Тестовое задание.

Java разработчик.

Разработать Spring Boot приложение для получения/хранения данных о поле и его владельце.

Функциональность:

1. **PUT /api/field** – добавление нового поля.
   1.1. На вход поступают данные в формате JSON:
   ```
   {
       "Lat": 23.9948834,
       "Lon": -44.243535,
       "FieldName": "My field number 1",
       "AccountName": "User1",
       "AccountEmail": "user1@mail.com"
   }
   ```
   1.2. Принимаем, что AccountName уникален.
   1.3. На выходе должны получить данные об идентификаторах поля и пользователя в формате JSON:
   ```
   {
       "FieldId": 133312,
       "AccountId": 23424
   }
   ```

2. **POST /api/field/<field_id>** – обновление данных поля.
   2.1. На вход поступают данные в формате JSON:
   ```
   {
       "Lat": 23.9948834,
       "Lon": -44.243535,
       "FieldName": "My field number 1",
       "AccountName": "User1",
       "AccountEmapi": "user1@mail.com"
   }
   ```

3. **DELETE /api/field/<field_id>** - удаление поля.

4. **GET /api/field** – получение данных о всех полях.
   Данные получаем в формате JSON:
   ```
   [
       {
           "FieldId": 133312,
           "Lat": 23.9948834,
           "Lon": -44.243535,
           "FieldName": "My field number 1",
           "AccountName": "User1",
           "AccountEmail": "user1@mail.com"
       }
   ]
   ```

5. **GET /api/field/<field_id>** – получение данных о конкретном поле

   Данные получаем в формате JSON:

   ```
   {
       "FieldId": 133312,
       "Lat": 23.9948834,
       "Lon": -44.243535,
       "FieldName": "My field number 1",
       "AccountName": "User1",
       "AccountEmail": "user1@mail.com"
   }
   ```

6. **GET /api/account** – получение данных обо всех аккаунтах

   Данные получаем в формате JSON:

   ```
   [
       {
           "AccountId": 23424
           "AccountName": "UserId",
           "AccountEmail": "user1@mail.com",
           "Fields": [
               "133312",
               "133313"
           ]
       }
   ]
   ```

7. **GET /api/account/<account_id>** – получение данных о конкретном аккаунте

   Данные получаем в формате JSON:

   ```
   {
       "AccountId": 23424
       "AccountName": "UserId",
       "AccountEmail": "user1@mail.com",
       "Fields": [
           "133312",
           "133313"
       ]
   }
   ```

Ошибки обращения должны возвращаться в виде:

```
{
    "Error": "field.does.not.exist",
    "Code": 5,
    "Description": "Field does not exist"
}
```

8. **GET /** - простое табличное легко читаемое отображение списка полей, учесть что полей может быть очень много.

Требования:

1. Важно показать навыки разработки промышленных приложений и уровень владения технологиями: JAVA, Hibernate, Spring Boot – обязательно, Thymeleaf, Bootstrap.
2. Код должен быть сохранён в системе контроля версий.
3. Качество кода должно соответствовать уровню промышленного, а не тестового приложения:
   3.1. Грамотное наименование классов;
   3.2. Легко читаемый и дружественный к рефакторингу код;
   3.3. Обработка исключений.

Результат:

1. Spring Boot приложение.
2. База данных Postgre.
3. В базе, в процессе работы, не должно оставаться мусора, например аккаунтов, у которых нет полей.