Описание:

Сервер API (JSON HTTP API)

Средства разработки: Java Framework: Play Framework или Spring boot

База данных: Oracle

Протокол: HTTP, порт 80 Функционал (запросы):

1. Загрузчик.
   * Передаем на сервер файл (картинка аватара JPG).
   * Сохраняем картинку в каталоге на сервере.
   * Ответ сервера - внутренний URI картинки.
2. Добавление нового пользователя.
   * Передаем на сервер персональные данные пользователя (URI картинки, имя пользователя, email и т.д.).
   * Сохраняем информацию в базе данных.
   * Ответ сервера - уникальный ID нового пользователя.
3. Получение информации о пользователе.
   * Передаем на сервер уникальный ID пользователя.
   * Читаем информацию из базы данных.
   * Ответ сервера - персональные данные пользователя (см. выше).
4. Изменение статуса пользователя (Online, Offline).
   * Передаем на сервер уникальный ID пользователя и новый статус (Online, Offline).
   * Изменяем статус пользователя.
   * Ответ сервера - уникальный ID пользователя, новый и предыдущий статус.

Примечание: на сервере выполняется запрос к внешнему API/базе данных. Так как это упрощенное тестовое задание необходимо реализовать "заглушку” с имитацией обращения и задержкой по времени 5-10 сек.

1. Статистика сервера.
   * Передаем параметры на сервер: 1. статус клиентов (Online, Offline или отсутствует), 2. уникальный ID (timestamp) запроса (может отсутствовать)
   * Ответ сервера - список пользователей со статусами и URI картинки, а также уникальный ID (timestamp) запроса.

Примечание: Если в запросе есть параметры, то сервер должен фильтровать по ним свой ответ. Если в запросе есть уникальный ID (timestamp) запроса (полученный ранее), то сервер должен вернуть только пользователей, у которых изменились статусы после (по времени)этого уникального ID (timestamp).

###### Обязательные требования:

- RESTful. - Все данные в формате JSON. - Сервер API должен быть спроектирован с учетом того, что запросы 3 и 5 имеет высший приоритет (по отношению к запросам 1, 2, 4) и должны быть выполнены максимально быстро. - Обработка ошибок.

###### Необязательные требования (желательно):

- Документирование кода. - Архитектура Сервера API должна быть рассчитана на высокую нагрузку и масштабирование. - Тесты.