании. Этот процесс стимулировал необходимость в продвижении отечественных аналогов или новых продуктов локального производства. Однако, многие российские компании и бренды территориально ограничены исключительно рамками регионов производства и не доступны массовому российскому потребителю. Вместе с тем имеют сопоставимое или лучшее качество и более доступны по стоимости. Для решения данной диспропорции, расширения региона продаж и стимулирования потребительского спроса, предлагается создать **единую цифровую платформу** для продвижение товаров и услуг на отечественном потребительском рынке.

Цель работы — разработать серверное приложение с графическим интерфейсом, реализующее функции реестра и акселератора компаний и брендов в интересах пользователей и потребителей.

Задачи работы:

- изучить существующие аналоги
- определить требуемый функционал
- разработать схему базы данных
- разработать серверное приложение
- разработать графический интерфейс (веб-сайт).

1. Аналитический раздел

Прежде чем приступить к реализации проекта, необходимо проанализировать существующие решения-аналоги, чтобы определить требуемый функционал и его теоретическое расширение, а также возможные проблемы.

1.1.1 Требуемые функции

Объективно, решения подобного рода могут иметь бесконечное число возможностей для расширения, в зависимости от конкретных задач пользователей, которые не нашли своего отражения в предлагаемом проекте. В этой связи, учитывая мое субъективное мнение, предлагается рассмотреть только основные возможности как ядро для дальнейшего масштабирования:

1. Возможность добавления своего бренда/компании/товара/…
2. Описание основных преимуществ , пользовательских свойств потребительских качеств предлагаемого продукта, количественные и качественные характеристики товара , предоставление дополнительной информации (число продаж, удовлетворенность потребителей и т.д.)
3. Общий подход, без специализации в конкретный пользовательский сегмент
4. Анализ полноты данных

1.1.2 Обзор существующих решений

Следует отметить, что довольно много решений, которые я нашел в Интернете были недоступны либо из-за блока Cloudflare-ом, либо из-за ошибок на 404, 503 и пр. В связи с этим, никакой информации о них нет.

	Возможность добавления	Описание преимуществ	Общий подход	Анализ полноты данных
Росреестр	Через официальную процедуру с заявкой	Нет	Да (для товаров, отвечающих критериям)	Да
https://madeinrussia.ru	Через официальную заявку	Да	Да	Нет
https://gisp.gov.ru/pp719v2/pub/prod/	Через выписку из реестра и официальную заявку	Да	Нет (только промышленность)	Да

1.2.1 Вывод

Все указанные в пункте 1.1.1 функции полезны и необходимы. Все они требуют реализации с возможностью расширения функционала.

Вместе с тем, указанные программные продукты имеют существенный недостаток, а именно: добавление новых брендов сопряжено со сложными

бюрократическими процедурами. Поэтому возникает необходимость в создании программного комплекса, свободного от этого недостатка.

2. Конструкторский раздел

В данном разделе мы разберем схему и архитектуру системы, язык программирования, базу данных и некоторые детали реализации.

2.1 Общая схема

Общая схема программного комплекса представлена на рис.1

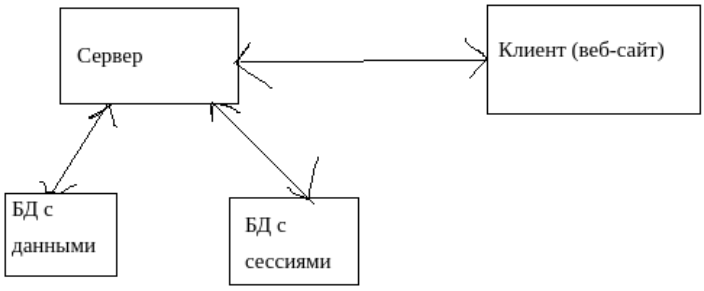


Рис.1 Схема комплекса

Вся информация о брендах, товарах и т.д. хранится на сервере. Кроме того, для пользователя предусмотрена возможность регистрации на сайте, после чего пользователю «выдается» регистрационная сессия, которая существует в некотором временном отрезке. Для оптимизации нагрузки на сервер, все эти сессии хранятся в отдельной базе данных.

2.2 Протокол взаимодействия

Для решения поставленной задачи был выбран протокол сетевого взаимодействия HTTP. Данный протокол является стандартом для индустрии, защищен и удобен в использовании.

2.3 Архитектура серверной части

Для общения с клиентской частью используется REST API. Это популярный архитектурный подход для создания API серверных приложений.

При проектировании серверного API по принципам REST ключевым фактором является то, какие ресурсы разработчик хочет сделать доступными клиентской части. Это значительно упрощает добавление нового функционала или предоставление дополнительной информации клиенту.

Все данные маршаллируются в JSON — универсальный формат хранения и передачи данных в структурированном виде «ключ-значение».
Как уже было сказано выше, сервер общается с двумя базами данных:

1. СУБД — хранит основную информацию о брендах, товарах, пользователях и тд.
2. СУБД — хранятся активные пользовательские сессии, которые удаляются по мере их истечения.

2.3.1 Выбор системы для первой СУБД

Выбор происходил между SQL и NoSQL СУБД.

Было принято решение использовать SQL базу данных, за возможность устанавливать гибкие ограничения на данные, такие как внешние ключи, проверочные ограничения и уникальные индексы. База данных позволяет относительно легко обрабатывать сложные объекты. Для дальнейшей обработки массива данных, где многие параметры взаимосвязаны, а у большинства полей существуют недопустимые значения, требующие корректировки, эти преимущества очень важны. В работе над проектом также очень важна ACID-совместимость, которую SQL обеспечивает лучше, чем NoSQL.

Конкретный диалект, который был выбран для разработки — PostgreSQL, за наличие конструкции UPSERT, которая позволяет легко обновлять сложный объект, а также за предоставление «из коробки» решения некоторых проблем с изоляцией транзакций.

В то же время, анализировались другие сценарии, при которых преимущества NoSQL базы данных могут дать больший выигрыш при разработке проекта. В таких сценариях обычно присутствует большое количеством неструктурированных данных, чего не ожидается в данном проекте. В итоге, в работе был применен вышеописанный продукт.

2.3.2 Проектирование первой СУБД

Для начала, опишем все объекты, которые предполагается использовать. Это: объект «бренд», «товар», «цена», «владелец», «контакт», «статистика», «пользователь».

1. Бренд. Включает в себя:
    - Id — уникальный индекс
    - Название — строку
    - Описание — общая текстовая информация
    - Локация — строка
    - Флаг «открыто» (1 — открыт, 0 — закрыт)
    - Список владельцев (объектов «владелец»)
    - Список контактов (объектов «контакт»)
    - Список статистических измерений (объектов «статистика)
    - Список товаров (объектов «товар)
    - Id пользователя — внешний ключ пользователя, который добавил данный бренд

    Реальная структура БД не подразумевает хранения списков, вместо этого используются внешние ключи к соответствующим таблицам.
2. Товар. Включает в себя:
    - Id — уникальный индекс
    - Название — строку
    - Описание — общая текстовая информация
    - Цену — объект типа «цена» (снова храним внешний ключ)
    - Пути до фотографий данного товара на сервере

Поскольку между ценами и товарами используется связь «один к одному», их можно было бы хранить в одной и той же таблице в соседних столбцах. Однако, с таким количеством полей не так удобно работать.

3. Цена. Включает в себя:
    - Id — уникальный индекс
    - Нижнюю грань ценового интервала — целое число
    - Верхнюю грань ценового интервала — целое число

    Объекты из данной таблицы связаны с объектами из таблицы товаров.
4. Владелец. Включает в себя:
    - Id — уникальный индекс
    - Имя — строку
    - Фамилию — строку
    - Отчество — строку
    - Секция «о себе» - общее текстовое описание

    Также есть неиспользуемое поле «история», оно зарезервировано под будущее расширение функционала
5. Контакт. Включает в себя:
    - Id — уникальный индекс
    - Тип — значение из enum типов контактов
    - Значение — собственно, контакт (почта, телефон, телеграм, …), в строковом виде

    Бренды и контакты связаны связью «многие ко многим».
6. Статистические измерения. В данной таблице хранятся статистически данные, которые владелец предпочел загрузить. Объекты включают:
    - Id — уникальный индекс
    - Название метрики — строка
    - Описание метрики — общая текстовая информация
    - Начало отсчета — дата, с которой проводились измерения
    - Конец отсчета — дата, до которой проводились измерения
    - Значение — итоговый результат, число с знаками после запятой

- Бренд, к которому это относится — внешний ключ

С таблицей брендов данные объекты связаны связью «один ко многим» (один бренд может иметь много записанных измерений)

7. Пользователь. Хранятся зарегистрированные на сайте пользователи. Включает в себя:
   - Id — уникальный индекс
   - Почта — email данного пользователя, он же логин
   - Пароль — хеш пароля данного пользователя, строка
   - Имя
   - Фамилия

По результатам структурирования получилась схема, представленная на рис.2

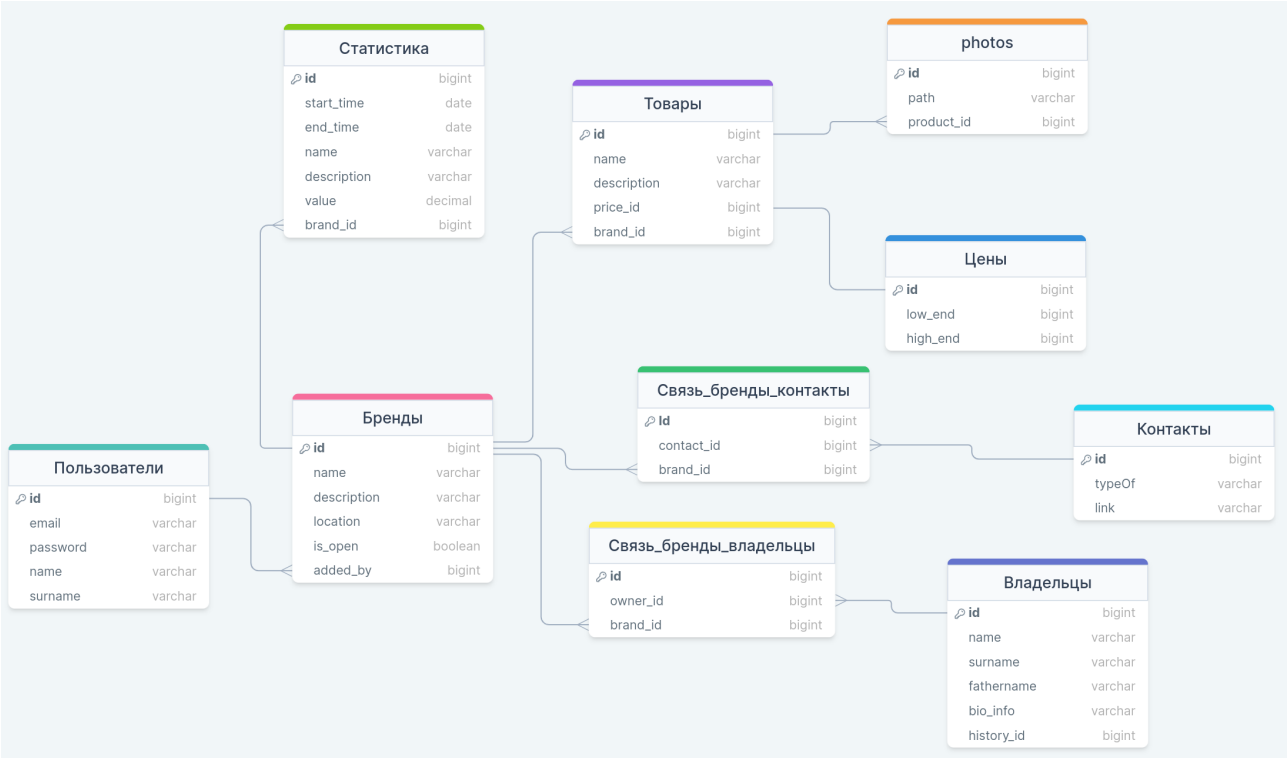


Рис.2 Схема базы данных

2.3.2 Проектирование второй СУБД

На цифровом портале для пользователя создана возможность регистрироваться. При входе пользователя в свой аккаунт, ему выдается сессия — UUID, т.е строка из 32 символов в 16-ричной системе счисления.

Принципиальным является то, что шанс коллизии при генерации случайных UUID пренебрежимо мал (0.00000006%), поэтому данные строчки можно использовать как уникальные идентификаторы каждой сессии.

Все сессии имеют одну фиксированную продолжительность по времени, при истечении которого, пользователю необходимо совершить вход в свой аккаунт заново, а объект сессии удаляется из базы данных. При таком подходе получается, что к базе данных с объектами пользовательских сессий будет генерироваться много простых запросов. Руководствуясь этим, можно заключить, что в данном случае нецелесообразно использовать SQL. Вместо этого было принято решение использовать TarantoolDB, NoSQL базу данных, хранящуюся в оперативной памяти, что позволяет значительно снизить время ответов на пользовательские запросы.

Надо заметить, что в данном случае часто используются другие похожие инструменты, такие как, например, Redis. Tarantool был выбран вместо него исключительно потому, что у меня уже был опыт работы с ним.
Схема получившейся таблицы представлена на рис.3

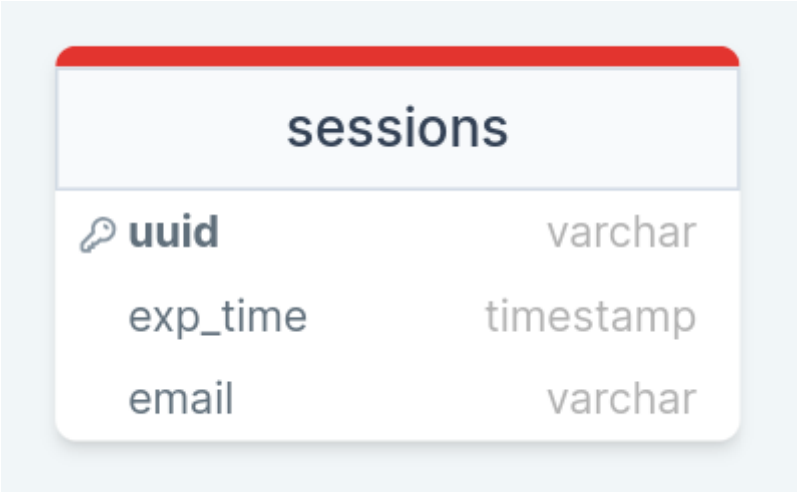


Рис.3 Схема БД с сессиями

2.3.3 Выбор языка программирования

Серверное приложение реализовано на языке Go. Это удобный, компилируемый императивный язык с элементами объектно-ориентированного подхода (отсутствует наследование и нет полноценного полиморфизма). Кроме

того, в него встроен сборщик мусора и используется строгая статическая типизация.

Благодаря компактному и лаконичному синтаксису код на языке Go очень легко читается и пишется, что позволяет добиться того же результата, что на других языках без использования сложных конструкций.

Другим важным преимуществом Go является безопасность. Язык предоставляет множество инструментов для обработки исключений. В это же время, стандартные и библиотечные функции возвращают возможные исключения, чтобы с ними было проще работать, поэтому шанс «падения» приложения из-за, например, некорректных данных значительно снижается.

В результате, невзирая на отсутствие полноценного ООП, язык Go с использованием интерфейсов и дженериков позволяет продуктивно работать с абстракциями.

2.3.4 Основные библиотеки

Для подключение к PostgreSQL базе данных использовались библиотеки go/sql и pq — стандартный в данной ситуации выбор. Данные библиотеки предоставляют все необходимые функции и регулярно обновляются.

Для подключения к TarantoolDB использовалась библиотека go-tarantool — как единственный полностью реализованный контроллер для этой СУБД.

Веб-сервер с REST API реализованы с помощью библиотеки go-fiber, очень мощного решения, дающего разработчику «из коробки» большое число функций. Приоритет выбора также обусловлен тем, что данная библиотека отлично задокументирована и регулярно обновляется, а также дает самую большую производительность.

2.4 Клиентская часть

Клиентом в данном случае является веб-сайт, размещенный в сети Интернет. Он предоставляет удобный графический интерфейс для использования всех функций приложения. Веб-сайт реализован с использованием фреймворка VueJS и языков HTML, CSS.

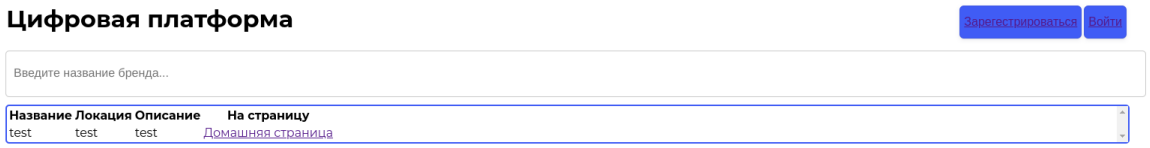
Фреймворк VueJS был выбран за:

- простоту в реализации реактивности данных
- виртуальный DOM, что позволяет сразу видеть изменения в процессе разработки
- удобную компонентную структуру
- обширную развитую экосистему
- хорошую документацию

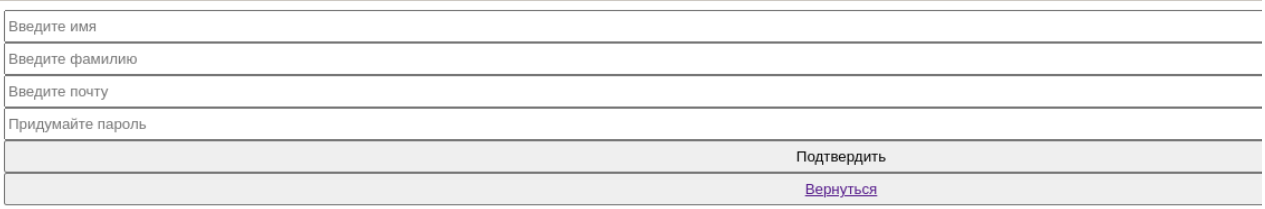
Чистый JS хоть и позволяет реализовать тот же функционал, требует для этого намного больше усилий.

3. Функциональный пример

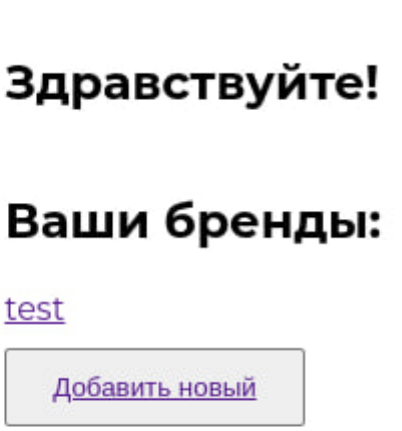
Главная страница:



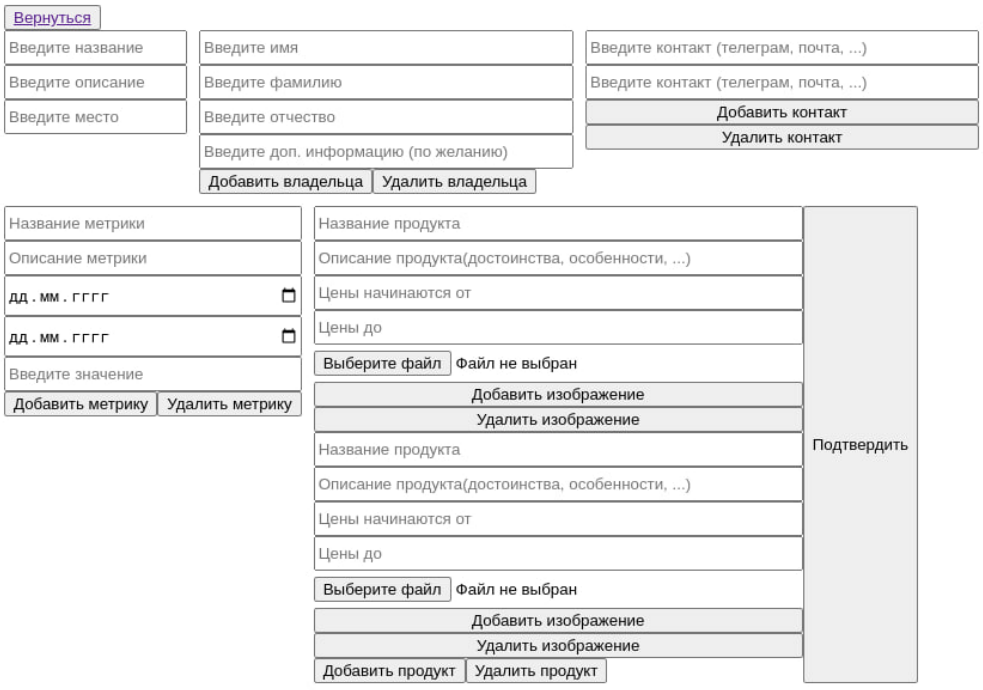
Регистрация:



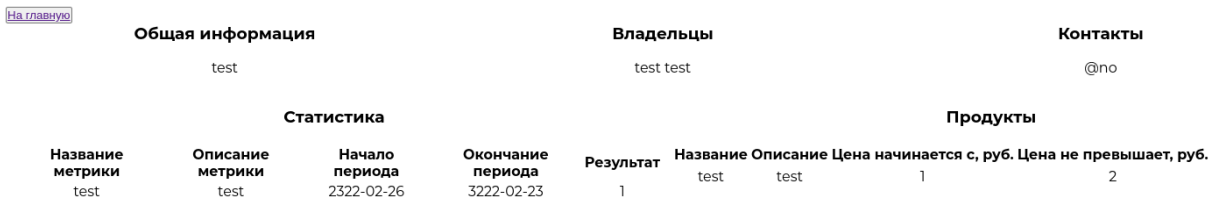
Личный кабинет:



Добавление новой компании:



Страница бренда:



Заключение

В результате проведенного анализа, разработке конструкторского раздела, в работе представлен программный комплекс, реализующий функции реестра и акселератора компаний и брендов.

Этот комплекс позволяет:

- добавлять и редактировать свои бренды
- просматривать добавленные другими пользователями бренды
- добавлять описание товаров и фотографии
- выполнять поиск компании/бренда по названию

Весь код написан самостоятельно, с использованием открытых источников.

Полный код серверного приложения:

https://github.com/verybadcoder01/accelerator

Полный код клиентской части:

https://github.com/verybadcoder01/accelerator-frontend

Список использованных источников

1. Документация языка Golang [Электронный ресурс] URL: https://go.dev/doc/ (Дата обращения 15.02.2024)
2. Справочник по языку Golang, [Электронный ресурс] URL: https://go101.org/ (Дата обращения 13.02.2024)
3. Документация библиотеки Go-fiber, [Электронный ресурс] URL: https://docs.gofiber.io/ (Дата обращения 25.12.2023)
4. Документация языка PostgreSQL [Электронный ресурс] URL: https://www.postgresql.org/docs/ (Дата обращения 15.02.2024)
5. Документация по СУБД Tarantool [Электронный ресурс] URL: https://www.tarantool.io/ru/doc/latest/ (Дата обращения 19.02.2024)
6. Документация по фреймворку VueJS [Электронный ресурс] URL: https://vuejs.org/guide/introduction.html (Дата обращения 10.02.2024)

7. Справочник по разработке веб-сайтов [Электронный ресурс] URL: https://developer.mozilla.org/ru/ (Дата обращения 19.02.2024)
8. Мартин Р., Чистая архитектура. Искусство разработки программного обеспечения // Издательский дом «Питер» - 2018