Geekbrains

**Автоматизация тестирования комплексного сервиса**

**по покупки туров в Интернет-магазине**

Программа: Разработчик - Тестировщик

Специализация: Тестирование мобильных приложений

Файвисович Мария Александровна

Группа: 4229

Москва

2024 год

Содержание

Введение. Что такое веб-приложение, его особенности

Глава 1. Веб-приложение Интернет магазина по продаже туров

* 1. Бизнес-часть
  2. Техническая часть

Глава 2. Этапы проведения тестирования

2.1 Запуск приложения

2.2 Запуск тестов

2.3 Запуск отдельных тестовых классов

2.4. Перезапуск приложения и отдельных тестов

Глава 3. Формирование отчетности, по результатам автотестов

3.1. Этапы формирование отчета AllureReport по результатам тестирования

Заключение. Разработка предложений по улучшению веб-приложения

Список используемой литературы

Приложения

**Тема проекта:** Автоматизация тестирования веб-приложения на основе веб-приложения Интернет магазина туров.

**Цель:** Проверить поддержку двух СУБД:

MySQL;

PostgreSQL.

Учётные данные и URL для подключения задаются в файле [application.properties](https://github.com/netology-code/qa-diploma/blob/master/application.properties).

**Какую проблему решает:**

**Задача:** автоматизировать позитивные и негативные сценарии покупки тура.

Задача разбита на 4 этапа:

Планирование автоматизации тестирования.

Непосредственно сама автоматизация.

Подготовка отчётных документов по итогам автоматизированного тестирования.

Подготовка отчётных документов по итогам автоматизации

**Введение**

Существует несколько основных подходов для продажи туров в Интернете:

1. Онлайн-турагентства: это веб-сайты, которые предлагают широкий выбор туров от различных туроператоров. Они предоставляют возможность поиска и бронирования туров онлайн, а также предлагают различные способы оплаты.

2. Туристические порталы: это веб-сайты, которые предоставляют информацию о различных туристических направлениях, отелях, авиабилетах и других услугах. Они также могут предлагать возможность бронирования туров и оплаты онлайн.

3. Веб-сайты туроператоров: это веб-сайты, которые принадлежат конкретным туроператорам. Они предлагают туры, которые они организуют, и предоставляют возможность бронирования и оплаты онлайн.

4. Мобильные приложения: это приложения для смартфонов и планшетов, которые позволяют пользователям искать и бронировать туры, а также оплачивать их.

Все эти виды приложений используют различные технологии и подходы к разработке, но все они предназначены для продажи туров и предоставления удобного способа бронирования и оплаты.

Для тестирования веб-приложений по продаже туров обычно используются следующие виды автоматического тестирования:

1. Функциональное тестирование: проверка соответствия функциональности приложения требованиям и ожиданиям пользователей.

2. Тестирование пользовательского интерфейса: проверка корректности отображения элементов интерфейса, их доступности и работоспособности.

3. Тестирование безопасности: проверка защиты данных пользователей и предотвращение возможных уязвимостей.

4. Тестирование производительности: проверка скорости работы приложения и его способности обрабатывать большие объемы данных.

5. Тестирование интеграции: проверка корректной работы приложения с другими системами, такими как платежные сервисы и кредитные сервисы.

6. Тестирование доступности: проверка доступности приложения для пользователей с различными ограничениями, такими как слабовидящие или люди с ограниченными физическими возможностями.

Каждый из этих видов тестирования может быть автоматизирован с помощью специальных инструментов и фреймворков для автоматического тестирования веб-приложений.

В данной дипломной работе выбрано автоматическое тестирование с использованием инструментов Gradle и командной строки и целью данного тестирования будет проверка поддержки двух СУБД и автоматизация позитивных и негативных сценариев покупки тура

**Глава 1. Веб-приложение Интернет магазина по продаже туров**

* 1. **Бизнес-часть**

Веб-приложение Интернет магазина по продаже туров является онлайн-турагентством, туристическим порталом. Оно предлагает возможность поиска и бронирования туров, а также оплаты их двумя способами: обычной оплатой по дебетовой карте и уникальной технологией - выдачей кредита по данным банковской карты. Приложение не обрабатывает данные банковских карт, а пересылает их банковским сервисам. Оно также сохраняет информацию о том, был ли платеж успешным и каким способом он был осуществлен в собственной системе управления базами данных (СУБД). Это веб-сервис, который предлагает купить тур по определённой цене двумя способами:

Обычная оплата по дебетовой карте.

Уникальная технология: выдача кредита по данным банковской карты

Само приложение не обрабатывает данные по картам, а пересылает их банковским сервисам:

сервису платежей, далее Payment Gate;

кредитному сервису, далее Credit Gate.

Приложение в собственной СУБД должно сохранять информацию о том, успешно ли был совершён платёж и каким способом. Данные карт при этом сохранять не допускается.

* 1. **Техническая часть:**

Само приложение расположено в файле [aqa-shop.jar](https://github.com/netology-code/qa-diploma/blob/master/aqa-shop.jar) (ссылка на репозиторий) и запускается стандартным способом java -jar aqa-shop.jar на порту 8080.

В файле [application.properties](https://github.com/netology-code/qa-diploma/blob/master/application.properties) (ссылка на репозиторий) приведён ряд типовых настроек:

учётные данные и URL для подключения к СУБД;

URL-адреса банковских сервисов.

Эмулятор написан на Node.js, для его запуска рекомендуется использовать Docker. Эмулятор расположен в каталоге [gate-simulator](https://github.com/netology-code/qa-diploma/blob/master/gate-simulator) (ссылка на репозиторий). Запускается командой npm start на порту 9999. Он позволяет генерировать предопределённые ответы для заданного набора карт. Набор карт представлен в формате JSON в файле [data.json](https://github.com/netology-code/qa-diploma/blob/master/gate-simulator/data.json) (ссылка на репозиторий).

**Глава 2. Этапы проведения тестирования**

**Рекомендации по запуску:**

**2.1 Запуск приложения**

Перед запуском необходимо выполнить следующие предусловия. Если у вас уже есть необходимое ПО, то понадобится только п.1 и запуск Docker.

2.1.1.Предусловия:

1. Необходимо склонировать репозиторий [git@github.com:Marya373/Diplom\_QA.git](mailto:git@github.com:Marya373/Diplom_QA.git)

или скачать архив по [https://drive.google.com/drive/folders/1EIIDldDva1-KUqFJ5V9QP6PbWN\_WRNdk?usp=drive\_link]. Или воспользоваться VCS Git, встроенной в IntelliJ IDEA.

2. Установать и запустить Docker Desktop. Это можно сделать здесь - https://docs.docker.com/desktop в зависимости от операционной системы.

3. Открыть проект в IntelliJ IDEA

2.1.2. Запустить необходимые базы данных (MySQL и PostgreSQL), а также NodeJS. Параметры для запуска хранятся в файле `docker-compose.yml`. Для запуска необходимо ввести в терминале команду:

```

docker-compose up

```

2.1.3. В новой вкладке терминала ввести следующую команду в зависимости от базы данных

- `java -Dspring.datasource.url=jdbc:mysql://localhost:3306/app -jar artifacts/aqa-shop.jar` - для MySQL

- `java -Dspring.datasource-postgresql.url=jdbc:postgresql://localhost:5432/app -jar artifacts/aqa-shop.jar` - для PostgreSQL

2.1.4. Приложение должно запуститься

**![app run]** ссылка на запуск

работать по адресу <http://localhost:8080/>

**2.2 Запуск тестов**

1. В новой вкладке терминала ввести команду в зависимости от запущенной БД в п.2 Запуска:

- `gradlew clean test -Ddb.url=jdbc:mysql://localhost:3306/app` - для MySQL

- `gradlew clean test -Ddb.url=jdbc:postgresql://localhost:5432/app` - для PostgreSQL

**2.3 Запуск отдельных тестовых классов**

Чтобы не запускать все тесты разом, предусмотрено два варианта запуска отдельных тестовых классов:

Вариант 1

1. В `build.gradle` изменить адрес БД. Для этого нужно заменить строчку `systemProperty 'db.url', System.getProperty('db.url')` на:

- `systemProperty 'db.url', System.getProperty('db.url', 'jdbc:mysql://localhost:3306/app')` - для MySQL

- `systemProperty 'db.url', System.getProperty('db.url', 'jdbc:postgresql://localhost:5432/app')` - для PostgreSQL

2. Запустить приложение (раздел "Запуск", в зависимости от БД)

3. Запустить необходимый тестовый класс командой в терминале: `gradlew clean test --tests PayHappyPathTest` , где PayHappyPathTest - тестовый класс, подлежащий запуску. Или запустить необходимый тестовый класс через IDE с помощью команды Run

Вариант 2

1. В `build.gradle` в раздел test добавить следующее:

```

filter {

includeTestsMatching('\*PayHappyPathTest')

}

```

где PayHappyPathTest - тестовый класс, подлежащий запуску

1. Выполнить раздел "Запуск". Выполнить раздел "Запуск тестов"

**2.4. Перезапуск приложения и отдельных тестов**

Если необходимо перезапустить приложение и/или тесты (например, для другой БД), необходимо выполнить остановку работы в запущенных ранее вкладках терминала, нажав в них Ctrl+С.

**Глава 3. Формирование отчетности, по результатам автотестов**

**3.1. Этапы формирование отчета AllureReport по результатам тестирования**

Шаг 1: В новой вкладке терминала или нажав двойной Ctrl, введите команду "gradlew allureServe".

Шаг 2: После выполнения команды, сгенерированный отчет AllureReport откроется в браузере автоматически.

Шаг 3: На основании данного отчета можно сделать отчет Summery.md (см. документацию)

Шаг 4: Чтобы остановить работу команды, нажмите Ctrl+C или закройте вкладку Run и нажмите Disconnect.

**Документация:**

План автоматизации тестирования веб-формы сервиса покупки туров интернет-банка (ссылка на Readme.md);

Комплексный отчет о проведенной автоматизации тестирования (ссылка на Summary.md)

**Инструменты:**

**Состав команды**: Файвисович М.А.

**Заключение** Разработка предложений по улучшению веб-приложения

**Список используемой литературы**

«Test Automation: A Practical Guide for Testers and Developers" by Mark Fewster and Dorothy Graham». Addison-Wesley Professional. 1999

<https://restapitutorial.ru/>

<https://gb.ru/>

<https://appium.github.io/>

<https://github.com/facebookarchive/WebDriverAgent>

<https://developer.android.com/reference/androidx/test/uiautomator>

**Приложения**