Job Search Engine (JSE) Project Documentation

# 1. Введение

Project: Job Search Engine (JSE)

Контракторы: Ospan Ramazan, Saparbek Ulpan, Abdigali Nurasyl

Цель:

JSE - это онлайн-платформа, предназначенная для упрощения взаимодействия между работодателями и соискателями. Она предоставляет обширный набор инструментов для поиска работы и подбора персонала, обеспечивая беспроблемный и эффективный процесс.

# 2. Функциональные Требования

1. Аутентификация пользователей:

• Безопасный механизм аутентификации.

• Регистрация пользователей как соискателей или работодателей.

• Обязательная верификация аккаунта через подтверждение по электронной почте.

2. Поиск и размещение вакансий:

• Соискатели могут искать вакансии по различным критериям (должность, местоположение, зарплата).

• Работодатели могут размещать вакансии с подробным описанием и требованиями.

3. Подача и обзор заявок:

• Соискатели могут подавать заявки на вакансии через платформу.

• Работодатели получают уведомления о новых заявках и могут просматривать профили соискателей.

4. Средства коммуникации:

• Платформа обеспечивает обмен сообщениями между работодателями и соискателями.

• Поддержка планирования интервью.

5. Логирование и мониторинг:

• Система логирует все действия (заявки, вакансии, коммуникации).

• Администраторы могут удаленно просматривать логи для мониторинга системы.

6. Обработка ошибок:

• Платформа должна отображать понятные объяснения при возникновении ошибок.

• Пользователи должны иметь возможность повторного действия после неудачной транзакции.

# 3. Use Case Diagram

Акторы:

1. Соискатель: Размещение заявлений, помощь и обратная связь, просмотр профилей компаний и вакансий, обмен сообщениями, авторизация и регистрация.

2. Работодатель: Размещение вакансий, обработка заявок, просмотр профилей соискателей, обмен сообщениями, авторизация и регистрация.

3. Администратор платформы: Просмотр данных пользователей, проверка подлинности, управление пользователями, обмен сообщениями.

4. Служба поддержки: Обработка обращений, ведение истории обращений.

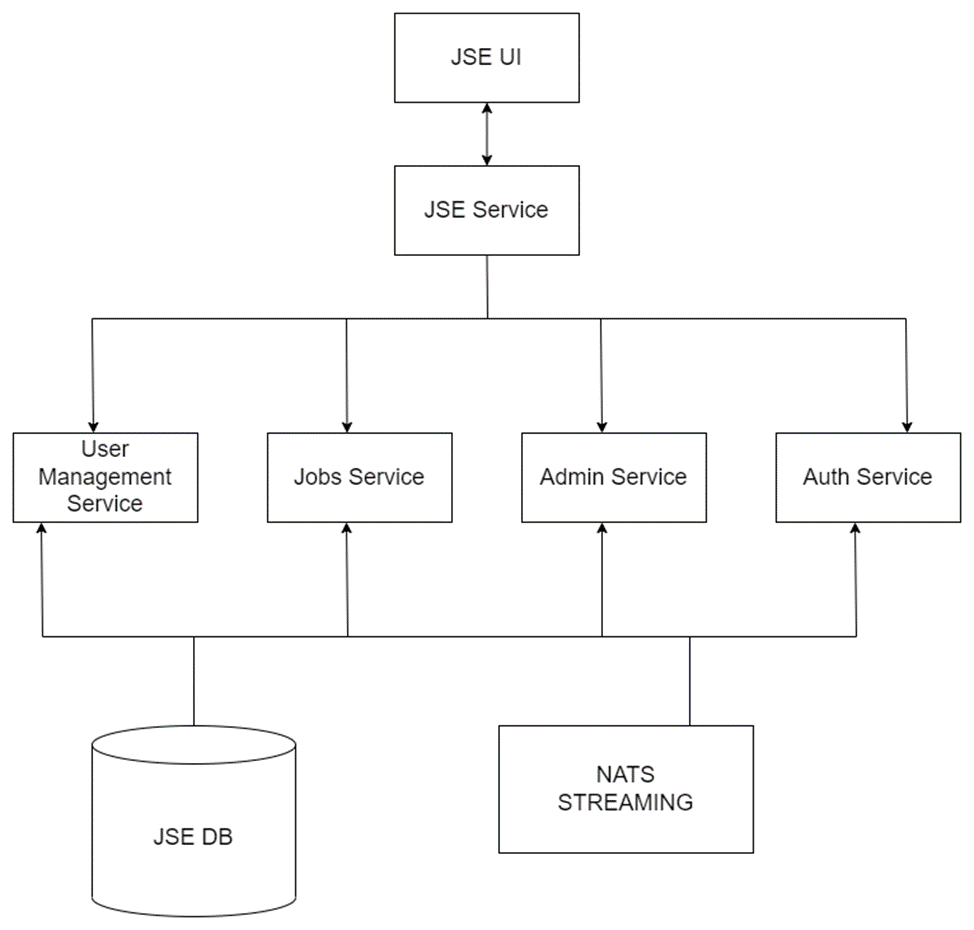
5. Администратор базы данных: Управление структурой данных и производительностью, создание резервных копий.

6. Бот: Ведение статистики, предоставление помощи и советов, оповещение о новых вакансиях.

7. База данных: Автоматическое сохранение данных пользователей и информации о запросах.

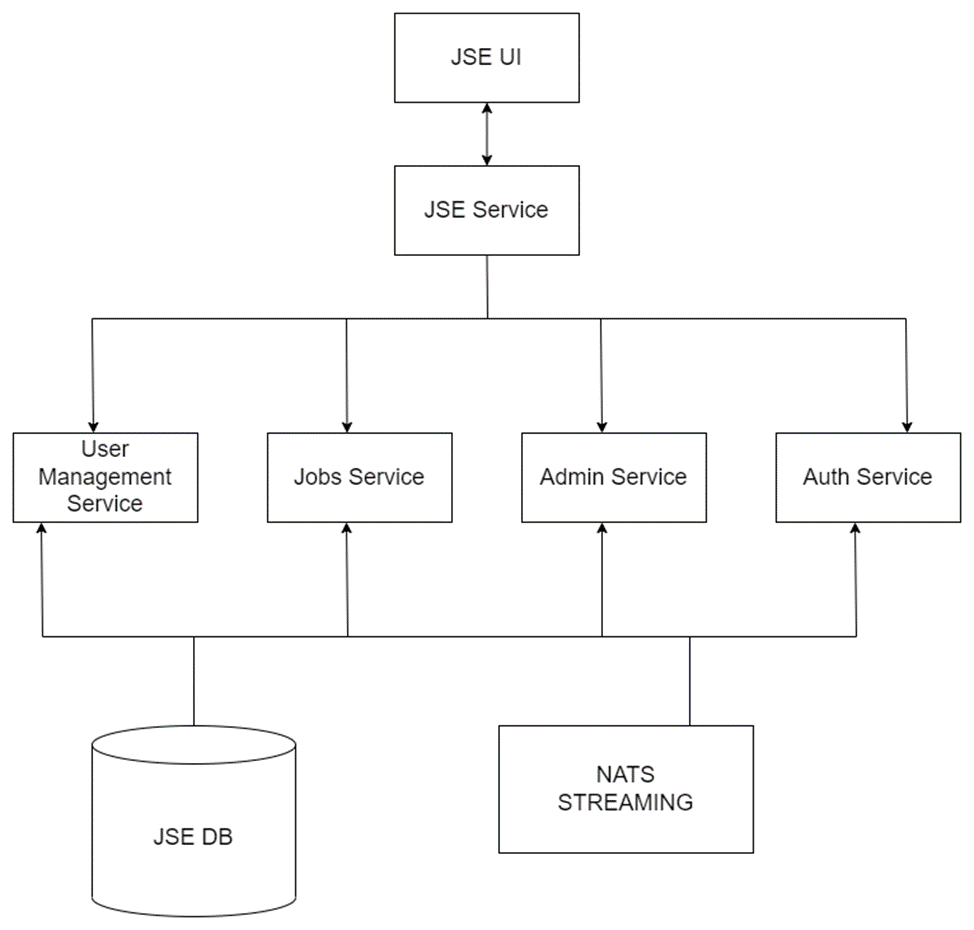
# 4. Activity Diagram

Диаграмма активности:



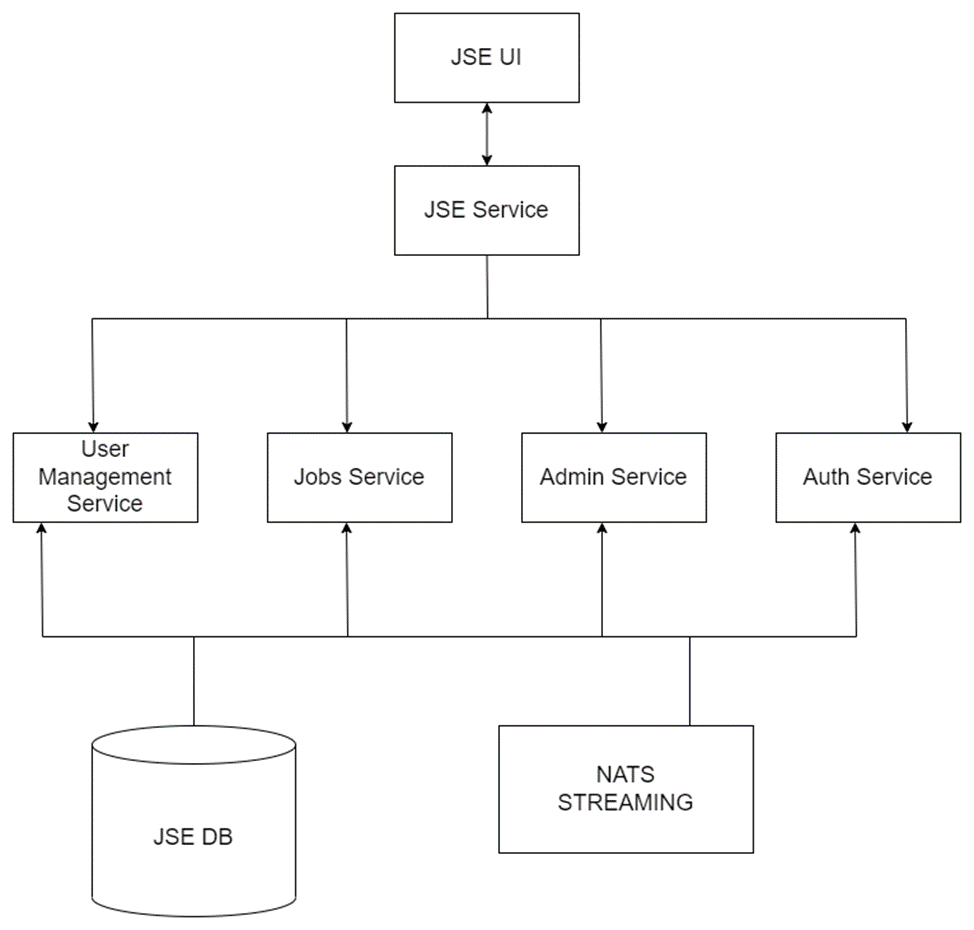
# 5. Sequence Diagram

Диаграмма последовательности:



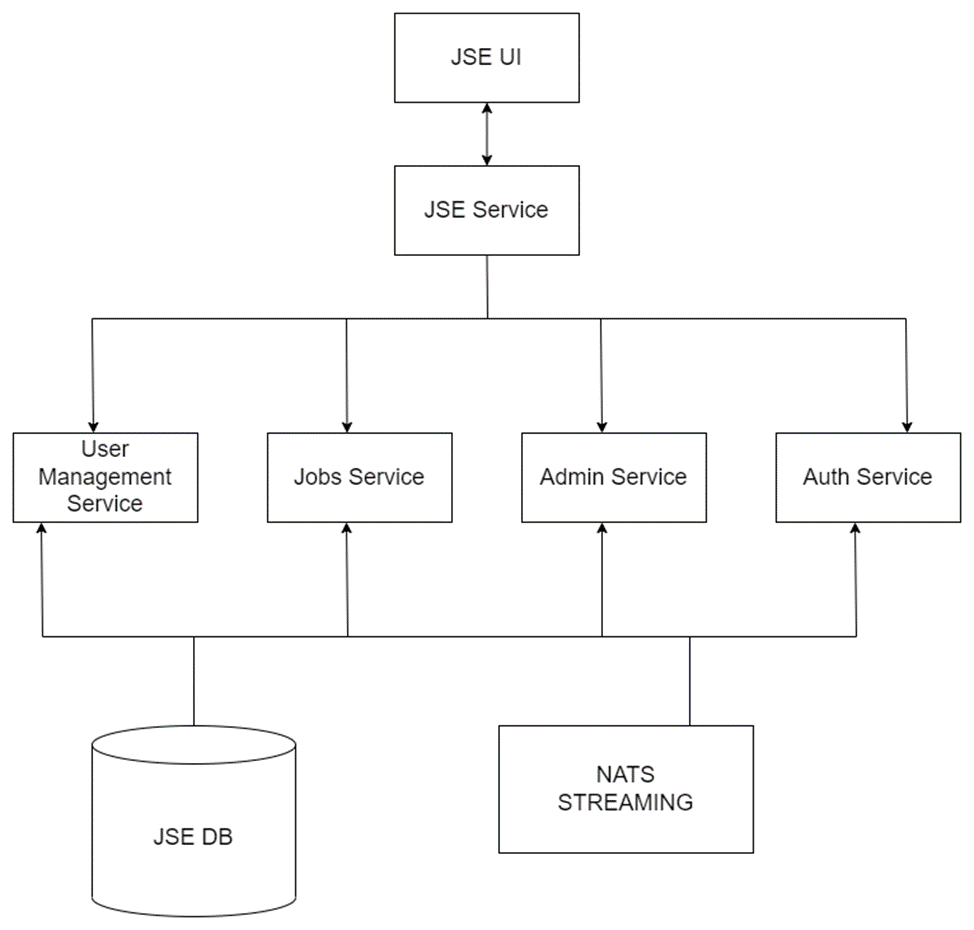
# 6. Class Diagram

Диаграмма классов:



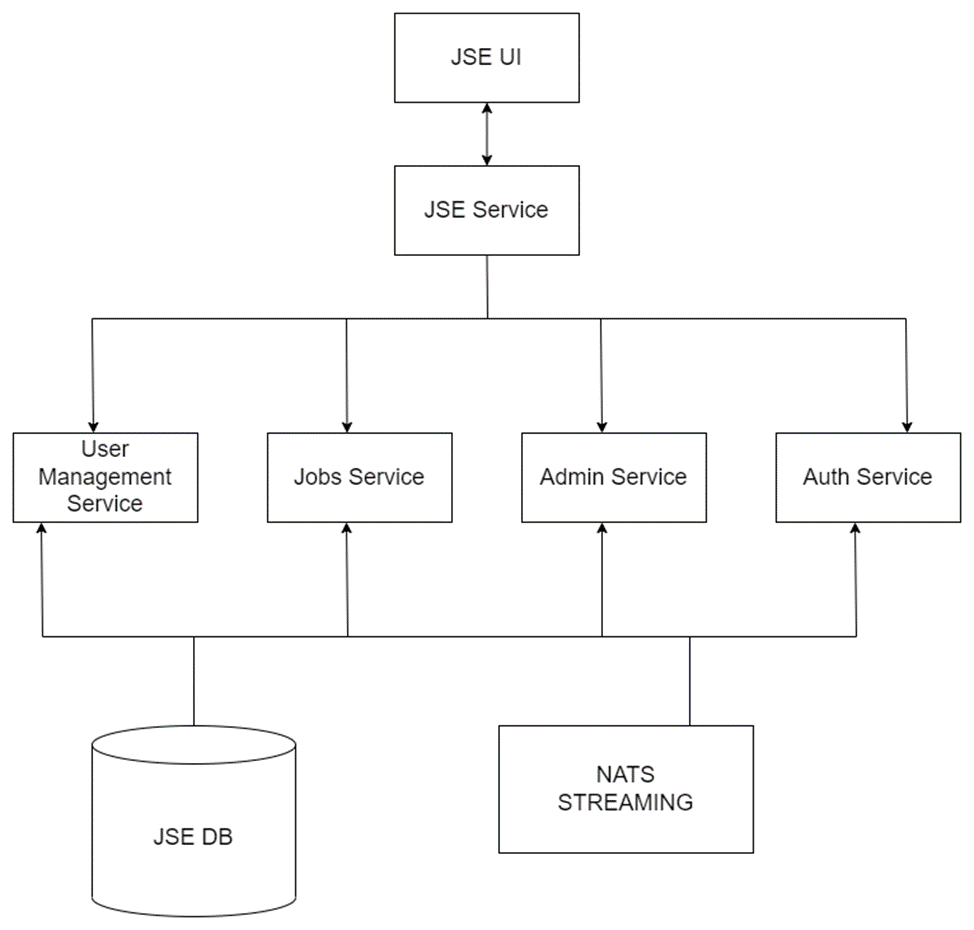
# 7. Component Diagram

Диаграмма компонентов:



# 8. Deployment Diagram

Диаграмма развертывания:



# 9. Design Patterns

1. Factory Method:

• Проблема: Создание разных типов объектов без жесткой привязки к конкретным классам.

• Решение: Использование общего фабричного метода для создания объектов.

• Преимущества: Упрощение добавления новых типов пользователей.

• Недостатки: Сложность поддержки кода.

2. Abstract Factory:

• Проблема: Создание семейств связанных объектов.

• Решение: Создание фабрики для создания семейств связанных объектов.

• Преимущества: Обеспечение согласованности и унификации.

• Недостатки: Увеличение сложности при добавлении новых семейств.

3. Adapter:

• Проблема: Несовместимость интерфейсов модулей.

• Решение: Создание промежуточного класса для согласования интерфейсов.

• Преимущества: Упрощение интеграции.

• Недостатки: Усложнение структуры проекта.

4. Decorator:

• Проблема: Добавление новых функций к существующим классам.

• Решение: Оборачивание базового класса для добавления новых функций.

• Преимущества: Динамичное расширение.

• Недостатки: Усложнение структуры кода.

5. Observer:

• Проблема: Уведомление об изменениях в системе.

• Решение: Создание списка наблюдателей и уведомление их об изменениях.

• Преимущества: Уменьшение жестких зависимостей.

• Недостатки: Увеличение сложности структуры.

6. State:

• Проблема: Изменение поведения в зависимости от состояния.

• Решение: Изменение поведения системы в зависимости от текущего состояния.

• Преимущества: Управляемые переходы между состояниями.

• Недостатки: Усложнение кода.

# 10. Architectural Style

Микросервисный стиль:

• Модульность: Каждый сервис отвечает за определенную часть функциональности.

• Масштабируемость: Легкость масштабирования системы.

• Гибкость: Асинхронное взаимодействие через механизмы, такие как NATS Streaming.

# 11. Заключение

Применение микросервисного стиля к проекту JSE обеспечивает модульность, гибкость и масштабируемость. Сервисы могут разрабатываться и развертываться независимо друг от друга, а взаимодействие осуществляется через асинхронные механизмы, такие как NATS Streaming и REST API. База данных JSE DB предоставляет надежное хранилище данных.