Санкт-Петербургский государственный университет

Кафедра Системного Программирования

Группа 21.Б15-мм

Разработка серверной части в проекте Giftee

Асеев Григорий Александрович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
старший преподаватель кафедры системного программирования, И. В. Зеленчук

Санкт-Петербург
2024

Оглавление

Введение

Выбор подарка является нетривиальной задачей, учитывая богатый ассортимент товаров и услуг. Выявлено, что 62 процента людей в одной из крупных стран[16] сталкиваются с проблемами выбора подарка. Для эффективного решения этой проблемы необходим персонализированный подход, который учитывает уникальные интересы и предпочтения людей. В связи с этим было принято решение разработать мобильное приложение, использующее методы машинного обучения, чтобы обеспечить удовлетворяющий запросам потребителей подбор подарков.

При изучении различных существующих подходов к подбору подарков был обнаружен ряд существенных недостатков. Данные сервисы обладают ограниченными возможностями персонализации подарков и, зачастую, предлагают ограниченные варианты на основе общих тематик или популярных предпочтений. Более того, существующие системы игнорируют динамические изменения в предпочтениях пользователей, что приводит к устаревшим рекомендациям. Кроме того, мало известно о методах применения нейронных сетей для создания рекомендательных систем в контексте подарочных сервисов. Это оставляет пространство для дальнейших исследований в этой области.

На основании вышесказанного, можно предположить, что разработка мобильного приложения с системой персонализированных рекомендаций позволит предложить наиболее подходящие варианты подарков, сокращая усилия и время, затрачиваемые на поиск нужного подарка. Регистрируясь в приложении и проходя специальное тестирование, пользователь становится источником данных, на основе которых будет создан уникальный цифровой профиль его интересов. Это может повысить точность и персонализацию рекомендаций подарков, учитывая динамические изменения в предпочтениях пользователя.

Для достижения поставленных целей необходимо реализовать серверную часть, обеспечивающую взаимодействие с мобильным приложением и рекомендательной системой. Также планируется использовать средства гранта от «Фонд-М»[17] для сбора данных, необходимых для

обучения и разработки рекомендательной системы, основанной на методах машинного обучения и алгоритмах фильтрации.

Дата сборки: 4 июня 2024 г.

1. Постановка задачи

Целью работы является разработка серверной части мобильного приложения для подбора подарков.

Для ее выполнения поставлены следующие задачи:

1. Провести опрос для выявления проблем при подборе подарков
2. Провести анализ существующих решений
3. Составить требования к серверной части
4. Спроектировать архитектуру серверной части
5. Спроектировать и подключить базу данных
6. Реализовать и развернуть на удаленном сервере бэкенд
7. Выполнить тестирование приложения

2. Обзор

Этот раздел представляет обзор уже существующих решений для подбора подарков, анализ проведенного опроса на тему подбора подарков, а также обзор используемых технологий, необходимый для понимания работы.

2.1. Существующие решения

Более детальный анализ конкурентов доступен по ссылке[1].

1. *Ozon*[11] — известный и крупный маркетплейс в России.
2. *Flowwow*[6] — онлайн-сервис для покупки товаров в локальных магазинах, специализирующийся на цветах.
3. *Podarki*[12] — интернет-магазин сувениров и подарков.

Вывод из анализа конкурентов показывает, что большинство существующих приложений подбора подарков сталкиваются с отсутствием рекомендательной системы и персонализированного подбора, что ограничивает их способность предлагать точные и уникальные рекомендации, соответствующие индивидуальным потребностям пользователей.

2.2. Опрос

Данные полученные с опроса[2] и структурированный результат[3] доступны по ссылкам в сносках.

Проанализировав график, можно заключить, что в контексте выбора подарков отсутствуют инновационные идеи, а также существует повсеместное беспокойство и сомнение в том, что выбранный подарок действительно понравится его получателю.

[1] https://docs.google.com/spreadsheets/d/1wurP_iqfqNhnmnvbK5glriuisa6i0ciBZ2X8dl2-E2Q/edit#gid=1992277964

[2] https://docs.google.com/spreadsheets/d/1wurP_iqfqNhnmnvbK5glriuisa6i0ciBZ2X8dl2-E2Q/edit#gid=581563839

[3] https://docs.google.com/spreadsheets/d/1wurP_iqfqNhnmnvbK5glriuisa6i0ciBZ2X8dl2-E2Q/edit#gid=926812180

Рис. 1: Трудности с выбором подарка

2.3. Требования к системе

1. Требуется поддержка безопасной авторизации пользователей
2. Серверная часть должна поддерживать функциональность поиска подарков по различным категориям и цене
3. Должна быть интеграция с сервисом рекомендательной системы с алгоритмами машинного обучения для анализа предпочтений пользователей
4. Сервер должен поддерживать хранение и управление социальными данными, такими как связи между пользователями и их взаимодействия, а также обладать функцией запроса проверки подарка другу
5. Требуется поддерживать подарки в избранном и вишлисте пользователя

2.4. Используемые технологии

1. *Golang*[7] — компилируемый многопоточный язык программирования. Выбор данного языка обусловлен его производительностью, масштабируемостью, простотой и надежностью.
2. *Chi*[4] — легковесный и высокопроизводительный маршрутизатор HTTP для языка программирования Go. Он широко используется

во многих проектах на языке Go благодаря гибкости и расширяемости.

3. *JWT-Go*[9] — библиотека на Go для создания и верификации JSON Web Tokens (JWT). Она применяется в проекте для обеспечения безопасной и авторизованной передачи данных между серверной частью и клиентским приложением.

4. *PostgreSQL*[13] — система управления реляционными базами данных, используемая в проекте для хранения данных приложения. Была выбрана как основная СУБД из-за её надежности и расширяемости.

5. *Gorm*[8] — библиотека для работы с базами данных в языке программирования Go. GORM позволяет создавать объекты базы данных без необходимости писать SQL-запросы вручную.

6. *Redis*[14] — это высокопроизводительный кеш данных с открытым исходным кодом, поддерживающий различные структуры данных, такие как строки, хэши, списки, множества и сортированные множества.

7. *Air*[2] — утилита командной строки, которая обеспечивает перезагрузку приложений Go в реальном времени.

8. *Docker*[5] — платформа, позволяющая упаковывать приложения и их зависимости в контейнеры. В проекте он используется для запуска и управления приложением в изолированной среде, обеспечивая однородность окружения разработки и развертывания приложения.

3. Реализация

В данном разделе представлено описание компонентов серверной части, ее архитектуры, структура базы данных, оптимизация работы с сервисом, тестирование серверной части и процесс развертывания.

3.1. Основная идея

Входа в приложение предполагает, что при регистрации пользователей через ввод электронной почты и пароля, оригинальный пароль будет заменяться хэшированным на основе алгоритмов криптографии. Аутентификация пользователей происходит путем сравнения введенных ими пароля с тем, что хранится в базе данных, при этом затем выдается JWT-токен, который сохраняется через cookie. Авторизация осуществляется путем проверки валидности полученного JWT-токена.

Рекомендательная система предполагает сбор результатов тестирования пользователя для последующей обработки и передачи на рекомендательный сервис, который в свою очередь обеспечивает персонализированные рекомендации товаров в соответствии с интересами пользователя.

Магазин будет осуществлять выгрузку товаров из базы данных, предоставлять функциональность поиска по имени, фильтрации товаров по цене и категориям, а также опции сортировки товаров по цене.

Социальная сеть включает возможность добавления друзей через поиск по фамилии и имени, дате рождения и другим характеристикам, а также отображение профилей друзей, включая базовую информацию и список желаемых подарков.

Профиль пользователя предоставляет возможности взаимодействия с корзиной — включая добавление и удаление товаров, имеющихся в наличии, в хранилище корзины конкретного пользователя, предоставление информации о точках доставки и выполнение оплаты заказа. Также в рамках данного сервиса пользователи могут изменять свои личные данные, включая пароль.

3.2. Архитектура серверной части

Для реализации проекта был выбран архитектурный паттерн Model-View-Controller (MVC)[10], который предоставляет шаблон для разделения приложения на компоненты, отвечающие за модели данных, представления и контроллеры.

Данный выбор обусловлен её свойствами гибкости и масштабируемости. Модель данных (Model) отвечает за обработку и хранение данных о подарках, предпочтениях пользователей и других сущностях. Представления (View) отвечают за отображение данных и взаимодействие с пользовательским интерфейсом. Контроллеры (Controller) обрабатывают запросы от пользователей, взаимодействуют с моделью и обновляют представления. Это позволяет эффективно управлять логикой, обработкой данных и взаимодействием с пользовательским интерфейсом, а также упрощает разработку и сопровождение приложения.

Для взаимодействия серверной части с внешним миром было решено использовать принципы REST (Representational State Transfer), который определен для построения масштабируемых и распределенных приложений.

Запросы к серверу будут отправляться по протоколу HTTP, а данные будут передаваться в формате JSON (JavaScript Object Notation). Это универсальный формат данных, который позволяет совершить обмен информацией между клиентскими приложениями и сервером, расширить и адаптировать функциональность при необходимости.

Сервисы для серверной части представляют собой отдельные модули, обеспечивающие конкретные функциональные возможности приложения. Были реализованы следующие сервисы: сервис авторизации отвечает за управление процессом аутентификации пользователей, сервис магазина обеспечивает управлением каталогом товаров, также сервис профиля обеспечивает управление данными пользователя, включая пополнение и изменение личной информации, а также позволяет взаимодействовать с корзиной. Каждый такой сервис представляет собой независимый модуль в рамках серверной части приложения, обеспечи-

вая свои функции.

ParserWB[4] был реализован на языке Python и отвечает за парсинг товаров по заданным запросам от RecSys. ParserWB осуществляет сбор товаров с учётом диапазонов цен, а также выполняет сортировку по популярности и цене. Таким образом, он обеспечивает актуальные данные для опросов и рекомендаций.

Реализован сервис Survey, который отвечает за корректировку интересов пользователей. Сервис предоставляет возможность составления опросов, в ходе которых пользователи могут положительно или отрицательно реагировать на различные товары. Результаты этих опросов сохраняются и используются для обучения модели рекомендательного сервиса (RecSys). Survey взаимодействует с несколькими другими сервисами, включая RecSys и ParserWB, для обеспечения корректной работы всей системы. На рисунке 2 представлена диаграмма активностей для функции составления опроса.

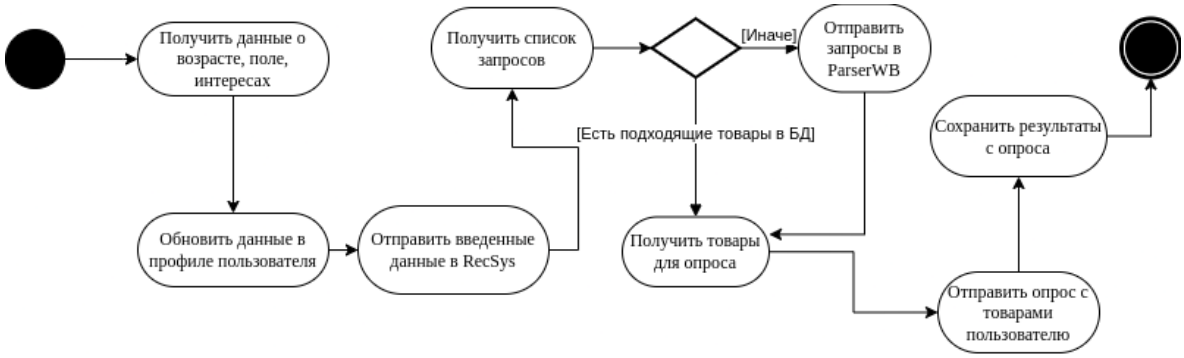

Рис. 2: Опрос для сбора интересов и понравившихся товаров

Сервис Match выполняет две ключевые функции: подбор подарков для друга и проверка товара другом. Match собирает информацию о друге через пользователя или его профиль в приложении. На основе собранных данных и с использованием сервисов RecSys и ParserWB, Match предлагает подходящие подарки. Пользователь может отправить товар на проверку другу. Друг получает уведомление с предложенным товаром от анонимного пользователя и может положительно или отрицательно на него реагировать. Отправляющий пользователь видит

[4] https://github.com/grrrance/ParserWB

реакцию друга, что помогает ему принимать решения о покупке. На рисунке 3 представлена диаграмма активностей для функции подбора подарка.

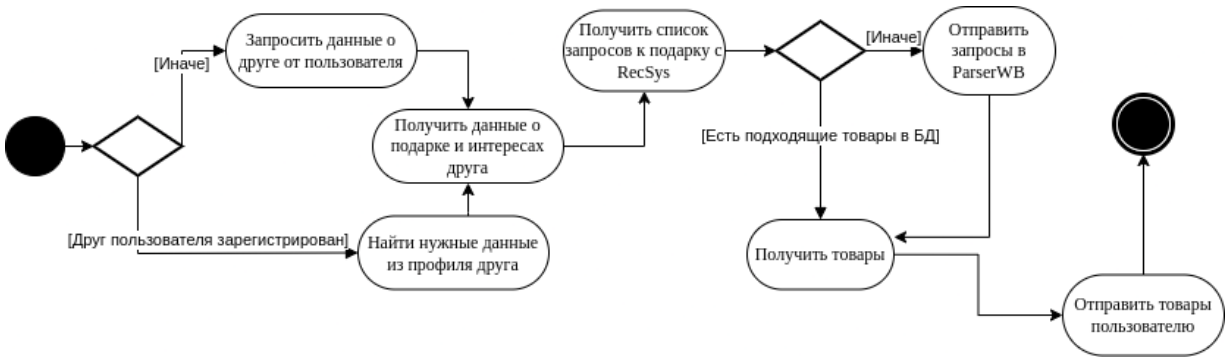

Рис. 3: Поиск подарков для друга

Для того, чтобы пользователи могли использовать функции Match, а также видеть вишлисты товаров друзей, был реализован сервис социальной сети.

Реализованна платежная система с помощью платформы Stripe для будущей монетизации приложения.

Также был реализован сервис для получения списка названий и координат городов России. Этот сервис предоставляет необходимую информацию о городах для различных целей, включая улучшение пользовательского опыта и обеспечение локализации контента.

Controllers (Контроллеры): часть приложения, которая обрабатывает запросы от пользователей и взаимодействует с логической частью. Контроллеры получают входные данные от пользователей через HTTP-запросы, вызывают соответствующие обработчики для выполнения требуемых операций и формируют ответы для отправки обратно пользователю.

Use case (прецедент использования): внутри сервиса представляет собой конкретный сценарий или функциональность, которую этот сервис реализует. Это описание того, как сервис будет взаимодействовать с другими компонентами системы и выполнять определенные задачи.

Repositories (Репозитории): интерфейсы, описывающие методы доступа и операции с базой данных. Репозитории представляют абстрак-

цию для работы с хранилищем данных и обеспечивают стандартизированный способ получения, сохранения, обновления и удаления сущностей из базы данных.

Entities (Сущности): основные объекты и данные, с которыми работает приложение. Сущности позволяют представить данные в приложении и определять их структуру и отношения. Они описывают пользователей, подарки, интересы и другие ключевые объекты бизнес-логики данного проекта.

DTO (Data Transfer Object) — объект, используемый для передачи данных между слоями программной системы.

На рисунке 4 представлена описываемая архитектура.

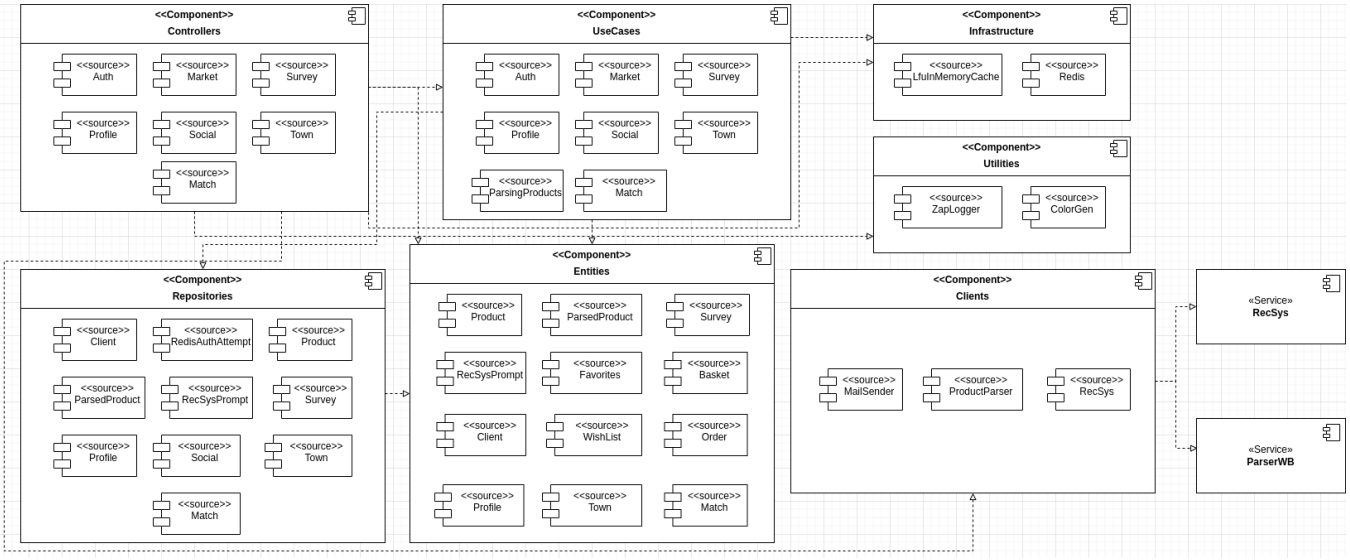

Рис. 4: Диаграмма компонентов сервера

3.3. База данных

База данных приложения реализует следующую структуру, которая представлена на рисунке 5:

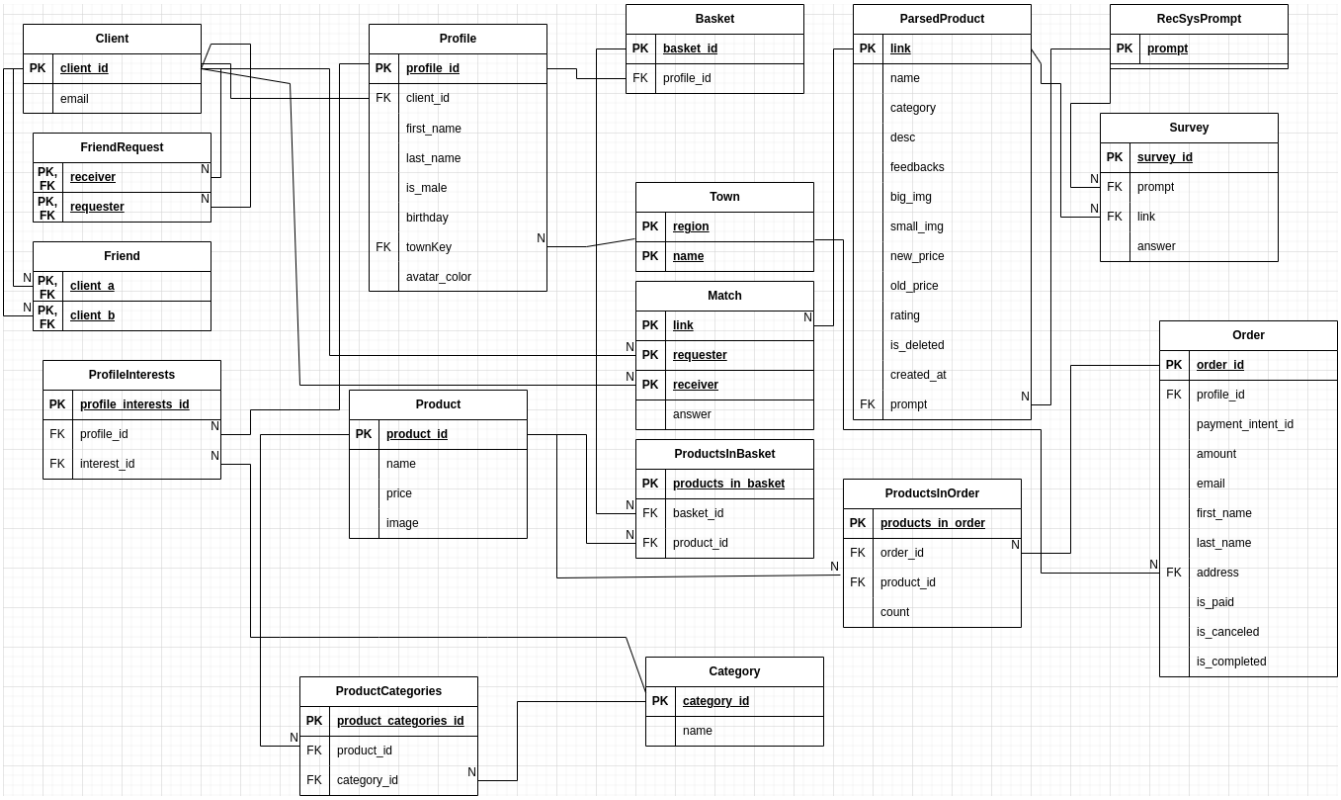

Рис. 5: Модель данных

Имеется сущность Client, которая хранит информацию о входных данных пользователя. Эта сущность связана с сущностью Profile, где хранится общая информация о клиенте, и данным отношением является отношение один-к-одному, аналогично связаны Profile и модель Basket, которая хранит список продуктов, готовых к оплате пользователем. Также существует связь многие-ко-многим среди таблиц Interest и Profile, Product и Category, а также Basket и Product, их связь реализована через промежуточные таблицы, содержащие идентификаторы из двух таблиц. Присутствует таблицы заявок в друзья и друзей пользователя с ключами Client. Интересы представляют из себя категории понравившихся пользователю товаров. Для сохранения понравившихся товаров из опроса была создана таблица Survey, а для реализации

функции проверки подарков в сервисе Match была описана таблица Match, которая состоит из ключей товара, пользователей и ответа получателя.

3.4. Оптимизация работы сервиса ParserWB

Сервис ParserWB выдаёт товары с задержкой из-за ограничений на парсинг, установленных Wildberries для защиты от частых запросов. Чтобы уменьшить количество обращений к ParserWB и снизить нагрузку на сервер, было реализовано следующее:

- Два уровня кеширования:
  - Кеширование ответов на запросы к серверу с помощью Redis
  - In-memory кеширование на запросы к репозиториям с вытеснением Least Frequently Used[5]
- Разработка репозитория распарсенных товаров с возможностями:
  - Безопасное удаление
  - Прогрев базы данных

Redis был выбран для кеширования ответов на запросы к серверу из-за его высокой производительности, надежности и гибкости. Redis работает в оперативной памяти, что обеспечивает мгновенный доступ к данным и высокую скорость обработки запросов. Поддерживает множество типов данных, что позволяет хранить различные структуры данных. Redis может сохранять данные на диск, что обеспечивает их сохранность при перезапуске сервера.

In-memory кеширование на уровне приложения с использованием алгоритма LFU было выбрано для дополнительного уровня кеширования: Алгоритм LFU (Least Frequently Used): вытесняет наименее часто

[5] https://github.com/grrrance/lfu-in-memory

используемые элементы, что позволяет хранить в памяти наиболее востребованные данные и оптимизировать использование памяти. Использование in-memory кеша позволяет значительно снизить количество обращений к базе данных, что уменьшает нагрузку на сервер и улучшает общую производительность системы. Уменьшение задержки получения товаров

Для уменьшения задержки получения товаров из сервиса ParserWB, который требует длительного времени на парсинг, были реализованы следующие меры:

Репозиторий обеспечивает безопасное удаление устаревших или неактуальных данных без нарушения работы системы. Он не удаляет данные полностью, а только помечает их, как удаленные. Это сделано по причине неактуальности товаров с Wildberries со временем. Если спустя время мы получим те же самые товары, что хранятся с удаленной меткой, то такие товары восстановятся.

Реализована автоматическая очистка неиспользуемых товаров поддержания актуальности данных. На рисунке 6 представлен код, демонстрирующий данные возможности.

```go
func (p *parsingRepo) RegisterProduct(ctx context.Context, product *models.ProductFromParser) (*models.ProductFromParser, error) {  grrrance *
    exists, err := p.existsProduct(ctx, product)
    if err != nil {
        return nil, err
    }

    var res *gorm.DB

    if exists {
        res = p.db.Model(&models.ProductFromParser{Link: product.Link}).Update( column: "is_deleted",  value: false)
        if res.Error != nil {
            return nil, res.Error
        }
        res = p.db.Model(&models.ProductFromParser{Link: product.Link}).Update( column: "created_at", time.Now())
    } else {
        res = p.db.Create(product)
    }

    return product, res.Error
}

func (p *parsingRepo) startGC() {  1 usage  new *
    for range time.Tick(p.cfg.Parsing.CleanupInterval) {
        curr := time.Now()
        p.db.Model(&models.ProductFromParser{}).Where( query: "created_at < ? AND is_deleted = false", curr.Add(-p.cfg.Parsing.LiveInterval)).Update( column: "is_deleted",  value: true)
    }
}
```

Рис. 6: Сборка мусора и безопасное удаление товаров

Предварительное заполнение базы данных товарами из ParserWB позволяет ускорить доступ к товарам и уменьшить задержки при выполнении запросов. Прогрев осуществляется в фоновом режиме, обеспечивая наличие актуальных данных в базе к моменту запроса от пользователя. На рисунке 7 представлена функция, которая обращается к ParserWB в фоновом режиме и заполняет товарами по возможным запросам из RecSys.

```go
func WarmUp(parsingUC parsing.UseCase, config *config.Config) error {  1 usage   gmrance *
    ctx := context.Background()
    wishes, err := readWishes()
    if err != nil { return err }

    for range time.Tick(config.Parsing.WarmInterval) {
        for _, w := range wishes {
            for minPrice, maxPrice := uint64(0), uint64(1000); maxPrice < 15001; minPrice, maxPrice = minPrice+1000, maxPrice+1000 {
                products, _ := client.ParseProducts(ctx, config, &modelsMarket.ParserFilter{
                    Prompt:    w.Name,
                    MinPrice:  minPrice,
                    MaxPrice:  maxPrice,
                    Count:     5,
                    IsSorting: true,
                    SortByAsc: true,
                })
                for _, product := range products {
                    if _, err := parsingUC.RegisterProduct(ctx, &product); err != nil { return err }
                }
            }
        }
    }

    return nil
}
```

Рис. 7: Прогрев товаров

3.5. Тестирование

Серверная часть была покрыта на 78% юнит тестами с помощью GoMock [6]

Был написан интеграционный тест на запрос поиска подарка. Первоначально среднее время ответа на данный запрос занимало 1 минуту 31 секунду. После внедрения кеширования и сохранения товаров в базу данных, среднее время ответа сократилось до 7 миллисекунд.

[6]https://github.com/uber-go/mock

Реализованные меры, включая два уровня кеширования и оптимизацию базы данных, позволили значительно улучшить производительность сервиса и уменьшить задержки при получении товаров.

3.6. Развертывание серверной части

Для развертывания бэкенда приложения на удаленном сервере были разработаны следующие шаги:

1. Создание Dockerfile[7] для сборки и упаковки приложения в контейнер. В этом файле были указаны необходимые шаги и зависимости для сборки и запуска приложения.

2. Использование Docker Compose для создания контейнеров. Docker Compose[8] файл был подготовлен для создания двух контейнеров — один для PostgreSQL и один для сервисов приложения. Это позволяет определить зависимости и связи между сервисами.

3. Внедрение утилиты перезагрузки приложения. Для облегчения разработки и внесения изменений в код приложения была использована утилита перезагрузки, такая как Air, что упростило процесс перезапуска приложения после модификаций.

4. После подготовки окружения процесс развертывания контейнеров с использованием Docker был инициирован на удаленном сервере.

Эти шаги обеспечили гибкое и эффективное развертывание серверной части приложения на хостинге, обеспечивая удобство управления и эффективное использование ресурсов.

3.7. Чат бот

В ходе работы был разработан чат-бот[9], написанный на Python с использованием библиотеки Aiogram[1].

[7] https://docs.docker.com/engine/reference/builder/
[8] https://docs.docker.com/compose/
[9] https://t.me/gifteebot

Этот бот использует API ChatGPT[3] для подбора подарков на основе введенных пользователем данных, а также парсит необходимую информацию о подарках на платформе Wildberries[15], предоставляя идеи и ссылки на конкретные товары. Бот был реализован с целью проверки основной идеи проекта и оценки его потенциальной эффективности.

Кроме того, проведен опрос[10], в котором приняло участие 21 человек, с целью оценки взаимодействия с чат-ботом. Результаты опроса доступны по следующей ссылке[11]. Проанализировав эти результаты, можно сделать вывод, что бот предоставляет релевантные подарки и идеи, соответствующие запросам пользователей.

[10] https://forms.gle/b4rK8hdcqThL2XCQ9

[11] https://docs.google.com/spreadsheets/d/1wurP_iqfqNhnmnvbK5glriuisa6i0ciBZ2X8dl2-E2Q/edit#gid=870807056

Заключение

За осенний семестр были достигнуты следующие результаты:

1. Проведен опрос, в ходе которого выявлены проблемы при подборе подарков
2. Проведен анализ существующих решений
3. Составлены требования к серверной части
4. Разработана архитектура серверной части
5. Спроектирована и подключена база данных
6. Реализован чат бот
7. Реализована и развернута на удаленном сервере основная часть бэкенда приложения, обладающая функциональностью сервисов авторизации, магазина и профиля.

В весеннем семестре были выполнены следующие задачи:

1. Реализованы сервисы с опросами, функциями подбора и проверки подарков, социальная сеть, города, изменена авторизация, добавлена платежная система
2. Реализован парсер товаров с wildberries
3. Реализовано кеширование с помощью Redis и In-memory и прогрев базы данных рекомендуемыми товарами, а также их очистка
4. Написаны тесты для серверной части

Код реализации мобильного приложения является коммерческой тайной, поэтому предоставляется текущая мобильная версия приложения[12], которая взаимодействует с удаленным сервером.

[12] https://clck.ru/3B5CKd

Список литературы

[1] Aiogram. — URL: https://aiogram.dev/ (дата обращения: 2023-12-24).

[2] Air. — URL: https://github.com/cosmtrek/air (дата обращения: 2023-12-24).

[3] Chat GPT, openai. — URL: https://chat.openai.com/ (дата обращения: 2023-12-24).

[4] Chi. — URL: https://go-chi.io/#/ (дата обращения: 2023-12-24).

[5] Docker. — URL: https://www.docker.com/ (дата обращения: 2023-12-24).

[6] Flowwow. — URL: https://flowwow.com/ (дата обращения: 2023-12-24).

[7] Golang. — URL: https://go.dev/ (дата обращения: 2023-12-24).

[8] Gorm. — URL: https://gorm.io/index.html (дата обращения: 2023-12-24).

[9] JWT-Go. — URL: https://github.com/golang-jwt/jwt (дата обращения: 2023-12-24).

[10] The Model-View-Controller (MVC). — URL: https://ru.wikipedia.org/wiki/Model-View-Controller (дата обращения: 2023-12-24).

[11] Ozon. — URL: https://github.com/MathAndMedLab/Medical-Web-App (дата обращения: 2023-12-24).

[12] Podarki. — URL: https://podarki.ru/ (дата обращения: 2023-12-24).

[13] PostgreSQL. — URL: https://www.postgresql.org/ (дата обращения: 2023-12-24).

[14] Redis. — URL: https://redis.io/ (дата обращения: 2024-06-02).

[15] Wildberries. — URL: https://www.wildberries.ru/ (дата обращения: 2023-12-24).

[16] Отчет проблем выбора подарков. — URL: https://www.businesswire.com/news/home/20221114005385/en/The-Majority-of-Americans-64.2-Report-They-Need-Help-When-It-Comes-to-Holiday-Gifting-Accord (дата обращения: 2024-06-04).

[17] Фонд содействия инновациям. — URL: https://online.fasie.ru/m/ (дата обращения: 2023-12-24).