**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**на разработку распределенной системы обработки информации**

**«Автоматизированная система контроля качества городской среды»**

**По дисциплине РСОИ**

**Москва 2022 г.**

## Техническое задание

Проект должен представлять собой портал для сбора информации о существующих недостатках городской среды как путём ручного ввода реальных пользователей системы через специальную форму, так и автоматизированным сбором в режиме реального время из открытых источников, которые доступны в сети интернет (социальной сети ВКонтакте). Собранные обращения будут проверяться модератором и наиболее качественные обращения будут подтверждаться (модератор будет нажимать на кнопку “проверено” на карточке с обращением, после того как модератор проверит и одобрит обращение, то система автоматически будет формировать PDF документ с содержимым обращения и будет его отправлять на электронную почту представителю администрации, который в свою очередь должен исправлять этот городской недостаток и отписываться в комментариях карточки обращения по происходящим работам и закрывать обращение, если проблема исправлена (представитель будет нажимать на кнопку “закрыть” на карточке с обращением). Портал будет прозрачно и понятно отображать информацию авторизированным и не авторизированным пользователям о всех собранных недостатках как в виде списка, так и на интерактивной карте.

Данная система подразумевает разработку:

* Frontend микросервиса, который будет создан на ReactJS с Bootstrap и SemanticUI;
* Backend Gateway Service микросервис (nodejs или java spring), который будет являться единой точкой входа в систему. Все запросы будут проходить через него, он будет выполнять проверку токена;
* Backend микросервиса (java spring), который будет покрывать большую часть сущностей (комментарии, роль, пользователь, организация, местоположение, вложение);
* Backend микросервиса (java spring) для сущности “обращение” (из-за высоких рисков выйти из строя, под воздействием высокой нагрузки);
* Backend микросервиса (java spring) “почтовый агент”, который будет предоставлять удобный интерфейс для отправки email писем, т.е. из любого микросервиса можно будет делать на него POST запрос с заданными параметрами содержимого письма и он в свою очередь будет отправлять письма;
* Backend микросервиса (nodejs или java spring) для сущности “файл”, который будет являться тоже наиболее высоконагруженным и риск выхода из строя будет достаточно высок, поэтому целесообразно его выделить как отдельный микросервис;
* Экспериментальный Backend микросервис (python) для сбора данных о недостатках городской среды из социальной сети ВКонтакте. Этот микросервис будет через vkapi вызов получать все картинки с геометками в пределах города и с помощью нейронной сети локализовать есть ли там, например разбитый тротуар или нет, а если есть, то этот микросервис будет создавать обращение (с указанием причины, места и изображения), т.е. делать post запрос на микросервис, который отвечает за сущность “обращение”.

Данная система будет включать в себя две базы данных PostgreSQL (отдельно для микросервиса который будет хранить большую часть сущностей и отдельно для того который будет хранить только обращения), а также одну базу данных MinIO для хранения бинарных файлов (изображений).

Каждый микросервис при необходимости может иметь доступ к связанной с ним базе данных, но не должен иметь доступа к базам данных других сервисов. Все запросы между микросервисами не требуют авторизации, так как проверка будет производиться только на gateway микросервисе. Запросы пользователей делятся на две категории: запросы, требующие авторизации пользователя, и запросы, доступные для всех пользователей, даже неавторизованных. Все ошибки будут обрабатываться; в случае недоступности некритичного функционала должна осуществляться деградация функциональности. Все действия на сервисах будут логироваться. Все сервисы будут собираться и разворачиваться через CI/CD.