После формирования данного списка пользователь может уже непосредственно просматривать каждый экспонат подробнее. Каждый элемент списка связан с соответствующей записью в базе данных. При нажатии на любой экспонат (для примера «Стример» из категории «Хранение информации» или «Камера электроника л-50» из категории «Видеотехника») пользователю открывается окно просмотра подробной истории элемента, в котором он может увидеть название экспоната, категорию и прочитать историческую справку о данном экспонате.

Созданное приложение поможет пользователям виртуально познакомиться с экспонатами экспозиции и в случае заинтересованности, посетить факультет физики и информационных технологий ГГУ имени Ф. Скорины и увидеть экспонаты в «живую».

Литература

1. Гриффитс, Д. Head First. Программирование для Android / Д. Гриффитс, Д. Гриффитс – Питер: 2018. – 140 с.

2. Steele, J. The Android Developer’s CookBook / J. Steele, N. To – Pearson Education - Inc. 2011. – 256 p.

3. Пискунов, А.Г. Руководство по SQLite для пользователей Windows / А.Г. Пискунов. – 59 с.

А. С. Яросевич (ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **В. В. Грищенко**, ст. преподаватель

СИСТЕМА УЧЕТА ДОХОДОВ И РАСХОДОВ НА БАЗЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Введение

Если вспомнить что было 5 лет назад, то можно заметить, как сильно с тех времен поменялось отношение к архитектуре микросервисов. Сначала микросервисы были очень популярны. [1] После успеха таких компаний как Amazon и Netflix разработчики приняли решили, что фактически микросервисная разработки ничем не отличается от разработки приложений. В настоящее время большинство осознают, что микросервисы являются по сути новым архитектурным стилем, который очень эффективен для решения большинства задач, но они также имеют свои плюсы и минусы. [2]

Основа

Целью данного веб-приложения является повышения финансовой грамотности клиентов, которые будут в дальнейшем пользоваться данным продуктом. Что по итогу развития финансовой грамотности предоставляет возможность сохранять и улучшать финансовое благополучие.

Со стороны разработки используется принцип проектирование систем на базе микросервисной архитектуры. Для совместной работы микросервисов применяется набор основных практик и паттернов из мира микросервисной архитектуры. [3] Большинство из этих подходов представлены в Spring Cloud (благодаря интеграции с продуктами Netflix OSS) — на деле это дополнительные библиотеки, которые расширяют возможности Spring Boot. Схема взаимодействия микросервисов представлена на рисунке 1.

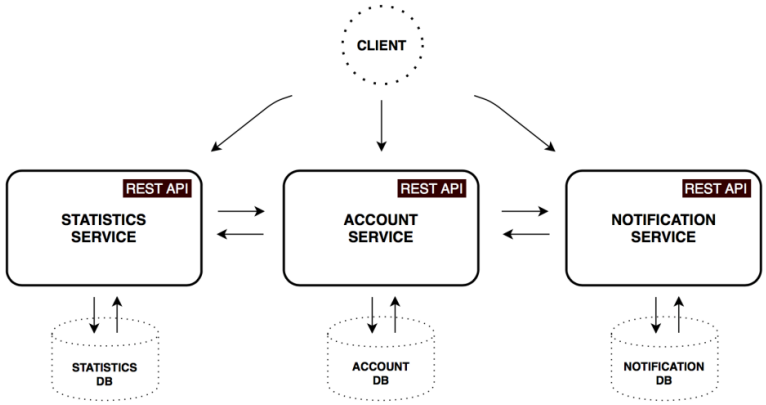

Рисунок 1 – Схема взаимодействия микросервисов

Приложение состоит из трех главных микросервисов, каждый отвечает за определенную бизнез-логику. Для каждого микросервиса предоставлена отдельная база данных, поэтому доступ к данным возможно получать через API приложения. [4]

Как основную базу данных для каждого микросервиса используется MongoDB.

Аккаунт сервис – отвечает за реализацию бизнес-логики и валидации по сохранению накоплений, расходов и доходов, а также имеет возможность настраивать аккаунт.

Аккаунт сервис имеет следующий функционал:

1) для указанного аккаунта получать необходимые данные;

2) получать данные для текущего аккаунта;

3) получать данные для аккаунта, который используется для демонстрации;

4) сохранение текущих данных аккаунта;

5) регистрация нового аккаунта.

Сервис сбора статистики – позволяет делать расчет основных параметров аккаунта, конвертирует значения к единой валюте и сохраняет данные в удобном виде для дальнейшего анализа. Полученный результат будет использоваться для отображения статистики пользователю и метрики за прошедшее время, а также для простейших прогнозов будет отображаться экстраполяция.

Сервис сбора статистики предоставляет следующий функционал:

1) для указанного аккаунта получать его статистику;

2) для текущего аккаунта получать статистику;

3) для демонстрационного аккаунта получать статистику;

4) создание и обновление временной точки для выбранного аккаунта.

Сервис уведомлений – позволяет хранить настройки уведомлений (периодичность напоминаний, как часто делать бэкапы). Осуществляет рассылку электронных сообщений по расписанию, предварительно собрав нужные данные у необходимых сервисов, если это требуется.

Сервис уведомлений выполняет следующие задачи:

1) получать настройки уведомлений для данного аккаунта;

2) сохранять настройки уведомлений для данного аккаунта.

Общение между сервисами упрощено за счет использования только синхронных rest-запросов.

Заключение

Отказоустойчивость приложения было достигнуто путем использования микросервисной архитектуры. Этот подход проектирования приложения, при которой несколько сервисов общаются друг с другом определенным образом, при помощи RESTful web-сервисов. [5] Главной чертой является то, что каждый микросервис позволяет разворачиваться и обновляться независимо друг от друга.

Литература

1 Микросервисная архитектура // Википедия [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Микросервисная_архитектура – Дата доступа: 22.03.2020.

2 Spring Cloud // Spring [Электронный ресурс] – URL: https://spring.io/projects/spring-cloud – Дата доступа: 22.03.2020

3 Веб-служба // Википедия [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Веб-служба – Дата доступа: 22.03.2020

4 MongoDB // Википедия [Электронный ресурс] – URL: https://en.wikipedia.org/wiki/MongoDB – Дата доступа: 22.03.2020

5 REST // Википедия [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/REST – Дата доступа: 22.03.2020