МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ
Факультет информационных систем и технологий
Кафедра «Информационные системы»
Дисциплина «Экономика и управление проектом»

Сервис автоматического код-ревью.
СТО.

Выполнил:
студент гр. ПИбд-41
Аглиулин Р. Р.
Проверил:
Желепов А. С.

Ульяновск, 2021 г

1. Реализация всех задумок СРО в виде продуманной архитектуры проекта.

В качестве архитектуры приложения мною была выбрана микросервисная архитектура, потому что она имеет следующие преимущества:

- Гибкий график релизов (каждый сервис имеет свой график релизов)
- Легкий порог вхождения в разработку отдельно сервиса (не придется изучать структуру всего приложения в целом, достаточно погрузиться в предметную область одного сервиса, сервисы между собой изолированы и взаимодействуют через четко определенные контракты)
- Возможность горизонтального масштабирования
- Сервисы могут быть написаны на разных языках

Микросервисная архитектура имеет свои издержки, связанные с DevOps, но в перспективе они окупятся.

На диаграмме ниже указаны следующие компоненты:

- PostgreSQL – реляционная СУБД
- Сервис пользователя – микросервис, отвечающий за процессы аутентификации, авторизации, хранения настроек пользователя.
- Сервис обработки событий – микросервис, принимающий по WebHook события от git-хостингов (GitHub, GitLab, BitBucket, etc), который фильтрует и преобразовывает их к единому формату, а затем отправляет их в шину данных
- Сервисы обработчики кода – набор типовых микросервисов, принимающих из шины данных файлы исходного кода с метаинформацией и отдающих результат анализа обратно

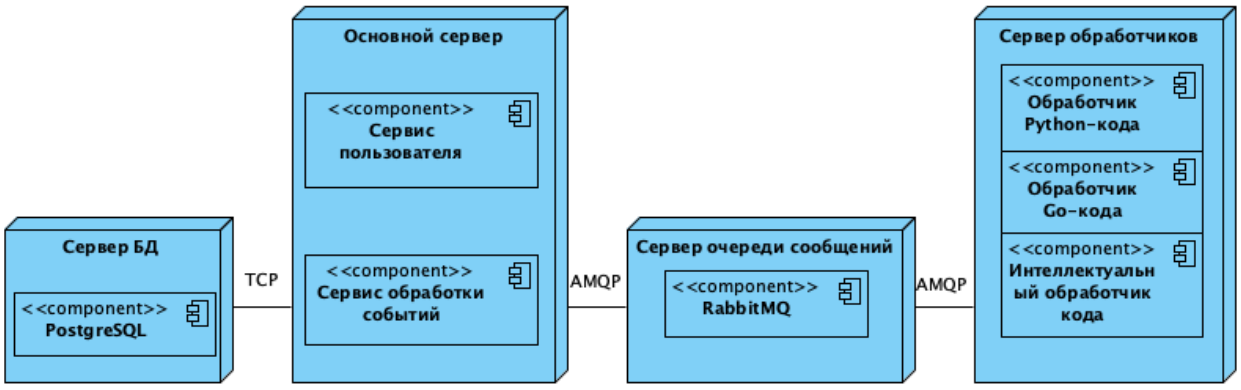

Рисунок 1. Диаграмма развертывания

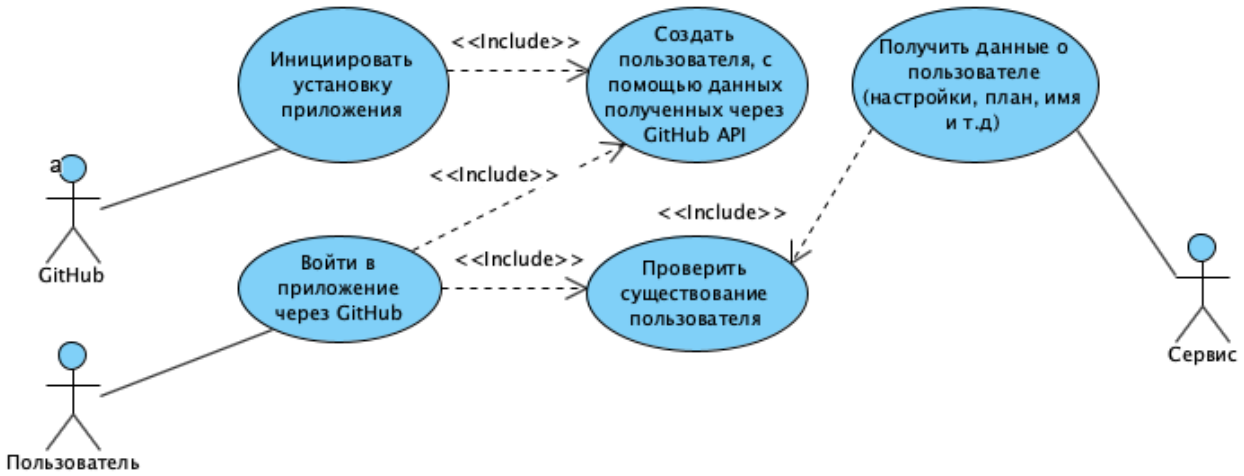

Рисунок 2. Use-Case диаграмма сервиса пользователя

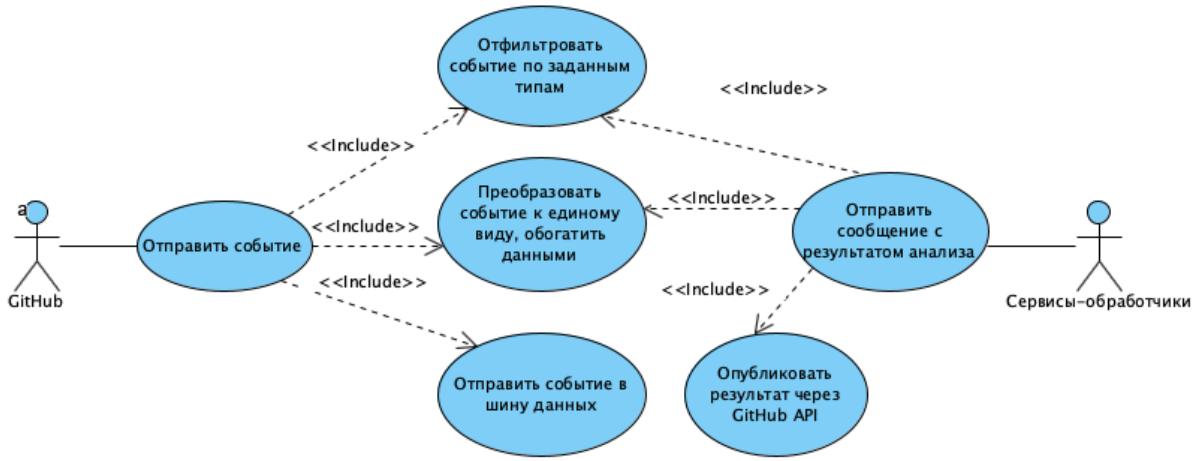

Рисунок 3. Use-Case диаграмма сервиса обработки событий

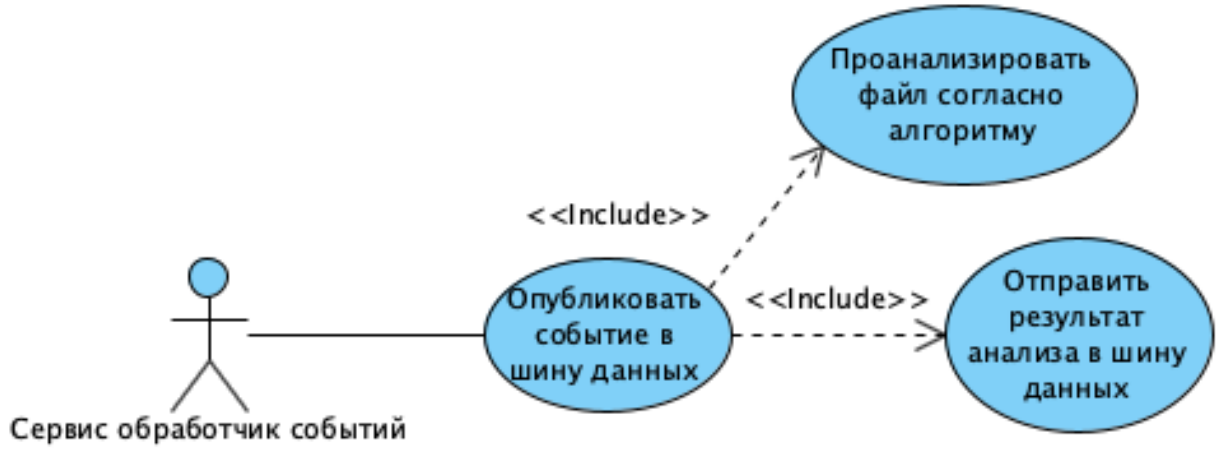

Рисунок 4. Use-case диаграмма сервиса-обработчика кода

2. Архитектура хранилища данных

В качестве основного хранилища данных будет использоваться реляционная СУБД PostgreSQL 13.2

Перечень таблиц:
identity.user:

1. id bigserial primary key
2. username varchar(255) unique not null
3. email varchar(255) unique not null
4. refresh_token varchar(255)
5. created_at timestamptz not null default now()
6. last_login_at timestamptz not null default now()

identity.role:

1. id serial primary key
2. code varchar(64) unique not null
3. title varchar(255) text not null default ‘’

identity.user_role

1. id bigserial primary key
2. user_id bigint not null references identity.user (id)
3. role_id bigint not null references identity.role (id)

settings.group:

1. id serial primary key
2. code varchar(64) unique not null
3. title varchar(255) not null default ‘’

settings.item

1. id serial primary key
2. code varchar(64) unique not null
3. title varchar(255) not null default ‘’
4. help_text varchar(512) not null default ‘’
5. form jsonb not null default ‘{}’::jsonb
6. group_id int not null references settings.group (id)

settings.item_role

1. id bigserial primary key
2. item_id int not null references settings.item (id)
3. role_id int not null references identity.role (id)

settings.value

1. id bigserial primary key
2. user_id bigint not null references identity.user (id)
3. item_id bigint not null references settings.item (id)
4. value jsonb not null default ‘{}’::jsonb

check.execution

1. id bigserial primary key
2. repo varchar(255) not null
3. pull_request int not null
4. head_sha varchar(255) not null
5. base_sha varchar(255) not null
6. executed_at timestamptz not null default now()

check.execution_feedback

1. id bigserial primary key
2. user_id bigint not null references identity.user (id)
3. execution_id bigint not null references check.execution (id)
4. rating smallint not null check(rating > 0 and rating < 11)
5. comment varchar(2048) not null default ‘’

3. Выбор технологий

Основной язык разработки – Go. Данный выбор объясняется следующими качествами данного языка:

- лаконичность и простота – в языке насчитывается всего 29 ключевых слов, его спецификация в разы короче спецификаций других языков. Меньший порог вхождение по сравнению с другими ЯП.
- возраст – языку всего 11 лет и за это время он не успел обрасти множеством устаревших конструкций.
- поддержка параллельного программирования: в отличие от других языков Go изначально разрабатывался с учетом требований к параллельному программированию, в языке используется свой планировщик и свои потоки (user-space)
- количество библиотек: несмотря на свой небольшой возраст, для Go существует множество библиотек. По количеству репозитороиев на GitHub с количеством звезд > 1000 (зачастую таким количеством звезд могут похвастаться лишь библиотеки) язык занимает 4 место

- Открытый исходный код и отсутствие проблем с лицензией
- Компиляция – Go является компилируемым языком, что значительно облегчает процесс деплоя приложения
- Строгая статическая типизация (отсутствие неявных преобразований как в C++, JS и т.д.)
- Проверка индустрией: Go используют такие компании, как: Google, Discord, Twitch, Amazon, Cloudflare, Uber, Intel, Twitter, Heroku, GitHub, Badoo, SoundCloud. На нем написаны следующие инструменты: Docker, Kubernetes, Istio, CockroachDB.

В качестве основного хранилища данных был выбран PostgreSQL благодаря следующим характеристикам:

- Поддержка JSON типа данных, что позволяет использовать PostgreSQL в качестве NoSQL БД.
- PostgreSQL соответствуют требованиям ACID (в отличие от MySQL)
- PostgreSQL имеет открытый исходный код и не имеет проблем с лицензией (в отличие от MySQL)
- Соответствует ANSI-стандартам SQL

RabbitMQ был выбран в качестве шины сообщений из-за данных качеств:

- Наличие удобного admin-интерфейса, где отображаются необходимые метрики
- Работает через стандартизированный протокол AMQP
- Поддерживает приоритет сообщений (В отличие от Apache Kafka)
- Обеспечивает персистентность данных
- Открытый исходный код

Для In Memory кэша был выбран Redis, так как он в отличие от Memcached:

- Предоставляет больше типов данных
- Обеспечивает персистентность
- Поддерживает больший размер ключей и значений
- Поддерживает кластеризацию

4. Релизы

Для облегчения процесса деплоя все наши сервисы должны соответствовать модели 12-факторного приложения (https://12factor.net/).

Будет использоваться Canary Release модель для получение возможности отладки нового релиза на небольшом количестве пользователей.

Также не исключается и зеркалирование трафика перед релизами.

5. Скрипт проведения собеседования

Требования к кандидату:

- От 1 года опыта коммерческой разработки
- Знание языка Go (или активное желание изучить, если опты разработки > 3 лет)
- Знание SQL
- Базовые знания ОС и архитектуры ПК (отличие процесса от треда, User-Space, Kernel Space, инвалидация кэша)
- Базовые знания алгоритмов (умение оценивать сложность алгоритма, знание устройства хэш-таблицы и способов борьбы с коллизиями, знание основных структур данных: связанный список, куча, бинарное дерево, стек и т.д.)
- Базовые знания UNIX командной строки
- Знания об устройстве WEB

Будет плюсом:

- Знание устройства Go (как устроен планировщик, garbage collector, преимущества горутин перед обычными потоками, как реализован канал, слайс, мапа и тд)
- Навыки проектирования систем
- Python

Скрипт:

1. Разговор о предыдущих проектах (вопрос о самой сложной задаче, о самой интересной)
2. Вопросы по Go:
    a. Преимущества языка
    b. Недостатки языка
    c. Отличие строгой типизации от статической
    d. Перечислить основные типы данных
    e. Отличие слайса от массива
    f. Когда нужно использовать указатели, а когда не стоит
    g. Есть ли наследование в Go
    h. Особенности интерфейсов в Go
    i. Что такое горутина
    j. Что такое канал
    k. Отличие буферизированного от небуферезированного канала
    l. Оператор select
    m. Для чего нужен context
    n. Что такое graceful shutdown
    o. Структура приложений в Go
    p. Задача на проверку базовых знаний: подсчет частот слов в предложении
    q. Пустой интерфейс, что такое, когда использовать
    r. Рефлексия в Go
    s. Кодогенерация в Go

3. Вопросы по SQL:
    a. Оператор JOIN, разновидности
    b. Подзапрос, скоррелированный подзапрос
    c. Агрегатные функции (что такое, примеры)
    d. Как отфильтровать запрос с GROUP BY по значению агрегатной функции (для чего нужен оператор HAVING)
    e. Оконные функции (что такое, примеры, PARTITION BY)
    f. Что такое CTE (когда использовать)
    g. Уровни транзакций
    h. Что такое индексы, какие индексы бывают в PostgreSQL их особенности
    i. Null в SQL
    j. Отличие EXPLAIN от EXPLAIN ANALYZE
    k. Что такое партиционирование
    l. ACID
    m. Нормальные формы
    n. One-to-Many, One-to-one, Many-to-Many (как реализовать)
    o. LATERAL JOIN
4. Вопросы по WEB:
    a. Что происходит при вводе google.com в строке браузера
    b. Что такое DNS
    c. Протоколы транспортного уровня
    d. Чем отличается TCP от UDP
    e. Структура запроса и ответа HTTP
    f. Особенности HTTP 2
    g. Что такое API
    h. Принципы REST API
    i. Что такое WebSocket, когда использовать
    j. Что такое CORS
    k. CSRF-атака
    l. Man-In-Middle атака
    m. Отличие Аутентификации от Авторизации
    n. Варианты реализации аутентификации
    o. Особенности session-based аутентификации
    p. Что такое JWT (преимущества)
    q. Ассиметричное и симметричное шифрование
    r. OAuth что такое, виды Flow
    s. OpenID Connect (что такое, чем отличается от OAuth)
5. Вопросы по проектированию систем:
    a. Для чего нужны шины данных
    b. Преимущества NoSQL над SQL
    c. Что такое шардирование
    d. Горизонтальное и вертикальное масштабирование
    e. Stateful vs Stateless
    f. 12 factor app

g. Для чего нужны In Memory (Redis)
h. Репликация БД (для чего, схемы)
i. Задание на проектирование: спроектировать сервис по сокращению ссылок