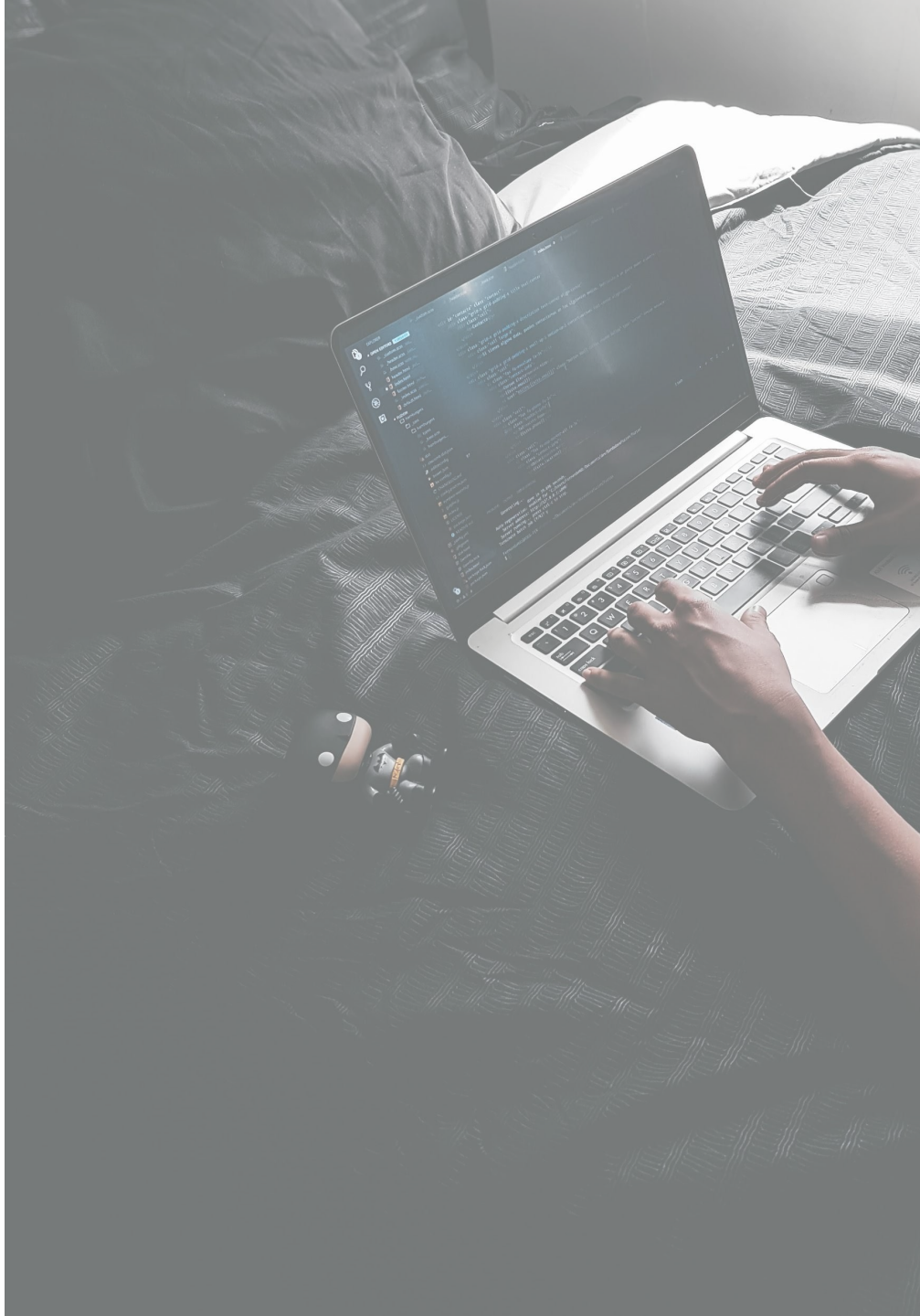
GeekBrains

Автоматизированное тестирование REST API с использованием rest-assured



Факультет тестирования ПО / Тестирование Backend на Java / Урок 3

GeekBrains

В ЭТОМ УРОКЕ

- Учимся писать первые тесты с библиотекой rest-assured
- Вспоминаем основные паттерны проектирования тестов
- Настраиваем логирование и отчётность


Rest-assured

Где скачать?

➔ ссылка на библиотеку в Maven Repository

Зачем использовать?

- Самая популярная библиотека для тестов
- Низкий порог входа
- Покрывает большинство бизнес-кейсов для тестирования
- Написание во fluent-стиле

GeekBrains

Структура теста

```
given()
   .headers(headers)
   .when()
   .get("https://api.spoonacular.com/recipes/716429/information" )
   .then()
   .statusCode(200);
```

Способы логирования

➔ Прописать в пропертях:

```
RestAssured.enableLoggingOfRequestAndResponseIfV
alidationFails();
```

GeekBrains

Способы логирования

➔ Прописать в запросе:

```
given()
   .headers(headers)
   .log()
   .all()
   .when()
   .get(url)
   .prettyPeek()
   .then()
   .statusCode(200);
```

Методы проверок

➔ Библиотека Hamcrest (встроена в rest-assured) после запроса:

```
assertThat(url, equalTo("username"));
```

GeekBrains

Методы проверок

➔ Проверки rest-assured внутри запроса

```
given()
            .expect()
            .body("success", is(true))
            .body("data.url", is("username"))
....
```

Работа с Properties

- Нужен, чтобы убрать хардкод значений
- Можно менять значения в рантайме без перекомпиляции кода
- Хранится в папке resources
- Формат: ключ=значение

Работа с Properties

1. Создать объект Properties:

```
Properties prop = new Properties();
```

2. Загрузить проперти:

```
prop.load(output);
```

3. Прочитать значение свойства по ключу:

```
prop.getProperty("username");
```

GeekBrains

Основные паттерны автотестирования

Избегаем повторений через абстрактный класс

GeekBrains

Основные паттерны автотестирования

«Чистим» созданные данные за собой через метод tearDown

GeekBrains

Основные паттерны автотестирования

Инициализируем переменные и создаём нужные данные в setUp

Цепочки запросов с Shopping list

- Создайте пользователя через Connect User (POST /users/connect)
- Сохраните username и hash пользователя (пароль также можно использовать для входа в https://spoonacular.com/profile)
- Создайте Shopping list
- Добавьте туда item
- Выгрузите содержимое Shopping List
- Удалите item

Ссылка на коллекцию с конкретными запросами (не забудьте подставить свой ApiKey и создать окружение для значений переменных)

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

1. Автоматизируйте GET /recepies/complexSearch (минимум 5 кейсов) и POST /recipes/cuisine (минимум 5 кейсов), используя rest-assured.
2. Сделать автоматизацию цепочки (хотя бы 1 тест со всеми эндпоинтами) для создания и удаления блюда в ShoppingList). Подумайте, как использовать tearDown при тестировании

   POST /mealplanner/:username/shopping-list/items

3. Воспользуйтесь кейсами, которые вы написали в ДЗ №2, перенеся всю логику из постман-коллекции в код.
4. Сдайте ссылку на репозиторий, указав ветку с кодом.

ВАШИ ВОПРОСЫ