**РОСЖЕЛДОР**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

|  |  |  |  |
| --- | --- | --- | --- |
| **К защите:** |  |  | |
| **Заведующий кафедрой** | **Информационные** | |
| **технологии транспорта** | | |
|  | д-р техн. наук, проф. | |
|  |  | В. И. Хабаров | |
| *подпись* |  | *инициалы, фамилия* | |
|  |  |  | |
| *дата* |  |  | |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тема:** | Тестирование и проверка работоспособности мобильного | | | | | |
|  | приложения «Пассажирам ОАО РЖД» | | | | | |
|  | |  | БР.БИСТ.06.2022 |  |  |
|  | |  | *шифр документа* |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Выполнил** |  |  |  | **Руководитель** |
|  |  | А. Б. Гарбуз |  |  |  | канд. техн. наук, доц.  А. А. Уланов |
| *подпись* |  | *инициалы, фамилия* |  | *подпись* |  | *инициалы, фамилия* |
|  |  |  |  |  |  |  |
| *дата* |  |  |  | *дата* |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Консультанты по разделам** |  |  |  |  |
| 1 |  |  |  |  |
|  |  |  |  |  |
| 2 |  |  |  |  |
|  |  |  |  |  |
| 3 |  |  |  |  |
|  |  |  |  |  |
| Нормоконтролер работы |  |  |  | ст. преп.  Т. А. Распопина |
|  |  | *подпись* |  |  |
|  |  |  |  |  |
|  |  | *дата* |  |  |

**2022 г.**

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

Факультет: Бизнес-информатики

Кафедра: Информационные технологии транспорта

Направление: 09.03.02 «Информационные системы и технологии»

Профиль: Интеллектуальные транспортные системы

|  |  |
| --- | --- |
|  | ***УТВЕРЖДАЮ****: зав. кафедрой «Информационные технологии транспорта»*  д-р техн. наук, проф.  В. И. Хабаров |
|  | *«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2022 г.* |

**З А Д А Н И Е**

**на выполнение выпускной квалификационной работы**

|  |  |  |  |
| --- | --- | --- | --- |
| студенту | Гарбузу Андрею Борисовичу | | |
|  |  | | |
| 1. Тема «Тестирование и проверка работоспособности мобильного приложения Пассажирам ОАО РЖД» утверждена приказом № 203/с от «30» мая 2022 г. | | | |
| 2. Задание выдано «12» мая 2022 г. | | | |
| 3. Срок сдачи законченной работы на кафедру «17» июня 2022 г. | | | |
| 4. Исходные данные: данные, полученные в ходе прохождения преддипломной практики | | | |
| 5. Содержание расчетно-пояснительной записки | | | |
| Наименование разделов и вопросов | | Примерное количество страниц | График (сроки) выполнения |
| Введение | | 1 | 13.05.2022 |
| Аналитическое исследование | | 4 | 13.05.2022 |
| Проектирование программы | | 10 | 15.05.2022 |
| Создание программы | | 14 | 28.05.2022 |
| Заключение | | 1 | 01.06.2022 |

6. Содержание и объемы графической части

|  |  |  |
| --- | --- | --- |
| Наименование графического документа (чертежа, схемы, графика) | Количество  листов  формата А1 | График  (сроки)  выполнения |
| Презентация PowerPoint | 22 | 05.05.2022 |

7. Консультанты по разделам

|  |  |  |
| --- | --- | --- |
| Наименование  раздела | Фамилия, И. О.  консультанта | Подпись консультанта,  дата выдачи задания |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Руководитель |  | А. А. Уланов |
|  | *(подпись, фамилия, И.О.)* |  |
| Задание к использованию принял |  | А. Б. Гарбуз |
|  | *(подпись студента)* |  |

УДК 004.41

**АННОТАЦИЯ**

В работе 47 страниц, 39 рисунков, 4 таблицы, 15 источников.

Ключевые слова: *автоматизированное* *тестирование, мобильное приложение, программа тестирования*.

Предметная область – автоматизированное тестирование. Цель работы – автоматизировать процесс тестирования мобильного приложение.

Во время выполнения работы проведен анализ предметной области, описаны бизнес-процессы, обоснован выбор языка и программного обеспечения, рассмотрено поэтапное проектирование программы.

Результатом выполнения работы является программа, разработанная в среде IntelliJ IDEA, на языке Java, с использованием Appium’а, и эмулятора Android Studio.

**ABSTRACT**

The work contains 47 pages, 39 figures, 4 tables, 15 sources.

Keywords: *automated testing, mobile application, testing program*.

The subject area is automated testing. The purpose of the work is to automate the process of testing a mobile application.

During the execution of the work, the analysis of the subject area was carried out, business processes were described, the choice of language and software was justified, the phased design of the program was considered.

The result of the work is a program developed in the IntelliJ IDEA environment, in Java, using Appium, and the Android Studio emulator.

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

Android – мобильная операционная система компании Google.

ЭБ – Электронный билет.

Лог / файл журнала — это текстовые файлы, в которых хранится информация о прошедших тестированиях, времени начала тестирования и окончания, а также об ошибках.

Система / Приложение / Программа / Робот – программа по тестированию мобильного приложения «Пассажирам ОАО РЖД».

Переменная среды – текстовая переменная операционной системы, хранящая какую-либо информацию, например, данные о настройках системы.

API – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

APK – формат архивных исполняемых файлов-приложений для Android и ряда других операционных систем, основанных на Android.

JSON – это популярный формат текстовых данных, который используется для обмена данными в современных веб - и мобильных приложениях.

Фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Скрипт – это набор команд, то есть строк кода, которые вкупе выполняют конкретную задачу.

Рефакторинг – это переработка кода, равносильное преобразование алгоритмов — процесс изменения внутренней структуры программы.

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc106132735)

[1 Аналитическое исследование 7](#_Toc106132736)

[1.1 Определение предметной области 7](#_Toc106132737)

[1.2 Определение проблемы предметной области 8](#_Toc106132738)

[1.3 Решение проблемы 8](#_Toc106132739)

[1.4 Постановка цели, задачи 9](#_Toc106132740)

[1.5 Вывод об аналитическом исследовании 10](#_Toc106132741)

[2 Проектирование программы тестирования 11](#_Toc106132742)

[2.1 Моделирование бизнес-процессов 11](#_Toc106132743)

[2.2 Обоснование выбора языка программирования 14](#_Toc106132744)

[2.3 Обоснование выбора среды разработки 15](#_Toc106132745)

[2.4 Вывод о проектировании программы 20](#_Toc106132746)

[3 Разработка программы 21](#_Toc106132747)

[3.1 Первоначальные настройки 21](#_Toc106132748)

[3.2 Разработка программы 24](#_Toc106132749)

[3.3 Вывод о разработке программы 34](#_Toc106132750)

[4 Руководство пользователя 35](#_Toc106132751)

[Заключение 45](#_Toc106132752)

[Список использованных источников 46](#_Toc106132753)

**ВВЕДЕНИЕ**

Тестирование – это процесс исследования и испытания программного продукта, который имеет цель проверить соответствие между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Тестирование мобильных приложений предполагает проверку работы программы по различным сценариям использования на большом количестве платформ. Множество переменных, связанных не только с операционной системой устройств пользователей, но и с размерами экранов, емкостью аккумуляторов, количеством SIM-карт и так далее.

Проверка работоспособности программы в разных условиях обязательна. Большинство пользователей не будут вникать в проблемы разработчика. Они просто перестанут пользоваться мобильным приложением.

Практическая значимость работы состоит в возможности нахождения ошибок работы мобильного приложения для их быстрого решения.

Применяемые методы исследования: табличный метод представления и обработки информации, метод сравнения, методы графического представления информации (графики, иллюстрации), методы моделирования.

В качестве источников информации использовались электронные ресурсы.

**1 Аналитическое исследование**

* 1. **Определение предметной области**

Предметной областью выпускной квалификационной работы является автоматизированное тестирование. Обычно после написания кода его проверяют. Если речь идет о функции, то может быть написан скрипт, который будет вызывать её с разными аргументами, и смотреть, что она будет возвращать. Если был сделан сайт или приложение, то его необходимо открывать, нажимать на все кнопки и ссылки, проверять что всё отображается. Такой процесс называется ручное тестирование, когда человек проверяет работу программы. Если автоматизировать этот процесс, и написать программу, которая проверяет правильность работы другой, то это называется автоматизированное тестирование [1].

Автоматизация, может применятся к нескольким подходам к тестированию:

- тестирование производительности (нагрузочное, стрессоустойчивое, тестирование на стабильность);

- регрессионное тестирование (проверка ПО на корректность функциональности);

- конфигурационное тестирование (выполнение тестов в разных условиях);

- функциональное тестирование (проверка работы интерфейса);

- установочное тестирование (проверка работы инсталляции продукта) [2].

В рамках выпускной квалификационной работы будет использоваться функциональных подход. В функциональном тестировании основную работу исполняют библиотеки Selenium, так как именно они позволяют связать скрипт и девайс, на котором происходит тестирование приложения.

**1.2 Определение проблемы предметной области**

Основными проблемами предметной области являются:

- человеческий фактор – человек в отличие от скрипта может легко пропустить какой-то шаг тестирования;

- долгое выполнение – при ручной проверке работоспособности тратится большое количество времени;

- сложный рефакторинг – при наличии ошибки с вашей стороны, или со стороны программиста из команды, сложно определить в какой части кода она произошла;

- необходимость вмешательства – программисту необходимо прерывать работу над каким-либо другим проектом для проведения тестирования;

- формирование отчетов – программисту необходимо вручную формировать отчет о проведенном тестировании.

В свою очередь автоматизация позволит проводить быстрое тестирование приложения. Тесты могут выполнятся в нерабочее время по заложенному сценарию, что исключает вмешательство человека в этот процесс. После выполнения теста программисту будет приходить уже сформированный отчет. В отчете будет указано на каком этапе были обнаружены ошибки.

**1.3 Решение проблемы**

Определив проблему предметной области, перейдем к ее решению, которое видится в разработке программы по тестированию мобильного приложения.

Автоматизированное тестирование — процесс, представляющий собой имитацию интерактивных действий пользователя и проверку результатов этих действий.

Исходя из определения автоматизированного тестирования, программа будет имитировать действия пользователя при покупке электронных билетов и производить проверку этих действий. Также программа будет разделена на классы, что позволит удобно дополнять получившийся код.

Объектом тестирования будет мобильное приложение «Пассажирам ОАО РЖД». Задание тестировать это приложение было выдано Новосибирским ИВЦ – филиалом ГВЦ ОАО «РЖД», как студенту целевого набора обучения.

Программа должна автоматизировать следующие бизнес-процессы:

- проведение оперативного тестирования мобильного приложения по заданному сценарию;

- предоставление информации о проведенном тестировании в виде записей в файле журнала.

Также она должна тестировать стандартный сценарий покупки ЭБ на поезда дальнего сообщения и пригородные маршруты. Примерный алгоритм программы тестирования должен выглядеть следующим образом:

- выбор временной зоны;

- выбор станции отправления/прибытия;

- выбор периода отправления поезда;

- выбор поезда по направлению туда/обратно;

- выбор вагона по направлению туда/обратно;

- авторизация пользователя;

- выбор вагонов и ест по направлениям;

- заполнение данных по пассажиру;

- переход к платежному шлюзу.

1.4 Постановка цели, задачи

Исходя из предложенного решения проблемы, на данном этапе необходимо установить цель и задачи.

Цель создания программы: автоматизация процесса тестирования мобильного приложения «Пассажирам ОАО РЖД».

Для достижения поставленной цели необходимо выполнить следующие задачи:

- описать бизнес-процессы выполнения тестирования мобильного приложения;

- построить концептуальную модель;

- обосновать выбор языка и среды разработки;

- разработать программу тестирования;

- разработать руководство пользователя.

1.5 Вывод об аналитическом исследовании

Анализ предметной области показал, что разрабатываемая программа должна обеспечить: снижение нагрузки на программистов, занимающихся сопровождением мобильного приложения, проведение оперативного тестирования по заданному сценарию, отображение достоверной и содержательной информации в виде записей в файле журнала, а также описание выявленных ошибок.

2 Проектирование программы тестирования

2.1 Моделирование бизнес-процессов

Для моделирования бизнес-процессов выбрана методология UML. Unified Modeling Language (UML) – унифицированный язык моделирования: оно подразумевает создание модели, описывающей объект. Подходит для широкого класса проектируемых программных систем, различных областей приложений, типов организаций, уровней компетентности, размеров проектов. Диаграммы UML описывают границы, структуру и поведение как всей системы, так и отдельных объектов в ее составе [3].

Унифицированный язык включает в себя 12 типов диаграмм, среди которых: четыре типа диаграмм относятся к структурным, пять типов описывают поведенческие особенности системы, и три представляют физическое функционирование системы (диаграммы взаимодействия).

Так, наиболее актуальными являются следующие типы диаграмм UML:

- диаграмма классов (Class diagram);

- диаграмма вариантов использования (Usecase diagram);

- диаграмма последовательности (Sequence diagram);

- диаграмма состояний (State diagram);

- диаграмма деятельности (Activity diagram).

Чаще всего диаграммы UML используются для описания рабочих процессов при разработке программного обеспечения. UML-диаграммы доступно передают информацию всем заинтересованным участникам процесса создания продукта.

Также стоит отметить, что при помощи UML можно легко преобразовать объектно-ориентированную модель в схему реляционной базы данных.

Первым шагом построена диаграмма вариантов использования (Use Case Diagram). Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью, так называемых вариантов использования. Диаграмма вариантов использования процесса «Проведение тестирования» представлена на рисунке 2.1.

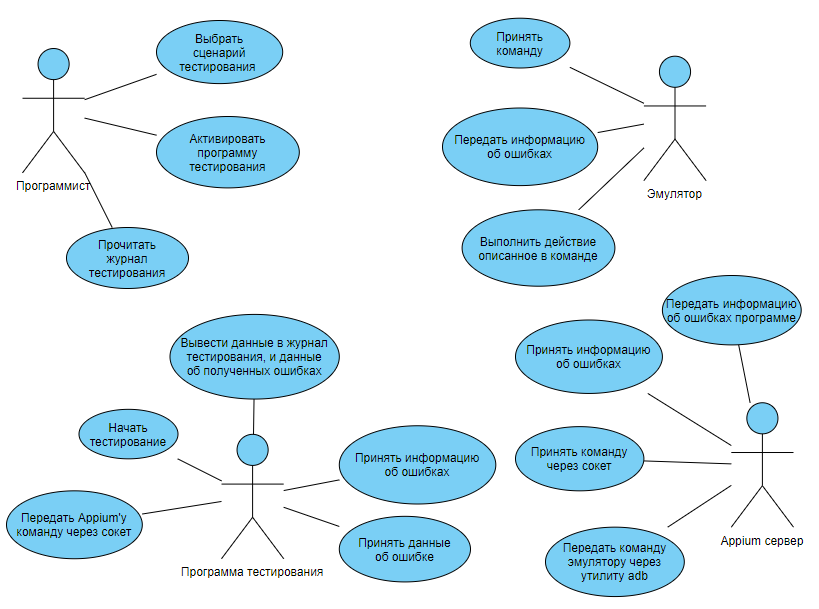


Рисунок 2.1 – Диаграмма вариантов использования процесса «Проведение тестирования»

На диаграмме видно, что в процессе тестирования участвуют четыре актера:

- программист;

- программа тестирования;

- программа для автоматизации тестирования Appium;

- эмулятор.

Вторым шагом построена диаграмма последовательности (Sequence Diagram). Данные диаграммы фокусируются на времени и визуально показывают порядок взаимодействия, используя вертикальную ось диаграммы для представления времени, когда отправляются сообщения. Полученная диаграмма последовательности процесса «Проведение тестирования» представлена на рисунке 2.2.

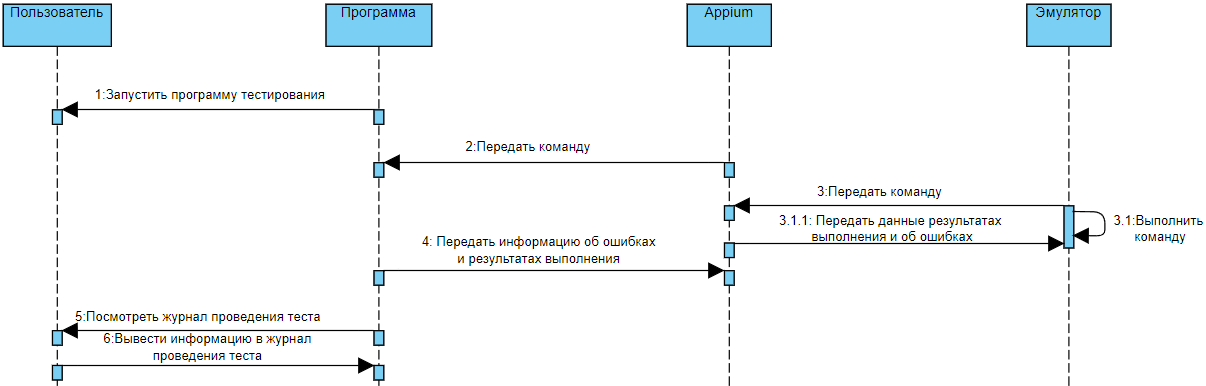


Рисунок 2.2 – Диаграмма последовательности процесса «Проведение тестирования»

Следующим шагом построена диаграмма активности (Activity diagram), которая используется для создания более «продвинутых» блок-схем, в этом случае часто применяются конструкции принятия решения, а также разделения и слияния потоков управления. Полученная диаграмма активностей процесса «Проведение тестирования» представлена на рисунке 2.3.

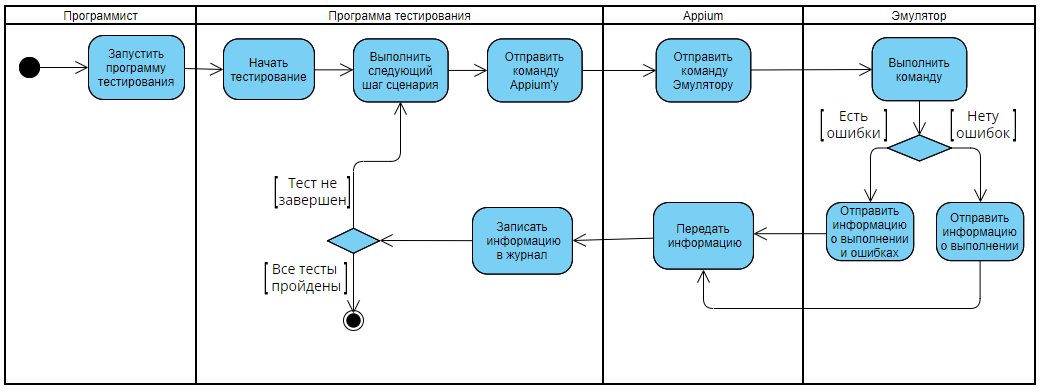


Рисунок 2.3 – Диаграмма активностей процесса «Проведение тестирования»

На диаграммах активностей и последовательности можно увидеть весь процесс тестирования.

**2.2 Обоснование выбора языка программирования**

После моделирования и анализа бизнес-процессов необходимо выбрать наиболее подходящий язык программирования для разработки программы. На сегодняшний день существует большое количество различных языков программирования, каждый из которых имеет свою специфику применения.

Так, для того чтобы выбрать наиболее актуальный язык программирования, была изучена статистика на 2021 год [4]. Статистика была составлена на основе опроса программистов работающих в сфере мобильных разработок. Для наглядности была создана диаграмма, рисунок 2.4.

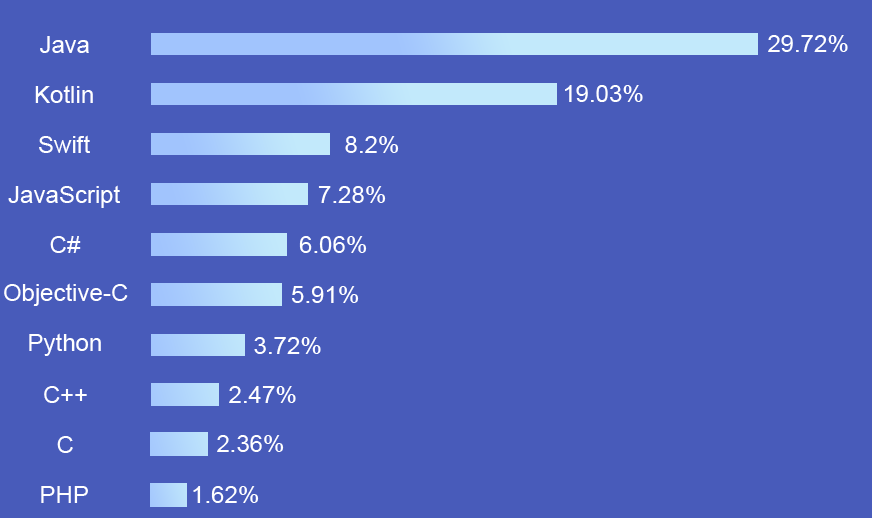


Рисунок 2.4 – Рейтинг языков программирования

Анализируя диаграмму, можно сделать вывод, что большее количество опрошенных программистов рекомендуют язык Java. Поэтому этот язык будет использоваться для создания программы по тестированию мобильного приложения.

Java [это высокоуровневый](https://translated.turbopages.org/proxy_u/en-ru.ru.d4dc3685-62a48dae-68bc971a-74722d776562/https/en.wikipedia.org/wiki/High-level_programming_language) [объектно](https://translated.turbopages.org/proxy_u/en-ru.ru.d4dc3685-62a48dae-68bc971a-74722d776562/https/en.wikipedia.org/wiki/Class-based_programming)-ориентированный [язык программирования на основе классов](https://translated.turbopages.org/proxy_u/en-ru.ru.d4dc3685-62a48dae-68bc971a-74722d776562/https/en.wikipedia.org/wiki/Object-oriented_programming)[,](https://translated.turbopages.org/proxy_u/en-ru.ru.d4dc3685-62a48dae-68bc971a-74722d776562/https/en.wikipedia.org/wiki/Programming_language) предназначенный для того, чтобы иметь как можно меньше [зависимостей](https://translated.turbopages.org/proxy_u/en-ru.ru.d4dc3685-62a48dae-68bc971a-74722d776562/https/en.wikipedia.org/wiki/Dependency_(computer_science)) от реализации [5]. [Скомпилированный](https://translated.turbopages.org/proxy_u/en-ru.ru.d4dc3685-62a48dae-68bc971a-74722d776562/https/en.wikipedia.org/wiki/Compiler) Java-код может работать на всех платформах, поддерживающих этот язык, без необходимости перекомпиляции. Одними из преимуществ языка являются его большое сообщество и обширное количество материалов, что упрощает его изучение [6].

Также Java схож с языком программирования C#, который был изучен в процессе обучения, что упростит освоение языка [7].

**2.3 Обоснование выбора среды разработки**

С учётом выбранного языка программирования, было принято решение разрабатывать приложение на интегрированной среде разработки IntelliJ IDEA, программой для автоматизированного тестирования будет выступать Appium, а в качестве эмулятора Android Studio.

Интегрированная среда разработки, ИСР (англ. IDE, Integrated Development Environment или Integrated Debugging Environment) – система программных средств, используемая программистами для разработки программного обеспечения (ПО) [8].

IntelliJ IDEA (II) – это интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains. Полноценный набор инструментов и функций для улучшения и усовершенствования каждого этапа разработки программного обеспечения [9].

Appium – Инструмент автоматизации с открытым исходным кодом для запуска сценариев и тестирования собственных приложений, мобильных веб-приложений и гибридных приложений на Android или iOS с использованием Selenium webdriver [10].

Android Studio (AS) – быстрый эмулятор смартфонов от Google позволяет запускать на компьютере последние версии операционной системы Android и тестировать необходимые вам приложения. Эмулятор Android Studio предоставляет практически все возможности реального Android-устройства (смартфона или планшета) прямо с компьютера. Android Studio, основанна на программном обеспечении [IntelliJ IDEA](https://ru.wikipedia.org/wiki/IntelliJ_IDEA) от компании [JetBrains](https://ru.wikipedia.org/wiki/JetBrains), официальное средство разработки Android приложений [11].

Для среды разработки будет использоваться Community версия, потому что она бесплатная.

Интерфейс IDE представлен на рисунке 2.5.

Интерфейс Appium’а представлен на рисунке 2.6.

Интерфейс Android Studio представлен на рисунке 2.7.

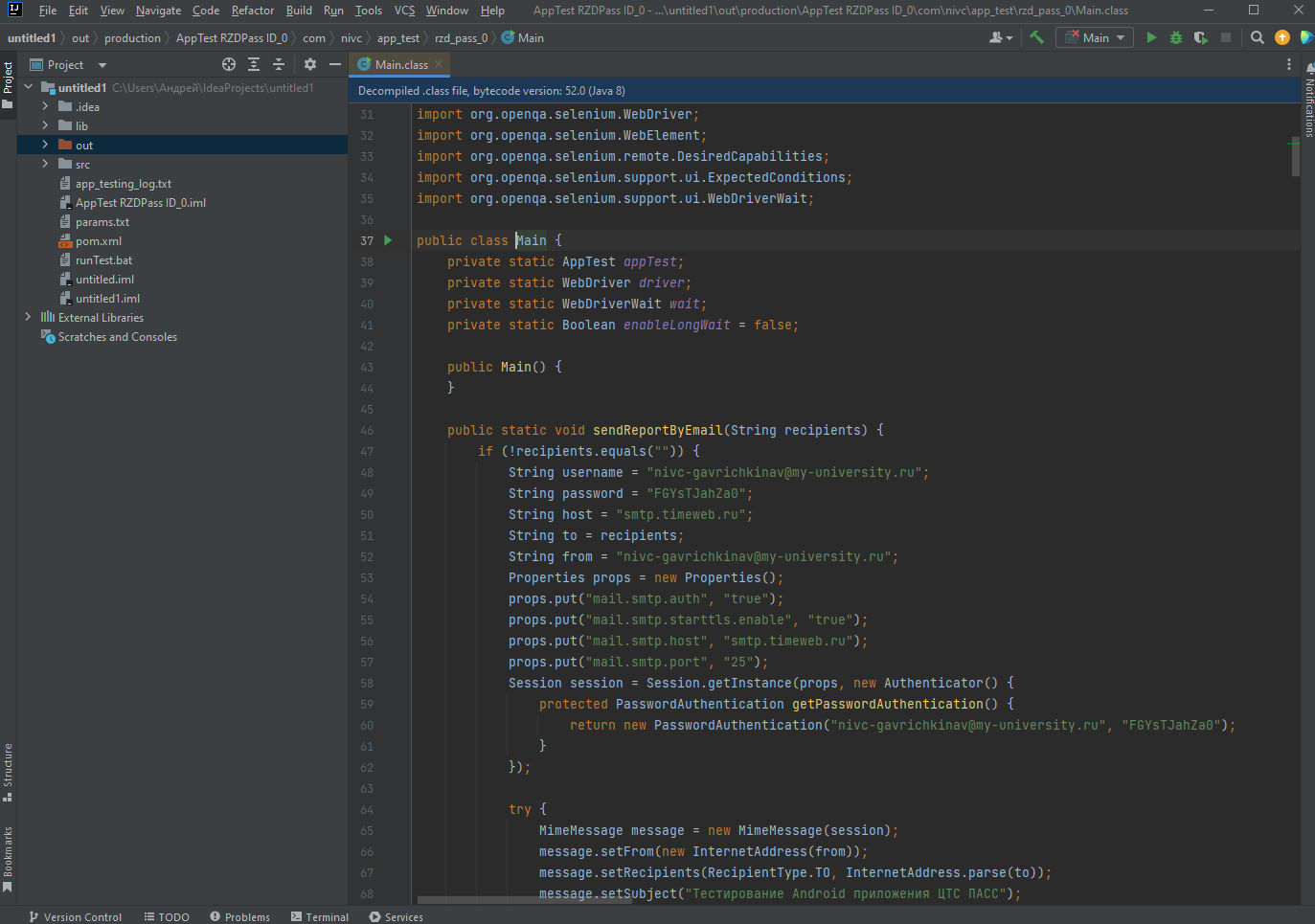


Рисунок 2.5 – Интерфейс IDE Intellij IDEA

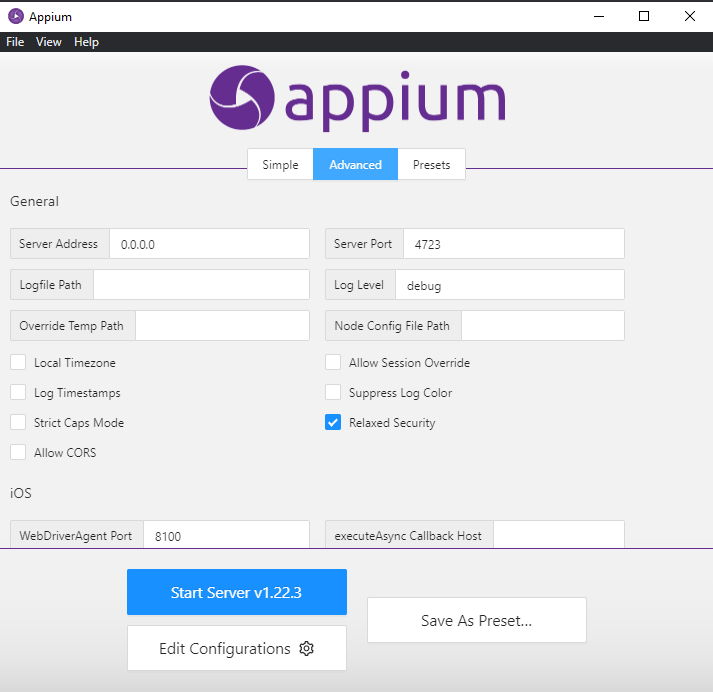


Рисунок 2.6 - Интерфейс Appium

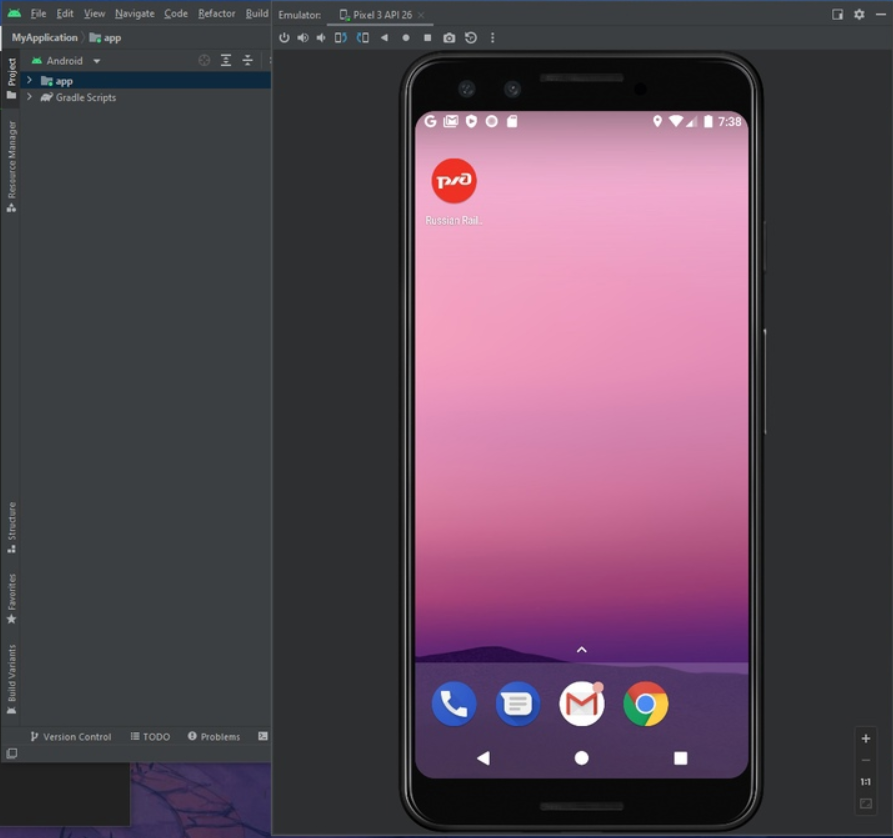


Рисунок 2.7 – Интерфейс Android Studio

Для автоматизированного тестирования мобильных приложений существует несколько универсальных программ, чтобы обосновать выбор Appium’а, приведем таблицу в которой сравним её с аналогами.

Сравнение программы для автоматизированного тестирования с аналогами представлена в таблице 2.1.

Таблица 2.1 – Сравнение программ автоматизированного тестирования

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Приложение | Платформы | Стратегии тестирования | Языки | Открытый ресурс | Вендор | Типы приложений |
| Detox | IOS, Android | grey-box | JavaScript | + | Google | native |
| Ranorex | IOS, Android | black-box, grey-box, white-box | C#, VB.NET | - | Ranorex | native,  web,  hybird |

Продолжение таблицы 2.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TestComplete  Mobile | IOS, Android | black-box,  white-box | JavaScript,  JScript, Python, VBScript | - | SmartBear | native,  web,  hybird |
| Appium | IOS, Android,  Windows | black-box | C#, Java,  Javascript,  Objective-C, PHP,  Python,  Ruby,Swift | + | JS Foundation | native,  web,  hybird |

Изучив составленную таблицу, можно выявить наиболее оптимальное приложение для создания тестов. Так как все приложения являются универсальными для всех платформ этот параметр не учитывается, зато очень сильно влияет доступность приложения.

Ranorex и TextComplete Mobile являются платными приложениями, и поэтому они менее желательны для использования, так как требуют лишних денежных затрат. В это время Appium и Detox находятся в отрытом доступе. Все приложения кроме Detox могут тестировать все типы мобильных приложений, что является значительным минусом для этой программы. Язык тестирование так же имеет вес, так как специалисту, не знающему язык программирования программы, в которой ему предстоит работать означает дополнительные затраты на время обучения.

Можно сделать вывод, что наиболее подходящее приложение для будет Appium. Он предоставляет возможности тестировать все виды приложений на различных платформах. Имеет открытый доступ и большое количество языков программирования, что позволит упростить работу по тестирования приложений. При тестировании происходит работа именно с сходным приложением. Тесты пишутся с помощью API, не прибегая к преобразованию приложения в какой-либо особый, совместный с фреймворком вид [12]. И Appium очень полезен при отсутствии опыта в автоматизации тестирования, что является огромным плюсом, графическое представление работы Appium’а представлено на рисунке 2.11.

В качестве эмулятора был выбран Android Studio, также были приведены аналоги выбранному приложению, таблица 2.2.

Таблица 2.2 – Сравнение программ для эмулятора Android

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Приложение | Платформы | Поддержка Java | Открытый ресурс | Типы приложений |
| Android studio | Android  IOS  Windows | + | + | native,  web,  hybird |
| PhoneGap | IOS  Android  Blackberry | - | \_ | native |
| B4X | Android  IOS  Windows | - | + | web,  hybird |
| RAD Studio | Android  IOS  Windows | + | - | native,  web,  hybird |
| Visual Studio | Android  IOS  Windows | + | + | native,  web,  hybird |

Как можно увидеть из таблицы, Android Studio схож по своим характеристикам с Visual Studio, что заставляет делать выбор между этими двумя программами. Выбор пал на Android Studio, так как в отличие от Visual Studio в нем без установки дополнительных инструментов имеется готовый эмулятор Android и его родной язык Java. В Visual Studio необходимо устанавливать Xamarin в качестве эмулятора, загружать SDK пакеты, и устанавливать расширение для поддержки Java.

Последним необходимо сравнить выбранную среду разработки IntelliJ IDEA с её аналогами. Сравнение IntelliJ IDEA с аналогами приведено в таблице 2.3.

Таблица 2.3 – Сравнение сред разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | NetBeans | IntelliJ Idea | Eclipse | Microsoft Visual Studio |
| Бесплатный, в открытом доступе | + | + | + | + |
| Встроенный компилятор и/или интерпретатор | + | + | + | + |
| Запуск кода из среды | + | + | + | + |
| Встроенная отладка | + | + | - | + |
| Не требуется расширение для  Java | + | + | + | - |
| Расширенный функционал | + | + | + | + |

Проанализировав таблицу было выяснено что NetBeans и IntelliJ IDEA являются хорошими кандидатами на роль среды разработки. Как среда разработки была выбрана IntelliJ IDEA, так как она является базовой IDE для Android Studio, располагает удобным интерфейсом и умной эргономикой. NetBeans больше предназначена для тех, кто занимается кодированием серверов на Java, имеет очень медленный отладчик, и её программы занимают много оперативной памяти.

**2.4 Вывод о проектировании программы**

Подводя итоги проектирования, можно сказать, что создание программы тестирования во многом автоматизирует работу программистов. Программисту достаточно выбрать необходимый сценарий тестирования и запустить программу на исполнение, она в свою очередь сделает всю работу.

Также методом сравнительного анализа произведен выбор языка программирования, программы для автоматизации тестирования, эмуляции Android и среда разработки.

3 Разработка программы

3.1 Первоначальные настройки

Для разработки программы, необходимо установить список компонентов или проверить их наличие:

* эмулятор Android Studio;
* Appium Inspector;
* Appium;
* Java версии 7 и выше;
* IntelliJ IDEA;
* переменные среды для Java и Android SDK [13].

Установив Android Studio необходимо зайти в параметры SDK. Установить нужную версию платформы Android, инструменты API, и девайс для эмулятора. Первым этапом открыть инструменты SDK и установить версию платформы Android версии 8.0, рисунок 3.1. Вторым этапом зайти в инструменты и поставить все элементы Android SDK, именно эти инструменты отвечают за взаимодействие с мобильными приложениями, рисунок 3.2.

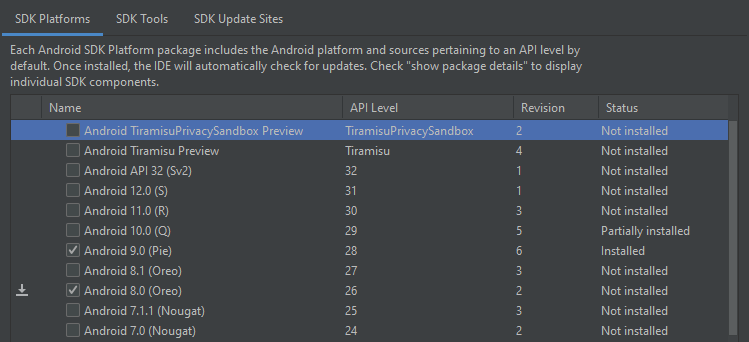


Рисунок 3.1 – Версия прошивки Android

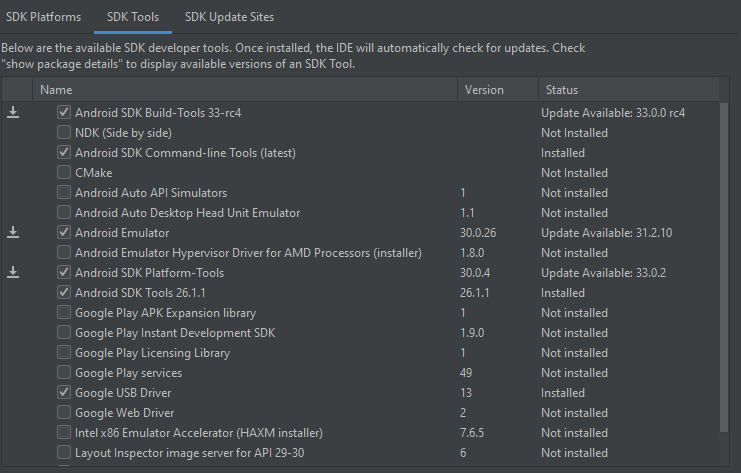


Рисунок 3.2 – Инструменты SDK

Третьим этапом нужно добавить девайс для эмулятора. Открываем Devise Manager и выбираем девайс по желанию, главное, чтобы он поддерживал нужную нам версию платформы. В рамках программы будет использован Pixel 2, рисунок 3.3. Дальше выбирается версия платформы, рисунок 3.4.

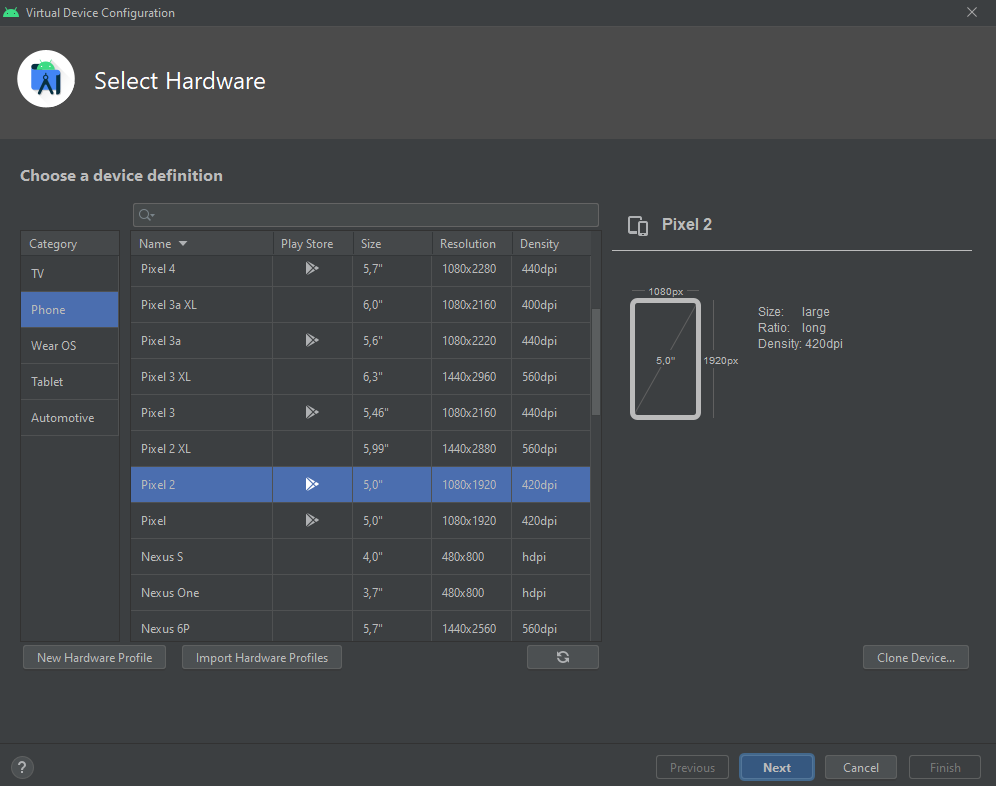


Рисунок 3.3 – Devices manager добавление нового девайса



Рисунок 3.4 – Выбор версии платформы Android

Четвертым этапом необходимо добавить переменные среды для Java и Android SDK. Для этого надо зайти в свойства компьютера и выбирать свойства, переменных среды. Создать новые переменные для директории Android SDK и Java, и добавить эти переменные в path, рисунки 3.5-3.7.

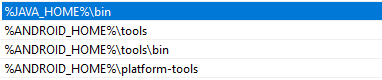


Рисунок 3.5 – Переменные среды в path

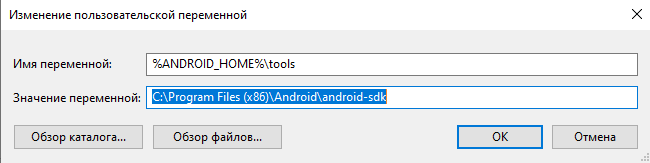


Рисунок 3.6 – Переменная среды для Android SDK

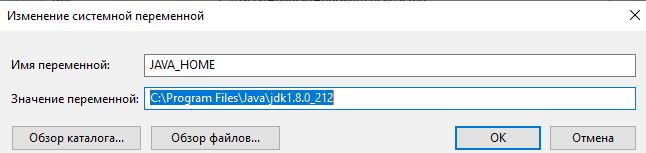


Рисунок 3.7 – Переменная среды для Java

Пятым этапом будет запуск программы Appium и установка ip сервера 127.0.0.1, как рекомендуют разработчики Appium [14], рисунок 3.8.

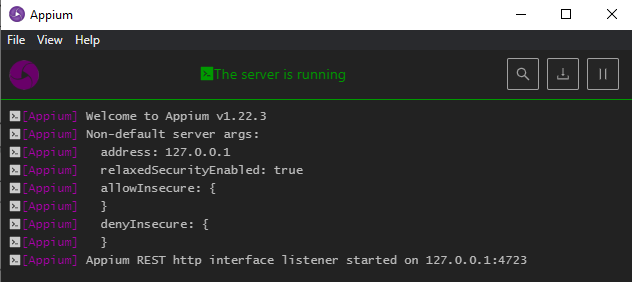


Рисунок 3.8 – Appium

Последним этапом будет установка и подключение Appium Inspector. При запуске программы нам необходимо вписать ip-адрес Appium сервера и номер порта. Также установить удаленный путь /wd/hub и прописать имя платформы, девайса и версию, рисунки 3.9-3.10 [15].



Рисунок 3.9 – Данные для подключения к хосту

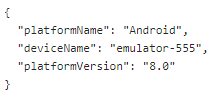


Рисунок 3.10 – Прописанные команды для подключения

3.2 Разработка программы

Когда произведены все первоначальные настройки можно приступить к разработке программы. Программа будет представлена в виде пяти классов. *Main*, в котором будет сформирован основной код программы. *Textlog* который будет связан с формированием файла журнала, для просмотра результатов тестирования, возникших ошибок, а также печати этой информации в консоль. *ErrorCodes*, в котором будет описано перечисление возможных ошибок и предупреждений, а также описана функция в которой указаны соответствия между кодами и натуральным описанием ошибок. *TestData*, для описания конфигурации каждого из тестов. *AppTest*, который будет являться ядром конфигурации всех тестов, потому что в нем будет производится загрузка параметров и списка тестов.

Распределение кода между классами прибавляет программе гибкости, так как при рефакторинге программы можно будет обратится к нужному классу, а не ко всему коду в целом. Весь проект показан на рисунке 3.11.

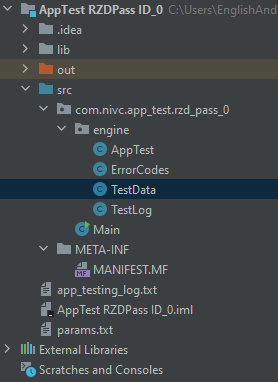


Рисунок 3.11 – Проект программы в IntelliJ IDEA

Первым этапом создан класс *Main* и подключены необходимые библиотеки, рисунок 3.12.

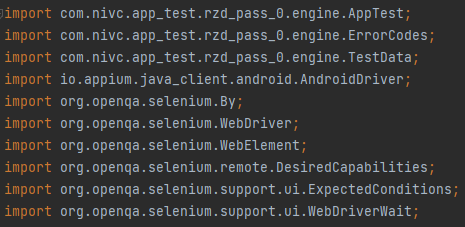


Рисунок 3.12 – Импортированные библиотеки

Библиотеки selenium крайне важны для работы с Appium, так как именно они позволяют обмениваться с данными с Appium сервером.

Вторым этапом создан класс *AppTest*. В него будет входить логин и пароль для авторизации пользователя. Два метода *getAppLogin* и *getAppPsw*, для вызова этих данных в классе Main, рисунок 3.13. Внутри был создан класс *Param* с методом для загрузки параметров тестирования и списка тестов из файла JSON. Это необходимо для тестирования определенного сценария использования мобильного приложения, рисунок 3.14. В директории программы будет находится файл «params.txt», который отвечает за конфигурирование различных функций теста, рисунок 3.15.

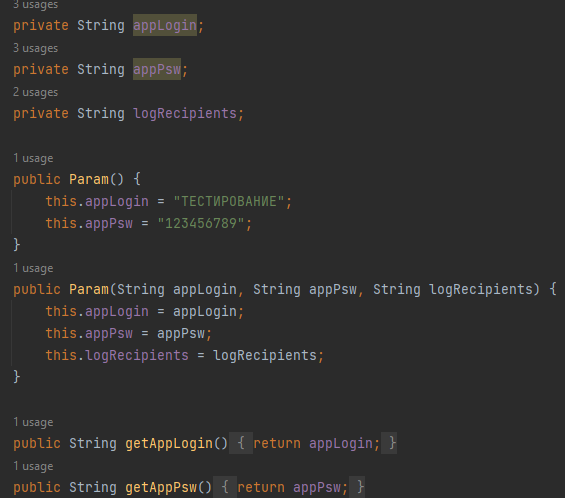


Рисунок 3.13 – Идентификация логина и пароля

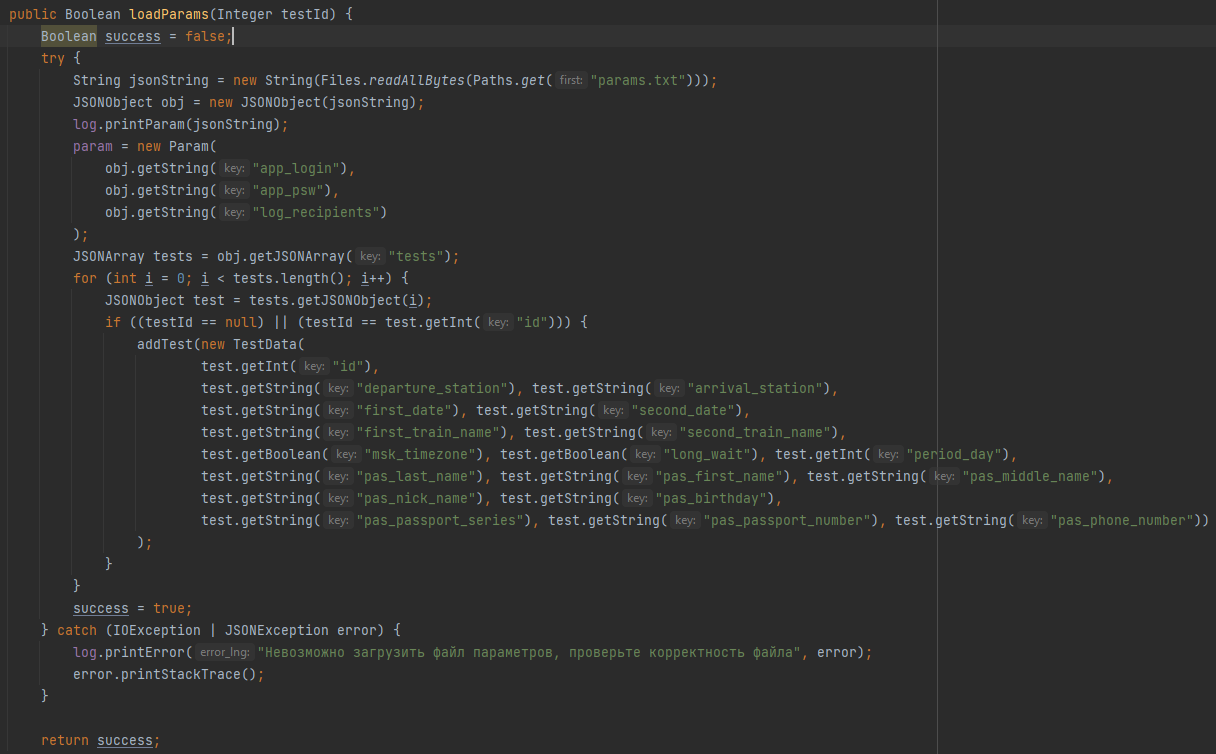


Рисунок 3.14 – Метод для загрузки параметров



Рисунок 3.15 – Файл параметров

Третьим этапом создан класс *ErrorCodes* в котором хранится перечисление возможных ошибок и предупреждений. Всего описано 42 ошибки, где для каждой ошибки назначен номер. Также создан метод *getErrorMsg*, где указаны соответствия между кодами и натуральным описанием ошибок, рисунки 3.16 - 3.17.

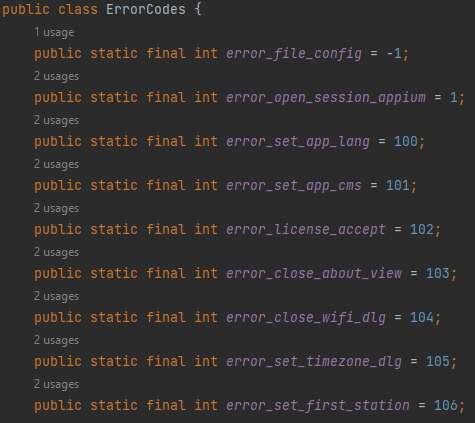


Рисунок 3.16 – Коды ошибок

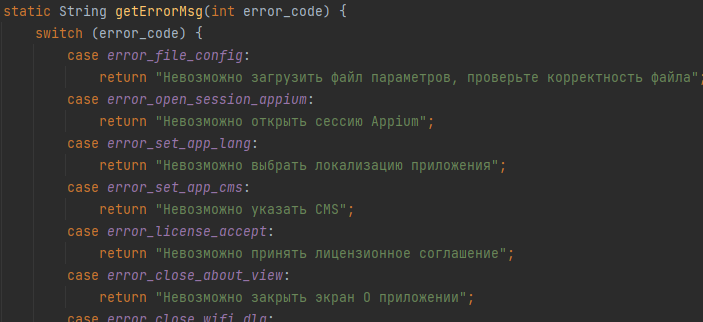


Рисунок 3.17 – Соответствия кода натуральному описанию ошибки

Четвертым этапом создан класс *TestLog*, с помощью которого будет производится печать логов в консоль и файл журнала. В данном классе описаны методы для работы с информацией об ошибках, поступающих от эмулятора программе. Класс содержит два метода *PrintError*, для вывода расширенной сообщения об ошибке в натуральной языковой форме. Какой метод будет использоваться зависит от передаваемых параметров. Метод *printMsg* для простой печати сообщения. Метод *printStartTest*, который сообщает о начале выполнения теста. Метод *printSuccessTest* сообщающий об успешном завершении теста. Метод *printParam* предназначенный для печати параметров файла конфигурации. Метод *writeMsg* для записи информации в файл/консоль. Все описанные выше методы показаны на рисунках 3.18 - 3.20.

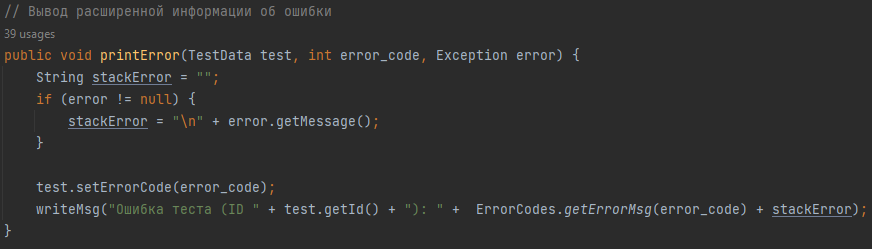


Рисунок 3.18 – Метод PringError

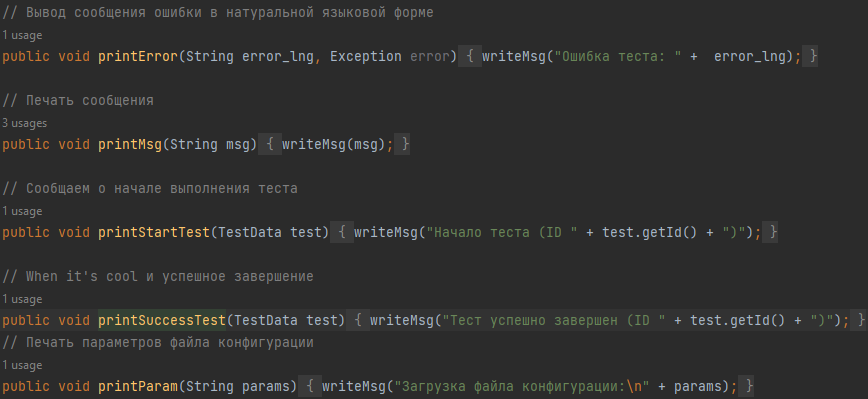


Рисунок 3.19 – Методы printError, printMsg, printStartTest, printSeccessTest и printParam

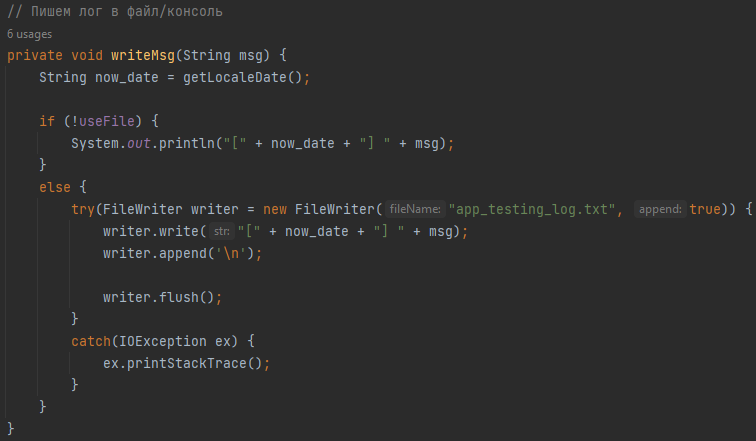


Рисунок 3.20 – Метод writeMsg

Пятый этап создание класса *TextData*, в котором хранится описание конфигурации каждого теста. В самом классе описаны все поля конфигурации, рисунок 3.21. Конструктор класса, который принимает данные об конфигурации, рисунок 3.22. Описаны методы для получения данных конфигурации, рисунок 3.23. Помимо этого создается метод для выбора даты отправления если она отсутствует и создан метода для получения кода ошибки.

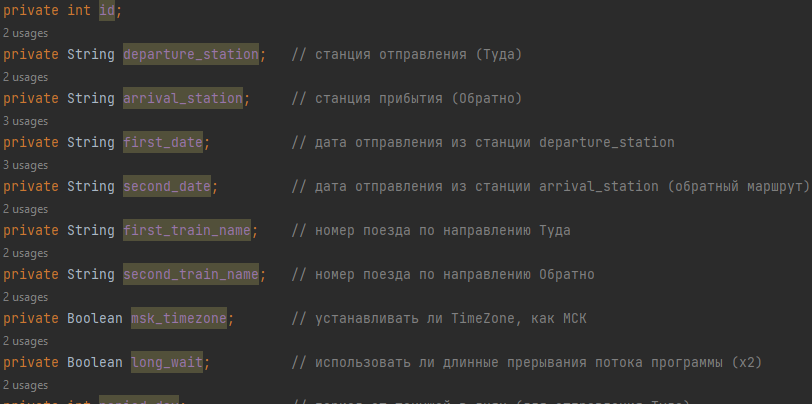


Рисунок 3.21 – Поля класса TestData

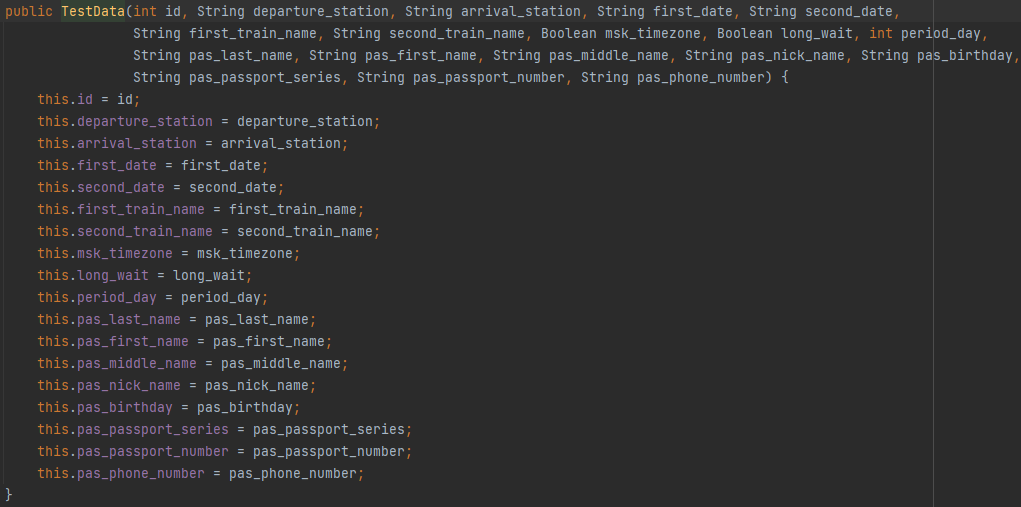


Рисунок 3.22 – Конструктор класса TestData

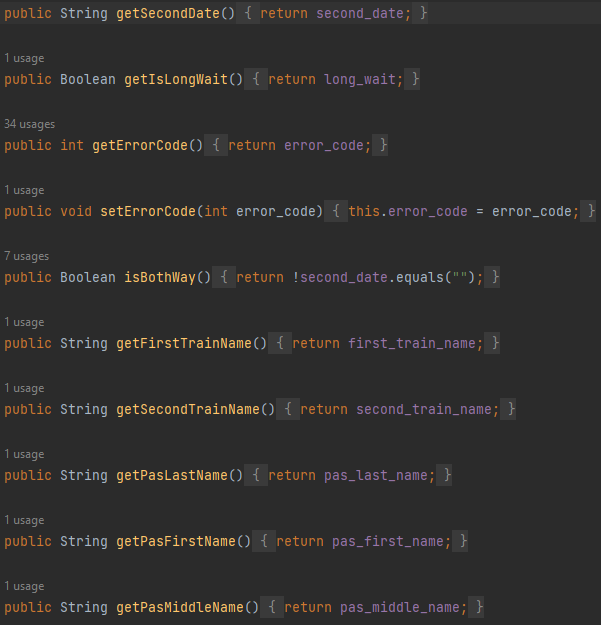


Рисунок 3.23 – Методы класса TestData

После создания классов можно перейти к проектированию скрипта для тестирования в классе Main. С начала созданы поля класса, экземпляры классов *Apptest*, *WevDriver* и *WebDriverWait* для работы с Appium. Переменная логического типа *enableLongWait* для длительного ожидания, в случае если элемент интерфейса будет долго появляться, рисунок 3.24.

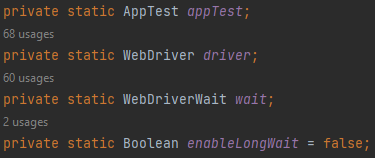


Рисунок 3.24 – Поля класса Main

Проектирование класса Main начинается с обращения к экземпляру класса *AppTest*, который отвечает за загрузку конфигурации. Отправляется сообщение в журнал о начале теста, и производится проверка загрузки конфигурации. Если конфигурация не была загружена, пишется сообщение о завершении тестирования, рисунок 3.25.



Рисунок 3.25 – Начало скрипта

Дальше идет блок кода, отвечающий за подключение среды разработки к Appium, передачи ему данных платформы, и файла APK, который содержит тестируемую программу. Прописана строка кода для связи с Appium сервером, где указан ip-адрес и номер порта, и передаются все данные платформы, рисунок 3.26.



Рисунок 3.26 – Установка параметров для подключения к Appium

Следующим этапом идет алгоритм тестирования по шагам, который основан на примере алгоритма при постановке задачи в разделе аналитического исследования. Каждый шаг представляет собой отдельный метод, рисунок 3.27.



Рисунок 3.27 – Алгоритм тестирования

Процесс тестирования заключен в цикл *for*, рисунок 3.28, который работает с экземпляром класса *AppTest*, пока он принимает тесты. После прохождения цикла, программа проверяет все ли тесты были выполнены. Если да, то сообщает в лог о успешном завершении теста и окончании работы, рисунок 3.29. Если нет, то цикл повторяется, но уже с новыми данными конфигурации.

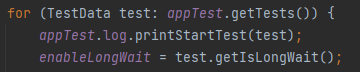


Рисунок 3.28 – Начало цикла for

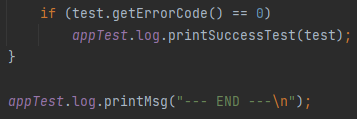


Рисунок 3.29 – Завершение цикла for

Как говорилось ранее, каждый этап алгоритма представлен отдельным методом. Каждый метод включает в себя блок *try catch*, для отлавливания ошибок. В часть *try* входит код связанный с использованием интерфейса приложения, то есть нажиманием на кнопки. Каждая кнопка имеет путь либо id, по которому к ней можно обратится. Данные пути и id можно найти при помощи Appium Inspector’а, рисунок 3.30. Важно знать, что не каждый элемент имеет путь и id, иногда будет доступен только один вариант обращения. После получения пути или id используется метод *FindElement*, который есть у экземпляра *WebDriver*. Он будет искать элемент по пути *By.xpath*, либо по id *By.id*. В части *catch* отлавливаются ошибки, которые могут возникнуть на определенном шаге алгоритма, информация об ошибке будет записана в журнал, рисунок 3.31.

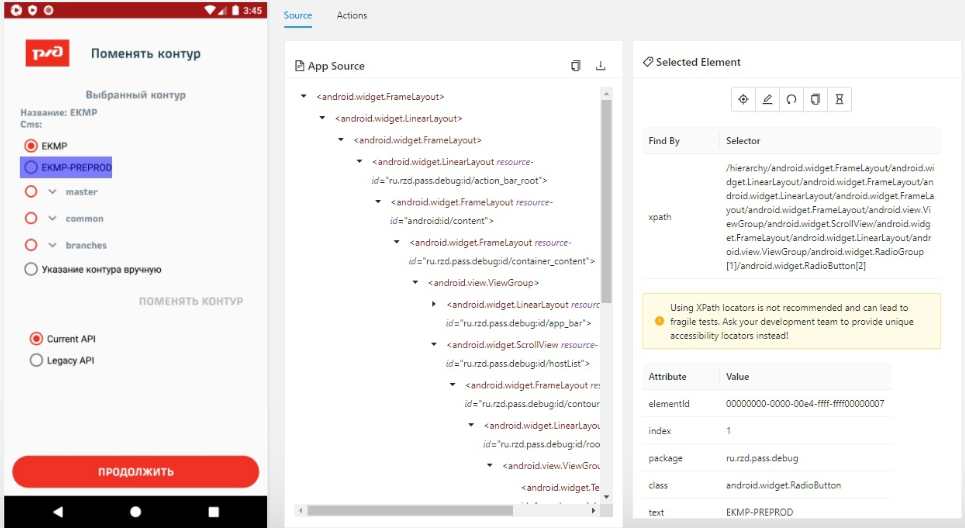


Рисунок 3.30 – Использование Appium Inspector

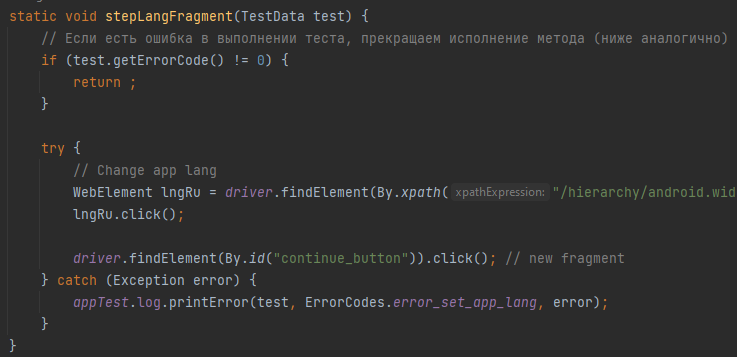


Рисунок 3.31 –Пример метода из алгоритма

3.3 Вывод о разработке программы

В ходе разработки программы был создан скрипт, позволяющий протестировать мобильное приложение «Пассажирам ОАО РЖД». Скрипт работает по стандартному сценарию покупки ЭБ на поезда дальнего сообщения и пригородные маршруты. Для удобства использования кода программа была разбита на классы, где каждый отвечает за свою область работы с данными. Код программы подкреплен комментариями что позволит без лишних трудностей понять за что отвечает каждый метод. Были описаны все шаги создания программы и показаны блоки кода каждого класса.

4 Руководство пользователя

Все элементы программы поставляются в отдельной папке. Где расположен файл журнала «app\_testing\_log.txt», сам программный продукт «TestProject*.*jar», конфигурация «params.txt», файл для запуска программы «runTest1», и файл APK, который является тестируемым мобильным приложением, рисунок 4.1.

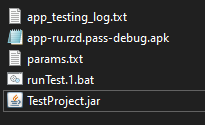


Рисунок 4.1 – Папка программы

Перед запуском программы необходимо запустить эмулятор Android и Appium. Запуск программы осуществляется открытием файла «runTest1.bat». Файл предполагает проход по всем тестам, которые описаны в файле конфигурации «params.txt».

После запуска файла должен начаться процесс тестирования, которое проводится по стандартному сценарию покупки ЭБ на поезда дальнего сообщения и пригородные маршруты. Весь процесс тестирования показан в таблице 4.1.

Таблица 4.1 - Алгоритм тестирования

| № | Шаг | Результат |
| --- | --- | --- |
|  | На устройстве (телефоне с ОС Android или iOs.) запустить приложение «РЖД Пассажирам» | Открылось приложение ЕКМП. |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| № | Шаг | Результат |
|  | В появившемся окне справочной информации (если оно появилось) нажать «Закрыть». | 3 - Справочная информация.jpg |
|  | Если открылось окно с предложением выбрать язык, необходимо выбрать (или оставить по умолчанию) «Русский» и нажать «Продолжить». Если же появилось окно вида 2 (с предложением выбора маршрута и даты отправления), перейти на шаг №7. | 1 - Выбор языка.jpgили 7 - Стартовая страница.jpg |
|  | В окне «Условия лицензионного соглашения» нажать «Принимаю». | 2 - Лицензионное соглашение.jpg |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| № | Шаг | Результат |
|  | В появившемся окне об определении доступности сетей wi-fi на вокзалах нажать «Нет». | 4 - Wi-fi.jpg |
|  | Если появилось окно с вопросом «Разрешить приложению РЖД Пассажирам доступ к данным о местоположении устройства?», нажать «Запретить». | 5 - Данные о местоположении устройства.jpg |
|  | В окне выбора часового пояса нажать «Использовать московское». | 6 - Выбор часового поезда.jpg |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| № | Шаг | Результат |
|  | Ввести станции отправления и назначения, выбрать из календаря дату «туда» (равную или большую текущей). | Станции выбираются из справочника, дата задана корректно. |
|  | Нажать на кнопку «Найти поезда» | После нажатия на кнопку отобразился список поездов на заданную дату. |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| № | Шаг | Результат |
|  | Выбрать тип вагона.  Если приложение запущено неавторизованным пользователем, необходимо авторизоваться под логином ТЕСТИРОВАНИЕ, пароль: 123456789 | После авторизации и выбора типа вагона, открылся список с вагонами выбранного типа, например, купе. |
|  | Выбрать произвольный вагон в списке. | Отображена информация о вагоне (тип, класс обслуживания, количество свободных мест, пиктограммы с услугами) и схема вагона. |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| № | Шаг | Результат |
|  | Выбрать на схеме вагона свободное место. | Выбранное место выделено цветом, отобразилось окно с информацией о стоимости и бонусах за покупку. |
|  | Нажать на текст «Выбрать 1 место. Продолжить» | Отобразилось окно «Данные пассажиров» |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| № | Шаг | Результат |
|  | Заполнить поле «Признак места», если оно актуально для выбранного вагона. Нажать на «+», чтобы добавить пассажира. | Открылось окно «Мои пассажиры».  Отобразилось меню «Добавить нового пассажира» и список ранее введенных пассажиров (если ранее пассажиры не были добавлены, то текст «Сохраненных пассажиров нет» |
|  | Добавить нового пассажира (ввести ФИО, номер паспорта, телефон) или выбрать пассажира из списка.  Отметить чек-бокс «Подтверждаю получение мною согласия третьих лиц на передачу их персональных данных на обработку».  Нажать «Продолжить» | Пассажир добавлен/выбран.  Отображается всплывающее окно «Страховки не оформлены. Оформите страховки или продолжите без оформления». |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| № | Шаг | Результат |
|  | Нажать «Купить» | Отобразилось окно «Получение чека». |
|  | Нажать «Продолжить» | Место забронировано. Осуществлен переход на окно оплаты». Возможен выбор оплаты Google Pay и банковской картой. |
|  | Выбрать «Банковская карта».  Нажать «Оплатить» | Отображается окно «Оплата» с полями для ввода реквизитов банковской карты. |

Если в ходе проверки ошибок не обнаружено и удалось выполнить все действия по сценарию (для пригородных поездов выбор места, покупка страховки и оформление багажа не производится), программа должна отправить соответствующее сообщение в файл журнала. Файл журнала после успешного завершения тестирования представлен на рисунке 4.2.



Рисунок 4.2 – Файл журнала

Тестирование идет согласно файлу конфигурации теста, состав которого виден на рисунке 4.2. Сам файл конфигурации как видно на рисунке 4.1, находится в папке программы и его можно изменять либо дополнять новыми конфигурациями. Специально для пользователя была описана структура всей конфигурации.

Вначале структуры содержится общая системная информация:

- *app\_login*, логин для авторизации в приложении (строка);

- *app\_psw*, пароль для авторизации в приложении (строка).

Далее описан массив «tests», который отвечает за входные параметры для каждого теста, в частности. Каждый объект массива содержит следующие параметры:

- *id*, ID запускаемого теста (число);

- *departure\_station*, станция отправления (строка);

- *arrival\_station*, станция прибытия (строка);

- *first\_date*, дата первого отправления (строка, формат «число месяц»);

- *second\_date*, дата обратного отправления (строка, необязательный параметр – оставить пустым, формат «число месяц»);

- *first\_train\_name*, номер приоритетного поезда для первого отправления (строка, необязательный параметр);

- *second\_train\_name*, номер приоритетного поезда для обратного отправления (строка, необязательный параметр);

- *msk\_timezone*, использовать ли часовой пояс МСК, (логическая переменная, true/false);

- *long\_wait*, удваивать ли время стандартного ожидания взаимодействия с Appium, (логическая переменная, true/false);

- *period\_day*, формирование даты отправления, как текущая + period\_day дней (Метод класса TestData, число);

- *pas\_last\_name*, фамилия пассажира (строка);

- *pas\_first\_name*, имя пассажира (строка);

- *pas\_middle\_name*, отчество пассажира (строка);

- *pas\_nick\_name*, ник пассажира (строка);

- *pas\_birthday*, день рождения пассажира (строка);

- *pas\_passport\_series*, серия паспорта (строка);

- *pas\_passport\_number*, номер паспорта (строка);

- *pas\_phone\_number* // номер телефона пассажира (строка, формат – без 8 и +7).

Таким образом можно создавать новые тесты индексируя их новым *id*. Чтобы запустить какой-то конкретный тест следует указать его *id* из массива «*test*». Такая команда будет выглядеть следующем образом «*java* *–jar* *TestProject.jar* *2*», где 2 это ID теста из файла конфигурации.

ЗАКЛЮЧЕНИЕ

Цель данной работы заключалась в проектирования программы автоматизированного тестирования работоспособности мобильного приложения «Пассажирам ОАО РЖД».

Для достижения поставленной цели были решены следующие задачи:

- проведено аналитическое исследование предметной области;

- описаны бизнес-процессы тестирования мобильного приложения;

- определены средства разработки программного обеспечения;

- проведен сравнительный анализ языков программирования и сред разработки;

- разработана программа тестирования;

- разработано руководство пользователя.

В процессе проведения практической и теоретической работы решены поставленные задачи и достигнута цель. Исходя из этого, выпускную квалификационную работу можно считать проработанной.

Результатом выполнения данной работы является программный продукт, отвечающий заявленным требованиям и готовый к использованию для тестирования мобильного приложения «Пассажирам ОАО РЖД».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 GitHub gist [Электронный ресурс] – UR: https://gist.github.com/ codedokode/a455bde7d0748c0a351a (Дата обращения: 06.06.2022)

2 Хабр [Электронный ресурс] – URL: https://habr.com/ru/post/160257/ (Дата обращения: 07.06.22)

3 UML для бизнес-моделирования: зачем нужны диаграммы процессов [Электронный ресурс]. – URL: https://evergreens.com.ua/ru/articles/uml-diagrams.html (Дата обращения: 03.06.2022)

4 Самые актуальные и популярные языки программирования [Электронный ресурс]. – URL: https://zen.yandex.ru/media/id/60a377aaaad0d8392f9dda44/samye-aktualnye-i-

populiarnye-iazyki-programmirovaniia-60a8f364e4677d6f7cf4f072 (Дата обращения: 03.06.2022)

5 Почему Java популярна в разработке [Электронный ресурс]. – URL: https://proglib.io/p/pochemu-java-ispolzuetsya-dlya-razrabotki-mobilnyh-prilozheniy-2021-07-01 (Дата обращения: 03.06.2022)

6 Топ 10 лучших сред разработки на Java[Электронный ресурс]. – URL: https://gitjournal.tech/10-luchshih-sred-razrabotki-na-java (Дата обращения: 03.06.2022)

7 Лучшие языки программирования для Android-разработчика [Электронный ресурс]. – URL: https://tproger.ru/articles/8-jazykov-programmirovanija-dlja-android-azrabotchika/ (Дата обращения: 03.06.2022)

8 Хабр [Электронный ресурс] – URL: [Электронный ресурс] – URL: https://habr.com/ru/post/160257/ (Дата обращения: 07.06.22)

9 JetBrains [Электронный ресурс] – URL: https://www.jetbrains.com/ru-ru/idea/ (Дата обращения 08.06.22)

10 Почему Appium для народа [Электронный ресурс]. – URL: https://habr.com/ru/post/488482/ (Дата обращения: 05.06.2022)

11 Tunecom [Электронный ресурс] – URL: https://tunecom.ru/virtual-machines/390-ispolzuem-android-studio-v-kachestve-jemuljatora-smartfona.html (Дата обращения: 07.06.22)

12 Appium [Электронный ресурс]. – URL: https://github.com/appium (Дата обращения: 05.06.2022)

13 Процесс тестирования мобильного приложения [Электронный ресурс]. – URL: https://habr.com/ru/company/touchinstinct/blog/197060/ (Дата обращения: 03.06.2022)

14 Running Tests [Электронный ресурс]. – URL: https://appium.io/docs/en/writing-running-appium/running-tests/ (Дата обращения: 05.06.2022)

15 Руководство по Appium для тестирования устройств под Android [Электронный ресурс]. – URL: https://habr.com/ru/post/488482/ (Дата обращения: 05.06.2022)