Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа программной инженерии

Курсовая работа

по дисциплине «Конструирование программного обеспечения»

Выполнил \_\_\_\_ Изображение выглядит как зарисовка, Штриховая графика, рисунок, штриховой рисунок

Автоматически созданное описание\_\_\_\_ студент гр. 5130904/20101

Савитченко Л.А.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Иванов А.С.

Санкт-Петербург

2025Оглавление

[Описание проекта 3](#_Toc198987772)

[1. Определение проблемы 3](#_Toc198987773)

[2. Выработка требований 3](#_Toc198987774)

[3. Разработка архитектуры и детальное проектирование 4](#_Toc198987775)

[4. Кодирование и отладка 7](#_Toc198987776)

[5. Unit-тестирование 7](#_Toc198987777)

[6. Интеграционное тестирование 7](#_Toc198987778)

[7. Сборка 8](#_Toc198987779)

Описание проекта

Приложение для оценки кандидата при приёме на работу. Участники проекта:

* Савитченко Леонид 5130904/20101
* Халиуллин Айдар 5130904/20101
* Фомин Никита 5130904/20101

Ссылка на репозиторий проекта: <https://github.com/Darik-Aidarik/AISupport>

1. Определение проблемы

При приёме на работу HR-специалисты в основном обращают внимание только на резюме кандидата, в которое можно вписать сколько угодно навыков и качеств, однако соцсети могут рассказать о человеке совершенно иную информацию. К сожалению, анализ соцсетей занимает сильно больше времени, чем стандартное прочтение резюме, что приводит к понижению эффективности HR, а потому зачастую попросту игнорируется. Это приводит к найму низкоквалифицированных и ненадёжных специалистов.

2. Выработка требований

Пользовательские сценарии:

Регистрация и авторизация:

Пользователь регистрируется в системе с помощью VK ID.

Выбор кандидата:

Пользователь выбирает из списка название профессии, а также при необходимости меняет тип, после чего вставляет ссылку на профиль в социальной сети VK.

Оценка кандидата:

Пользователь может просмотреть как общую информацию, так и отдельно положительные/отрицательные качества, результаты тестов Люшера и Герчикова, а также рекомендции ИИ.

Сохранение истории запросов:

Пользователь может просматривать историю своих запросов, чтобы быстро получить информацию по ранее интересовавшим кандидатам.

Оценка числа пользователей:

Ожидаемое количество пользователей: 10 000 в сутки.

Период хранения информации: 5 лет.

3. Разработка архитектуры и детальное проектирование

**Характер нагрузки на сервис:**

* Соотношение R/W нагрузки: 75% чтения (запросы информации с соцсети), 25% записи (регистрация, история запросов).
* Объемы трафика: ~1 000 000 запросов в месяц.
* Объемы дисковой системы: ~50 ГБ для хранения истории запросов и данных пользователей.

**Диаграммы C4 Model:**

**Контекстная диаграмма:**

Взаимодействие пользователя с приложением, внешним API VK и базой данных.

A diagram of a diagram

Description automatically generated

**Диаграмма контейнеров:**

Логическая часть, база данных PostgreSQL, внешний API погоды.

A black background with white rectangles and black text

Description automatically generated

**Схема базы данных и обоснование**

**Описание таблиц:**

scan\_history (содержит данные пользователей):

*id — уникальный идентификатор(первичный ключ).*

*lastName — фамилия кандидата.*

*firstName — имя кандидата.*

*birthDate — дата рождения кандидата.*

*criteriaCommun — критерий общительности.*

*criteriaLiter — критерий грамотности.*

*criteriaActivity — критерий активности.*

*criteriaConcen — критерий вовлечённости.*

*criteriaRedFlag — степень дивиации.*

*result — результат.*

*link — ссылка на страницу кандидата в соцсети.*

**Обоснование выбранной схемы:**

**Нормализация:**

* Данные разделены по разным столбцам для удобства поиска нужного параметра.
* Данные о кандидатах сохраняются после завершения программы.

**Производительность:**

* Индексы на первичные ключи (id) ускоряют поиск.

**Масштабируемость:**

При росте нагрузки (>10k пользователей):

* Репликация базы данных (PostgreSQL) для распределения запросов на чтение.
* Шардинг таблицы scan\_history по id для равномерного распределения нагрузки.

Схема масштабирования:

* Горизонтальное масштабирование серверной части с использованием балансировщика нагрузки.
* Репликация базы данных для распределения нагрузки.

4. Кодирование и отладка

* Реализована регистрация и авторизация пользователей.
* Интеграция с внешним API погоды (vk-api).
* Сохранение и отображение истории.
* Каждый участник команды внес коммиты в репозиторий.

5. Unit-тестирование

* Написаны тесты для проверки корректности работы API.
* Пример теста: проверка получения информации о кандидате из его VK-профиля.

6. Интеграционное тестирование

Реализован сценарий:

1. Пользователь авторизуется.
2. Запрашивает информацию для кандидата https://vk.com/nekoko.
3. Получает результат – характеристику как с общей информацией, так и с отдельно вынесенными положительными/отрицательными сторонами, тестами люшера и герчикова, а также рекомендацией от ИИ.

4. Проверяет, что запрос сохранён.

7. Сборка

**Инструкция для запуска:**

**Сборка проекта:**

Выполните команду:

docker-compose build

**Запуск Unit тестов:**

Для запуска unit-тестов выполните:

*docker-compose run --rm app pytest tests/unit -v*

Где app — название сервиса в docker-compose.yml, а tests/unit — путь к unit -тестам.

**Запуск интеграционных тестов:**

Для запуска интеграционных тестов выполните:

*docker-compose run --rm app pytest tests/integration -v*

Интеграционные тесты проверяют взаимодействие компонентов (например, API и базы данных).

**Запуск приложения:**

Для запуска приложения выполните:

*docker-compose up*