Утверждаю

Руководитель предприятия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, печать предприятия)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

ОТЧЕТ

по производственной технологической практике

в ООО «Форанкс» 17.06.2024 ­– 12.07.2024

(наименование предприятия, сроки практики)

Исполнитель

студент 3 курса 4 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пшенко А.Ф.

(подпись, дата) (Ф.И.О.)

Руководитель практики от предприятия

заместитель директора Бородако М.И.

(должность, печать предприятия) (подпись, дата) (Ф.И.О.)

Руководитель практики от университета

преп.-стажер Якунович А.В.

(должность, уч. звание) (подпись, дата) (Ф.И.О.)

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Минск 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc171677146)

[1 Описание компании ООО «Форанкс» 4](#_Toc171677147)

[1.1 История создания компании и основные виды деятельности 4](#_Toc171677148)

[1.2 Информационные технологии ООО «Форанкс» 5](#_Toc171677149)

[2 Описание используемых технологий 6](#_Toc171677150)

[2.1 Skype for Business 6](#_Toc171677152)

[2.2 Design Studio 6](#_Toc171677153)

[2.3 OpenVPN 6](#_Toc171677154)

[2.4 JBoss 6](#_Toc171677155)

[3 Индивидуальное задание 8](#_Toc171677156)

[3.1 Постановка задачи 8](#_Toc171677158)

[3.2 Выбор подходящих технологий 9](#_Toc171677159)

[3.3 Реализация программного средства 9](#_Toc171677160)

[4 Тестирование программного средства 14](#_Toc171677161)

[ЗАКЛЮЧЕНИЕ 16](#_Toc171677162)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 17](#_Toc171677163)

[ПРИЛОЖЕНИЕ А 18](#_Toc171677164)

[ПРИЛОЖЕНИЕ Б 20](#_Toc171677165)

[ПРИЛОЖЕНИЕ В 22](#_Toc171677166)

# ВВЕДЕНИЕ

Сейчас информационный мир быстро расширяется, в результате чего мы сталкиваемся с большим объемом информации и таким же большим объемом технологических инструментов для взаимодействия с ним. Языки программирования и области применения программ развиваются не менее быстро. Возникают новые компании, какие-то из них пополняют обилие программ на рынке, а какие-то, напротив, предлагают другим пакет услуг, включающий в себя стандартные, но необходимые IT-решения. Одной из компаний, которая предоставляет качественное и необходимое банковское ПО является ООО «Форанкс».

Прохождение производственной практики является важным этапом обучения. Это специфический вид учебного процесса, в ходе которого осуществляется связь обучения с производством, где студент сам может увидеть производство и что-то попробовать сделать, а также выполнить свое индивидуальное задание. Практика должна способствовать формированию у студентов профессиональных практических знаний, умений и навыков, необходимых для будущей работы на предприятии.

Прохождение практики осуществлялось в одном из офисов ООО «Форанкс».

Целью производственной практики является ознакомление студентов с реальными условиями работы на предприятии. Проходя практику, студенты имеют возможность выполнить задания и показать себя, и, кроме того, успешно справившиеся с заданием студенты могут быть приглашены на стажировку, а позднее и на работу в данное предприятие после прохождения производственной практики.

В этот период было применено на практике ПО, которое используется сотрудниками организации. Было выполнено индивидуальное задание по разработке заданного приложения.

Также при прохождении практики были поставлены следующие задачи:

* ознакомиться с историей создания и деятельностью компании;
* ознакомиться с организационной структурой компании;
* ознакомиться с информационными технологиями, используемыми в компании;
* ознакомиться с методами информационной безопасности в компании;
* сформировать план работ на время производственной практики;
* проанализировать и обобщить полученную информацию;
* результат выполнения включить в отчет по практике.

# 1 Описание компании ООО «Форанкс»

## История создания компании и основные виды деятельности

ООО «Форанкс» — это компания, специализирующаяся на разработке, внедрении и поддержке автоматизированных банковских систем с 2004 года. Компания была создана в результате преобразования компании «СТ-Софт» и с июля 2007 года существует как самостоятельная структура.

«Форанкс» предоставляет своим клиентам услуги по разработке собственного программного обеспечения, локализации, внедрению и поддержке программного обеспечения фирм-партнеров. Компания является партнером компании Temenos, согласно партнерскому соглашению, имеет право участвовать в проектах по внедрению системы Temenos Transact (T24) и ее отдельных компонентов. ООО «Форанкс» также имеет официальный статус «Upgrade Partner» компании Temenos.

Компания успешно прошла все стадии внешнего аудита в 2021 году и получила дополнительный официальный статус «Temenos Development Partner». Новый статус партнерства позволяет гарантировать клиентам, что IT и бизнес-процессы ООО «Форанкс» соответствуют международным стандартам. Компания имеет в своем штате необходимое количество сертифицированных специалистов с соответствующим набором компетенций для оказания услуг по разработке в рамках внедрения и развития банковской автоматизированной системы Temenos Transact и ее отдельных компонентов.

Компания «Форанкс» оказывает дополнительные услуги, такие как анализ, обучение и оптимизацию. Она также имеет партнеров, таких как Tech Mahindra / SOFGEN.

Все работы по внедрению и поддержке проектов выполняются высококвалифицированными специалистами компании, среди которых: руководители проектов, бизнес-аналитики, технические консультанты и специалисты по тестированию. Компания «Форанкс» гибка, мобильна и следует интересам своих клиентов, помогая им совершенствовать свой бизнес и делать свои возможности неисчерпаемыми.

## 1.2 Информационные технологии ООО «Форанкс»

Каждому сотруднику компании предоставляется стандартный пакет программ, а также дополнительный в соответствии с занимаемой им должностью. Стандартный пакет программ включает:

* *Skype for Business* для ведения деловой переписки, проведения совещаний и переговоров. Данное ПО также позволяет компании отказаться от использования стационарных телефонов.
* *Design Studio* – плагин для *Eclipse*, который используется в рабочем процессе в качестве IDE.
* *OpenVPN* – открытое ПО для подключения к частной виртуальной сети компании.

В зависимости от технологий, используемых на проекте, специалисту может предоставляться следующее ПО:

* *JBOSS* – платформа сервера приложений с открытым кодом.

При необходимости сотрудник может скачать лицензированное ПО со внутреннего сетевого хранилища.

# 2 Описание используемых технологий



## Skype for Business

Skype for Business был корпоративной версией популярного мессенджера Skype, разработанной компанией Microsoft для использования в бизнес-среде. Он предоставлял возможность для проведения онлайн-встреч, видеоконференций, обмена мгновенными сообщениями и предоставления доступа к рабочим файлам и документам.

Skype for Business был интегрирован в другие программные продукты Microsoft, такие как Microsoft Office и SharePoint, и предлагал расширенные функции для бизнес-пользователей, такие как совместная работа над документами, запись встреч, шифрование данных и многое другое. Он также интегрировался со сторонними приложениями и устройствами для удобства пользователей.

В 2018 году Microsoft объединила Skype for Business с другой своей программой - Microsoft Teams, чтобы создать новый универсальный инструмент для коммуникации и совместной работы в офисной среде.

## Design Studio

Design Studio - это интегрированная среда разработки (IDE), разработанная компанией Temenos для создания пользовательских интерфейсов в приложениях на базе платформы Temenos Transact. Design Studio предоставляет разработчикам широкий набор инструментов для создания и настройки пользовательских интерфейсов, что облегчает создание профессионально выглядящих и удобных в использовании интерфейсов для конечных пользователей.

## OpenVPN

OpenVPN - это открытый программный продукт (open-source), который обеспечивает безопасное и защищенное соединение между компьютерами в сети Интернет. Он использует протокол SSL/TLS для шифрования данных и создания виртуальной частной сети (VPN) между удаленными компьютерами или сетями.

OpenVPN может использоваться для обеспечения безопасного удаленного доступа к корпоративным ресурсам, для обхода цензуры в Интернете, для обеспечения безопасного соединения в публичных Wi-Fi сетях и многого другого.

OpenVPN может работать на операционных системах Windows, macOS, Linux, Android и iOS, а также на многих других платформах. Он также может быть интегрирован с другими программными продуктами, такими как маршрутизаторы и брандмауэры, для обеспечения безопасного соединения на уровне всей сети.

## JBoss

JBoss - это платформа для разработки и развертывания приложений на основе Java, которая была приобретена компанией Red Hat в 2006 году. JBoss представляет собой набор инструментов и приложений, которые позволяют разработчикам создавать и развертывать приложения на основе Java, такие как веб-приложения, приложения для обработки бизнес-логики и многие другие.

JBoss предоставляет ряд готовых решений и инструментов для разработки приложений, таких как JBoss Enterprise Application Platform, JBoss Data Grid, JBoss BPM Suite и другие. Он также предоставляет средства для управления и мониторинга приложений, включая инструменты для анализа производительности и отладки.

JBoss использует технологии, такие как JavaServer Faces (JSF), Hibernate, Java Messaging Service (JMS), Enterprise JavaBeans (EJB), и многие другие, что делает его одним из наиболее гибких и расширяемых решений для разработки приложений на основе Java.

JBoss также имеет широкую поддержку сообщества разработчиков и пользователей, что позволяет быстро решать проблемы и получать помощь при разработке приложений.

JBoss является одним из наиболее популярных и широко используемых решений для разработки и развертывания приложений на основе Java. Он позволяет разработчикам быстро создавать и развертывать приложения, обеспечивая высокую производительность и масштабируемость.

# 3 Индивидуальное задание



## Постановка задачи

Целью данной работы будет разработка веб-сервиса, который принимает SOAP-запрос (листинг 3.1) и отправляет ответ, содержащий код ошибки, текстовое название ошибки и ее описание. В случае, если ошибок не возникло, клиенту отправляется сообщение «OK».

Реализация проекта будет выполняться на языке программирования Java. Кроме того, должно проверяться, что содержимое значения тега clientid обязательно содержит 8 цифр.

Листинг 3.1 –SOAP Request

|  |
| --- |
| <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ser="http://service.getinfo.com/">  <soapenv:Header/>  <soapenv:Body>  <ser:GetInformation>  <GetInformationArguments>  <username>Artyom</username>  <password>qwerty</password>  <parameters>  <paramKey>clientid</paramKey>  <paramValue>11111111</paramValue>  </parameters>  <parameters>  <paramKey>paramKey2</paramKey>  <paramValue>paramValue2</paramValue>  </parameters>  <serviceId>8</serviceId>  </GetInformationArguments>  </ser:GetInformation>  </soapenv:Body>  </soapenv:Envelope> |

Листинг 3.2. SOAP Response

|  |
| --- |
| <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  <soap:Body>  <soap:Fault>  <faultcode>soap:Errors</faultcode>  <faultstring>Validation Errors</faultstring>  <detail>  <error>  <code>10</code>  <info>non unique clientid</info>  <message>user with such clientid is already exist</message>  </error>  </detail>  </soap:Fault>  </soap:Body>  </soap:Envelope> |

## Выбор подходящих технологий

В данном проекте было решено использовать Java 8 (JDK 1.8) по причине того, что эта версия является стабильной и широко используемой в банковской сфере.

Основой проекта станет JAX-WS. Это фреймворк для разработки веб-сервисов на языке Java, который позволяет создавать SOAP-сервисы. Он предоставляет множество функций и возможностей, которые помогают ускорить процесс разработки и сократить количество кода, необходимого для создания веб-сервисов.

Для развертывания приложения был выбран сервер JBoss/Wildfly 12.0.0. Это сервер приложений, который поддерживает Java EE и обеспечивает надежное развертывание и управление приложениями. Он обеспечивает высокую производительность, масштабируемость и безопасность.

Если приходит запрос с валидными данными, данные о пользователе сохраняются в БД. В данном проекте используется легковесная реляционная база данных SQLite. Она проста в использовании и не требует установки сервера базы данных, что делает ее идеальной для небольших приложений и тестовых сред.

## Реализация программного средства

Как уже говорилось ранее, JAX-WS позволяет упрощать процесс написания веб-сервисов путем создания Java-классов с аннотациями. После создания такого класса «под капотом» происходит генерация WSDL-схемы, которая описывает функциональные возможности нашего сервиса: какие запросы он принимает и что отправляет в ответ. Класс, на основе которого формируется WSDL-схема, приведен в листинге 3.3.

Листинг 3.3 – Класс, на основе которого формируется WSDL-схема

|  |
| --- |
| @XmlRootElement(name = "GetInformationArguments")  public class GetInformationArguments {  private String password;  private String username;  private String serviceId;  private List<Parameters> parametersList;  @XmlElement  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  @XmlElement  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  @XmlElement  public String getServiceId() {  return serviceId;  }  public void setServiceId(String serviceId) {  this.serviceId = serviceId;  }  @XmlElement(name = "parameters")  public List<Parameters> getParametersList() {  return parametersList;  }  public void setParametersList(List<Parameters> parametersList) {  this.parametersList = parametersList;  }  public int getClientidFromParameters() {  if (parametersList != null) {  for (Parameters param : parametersList) {  if ("clientid".equals(param.getParamKey())) {  try {  return Integer.parseInt(param.getParamValue());  } catch (NumberFormatException e) {  e.printStackTrace();  return 0;  }  }  }  }  return 0;  }  } |

Полученная WSDL-схема приведена в приложении А. Для хранения ошибок, которые могут возникнуть при валидации данных, было создано перечисление ErrorKey, которое хранит название ошибки, и два класса: структура самой ошибки и класс, хранящий сами ошибки. Перечисление приведено в листинге 3.4.

Листинг 3.4. Перечисление, хранящее название всех ошибок

|  |
| --- |
| public enum ErrorKey {  EMPTY\_PASSWORD,  EMPTY\_USERNAME,  EMPTY\_PARAMETERS,  EMPTY\_PARAM\_KEY,  EMPTY\_PARAM\_VALUE,  INVALID\_CLIENTID,  EMPTY\_SERVICEID,  INSUFFICIENT\_PARAMETERS,  NO\_CLIENT\_ID,  MULTIPLE\_CLIENT\_IDS,  NON\_UNIQUE\_CLIENT\_ID  } |

Бизнес-модель самой ошибки (класс ErrorDetail) приведена в листинге 3.5.

Листинг 3.5. Бизнес-модель ошибки

|  |
| --- |
| public class ErrorDetail {  private String code;  private String message;  private String detail;  public ErrorDetail(String code, String message, String detail) {  this.code = code;  this.message = message;  this.detail = detail;  }  public void setCode(String code) {  this.code = code;  }  public void setMessage(String message) {  this.message = message;  }  public void setDetail(String detail) {  this.detail = detail;  }  public String getCode() {  return code;  }  public String getMessage() {  return message;  }  public String getDetail() {  return detail;  }  } |

Класс, хранящий сами ошибки, приведен в листинге 3.6. Каждая ошибка, по сути, является объектом словаря, где ключом выступает значение из перечисления ErrorKey, а значением – объект класса ErrorDetail.

Листинг 3.6. Класс, хранящий все ошибки

|  |
| --- |
| public class ErrorRegistry {  private static final Map<ErrorKey, ErrorDetail> errorMap = new HashMap<>();  static {  errorMap.put(ErrorKey.EMPTY\_PASSWORD, new ErrorDetail("1", "is empty", "password is empty"));  errorMap.put(ErrorKey.EMPTY\_USERNAME, new ErrorDetail("2", "is empty", "username is empty"));  errorMap.put(ErrorKey.EMPTY\_PARAMETERS, new ErrorDetail("3", "is empty", "parameters block is empty"));  errorMap.put(ErrorKey.EMPTY\_PARAM\_KEY, new ErrorDetail("4", "is empty", "paramKey is empty"));  errorMap.put(ErrorKey.EMPTY\_PARAM\_VALUE, new ErrorDetail("5", "is empty", "paramValue is empty"));  errorMap.put(ErrorKey.INVALID\_CLIENTID, new ErrorDetail("6", "invalid paramValue", "paramValue should contain exactly 8 digits for clientid"));  errorMap.put(ErrorKey.EMPTY\_SERVICEID, new ErrorDetail("7", "is empty", "serviceid is empty"));  errorMap.put(ErrorKey.INSUFFICIENT\_PARAMETERS, new ErrorDetail("8", "insufficient parameters", "two parameter blocks are required"));  errorMap.put(ErrorKey.NO\_CLIENT\_ID, new ErrorDetail("8", "clientid is required", "one paramkey must be clientid"));  errorMap.put(ErrorKey.MULTIPLE\_CLIENT\_IDS, new ErrorDetail("9", "multiple client ids", "you must input only one clientid parameter"));  errorMap.put(ErrorKey.NON\_UNIQUE\_CLIENT\_ID, new ErrorDetail("10", "non unique clientid", "user with such clientid is already exist"));  }  public static ErrorDetail getError(ErrorKey errorKey) {  return errorMap.get(errorKey);  }  } |

Класс, который является самим веб-сервисом, приведен в листинге 3.7. Он наглядно демонстрирует общую логику работы сервиса: приходит SOAP-запрос с данными (объект класса GetInformationArguments), вызывается метод для валидации данных, и если все данные валидны, формируется ответ пользователю с сообщением «ОК», а пользователь сохраняется в базу данных. В противном случае формируется ответ, содержащий все ошибки валидации.

Листинг 3.7. Веб-сервис, принимающий запрос и формирующий ответ

|  |
| --- |
| @WebService  public class ValidateInformation {  @WebMethod(operationName = "GetInformation")  public GetResponse getInformation(@WebParam(name = "GetInformationArguments") GetInformationArguments arguments) {  List<ErrorDetail> errors = ValidationUtil.validateArguments(arguments);  if (!errors.isEmpty()) {  SOAPErrorUtil.handleValidationErrors(errors);  }  ClientDAO clientDAO = new ClientDAO();  clientDAO.saveClient(arguments);  GetResponse response = new GetResponse();  response.setMSG("OK");  return response} } |

Класс, служащий для валидации параметров запроса, приведен в приложении Б. Если все данные валидны, пользователь сохраняется в базу данных. Класс, служащий для сохранения пользователя в базу данных, приведен в листинге 3.8.

Листинг 3.8. Класс для сохранения пользователя в базу данных

|  |
| --- |
| public class ClientDAO {  public void saveClient(GetInformationArguments arguments) {  Connection conn = null;  PreparedStatement stmt = null;  try {  conn = SQLiteConnection.getConnection();  String query = "INSERT INTO Clients (clientid, username, password, serviceid) VALUES (?, ?, ?, ?)";  stmt = conn.prepareStatement(query);  stmt.setInt(1, arguments.getClientidFromParameters());  stmt.setString(2, arguments.getUsername());  stmt.setString(3, hashPassword(arguments.getPassword()));  stmt.setString(4, arguments.getServiceId());  stmt.executeUpdate();  } catch (SQLException e) {  e.printStackTrace();  } finally {  if (stmt != null) {  try {  stmt.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  }  private String hashPassword(String plainTextPassword) {  return BCrypt.hashpw(plainTextPassword, BCrypt.gensalt());  }  } |

Пароли пользователей в базе данных не хранятся в открытом виде, а хешируются с помощью функции hashpw из BCrypt.

После этого был создан еще один SOAP-сервис, принимающий в запросе clientid в качестве параметра и проверяющий наличие такого клиента в базе данных. Если клиент найден, информация о нем возвращается в ответе, если нет – сообщение о том, что такой клиент не найден.

Принцип его разработки абсолютно идентичен основному запросу, описанному выше.

# 4 Тестирование программного средства

Тестирование программы является важнейшим этапом в процессе разработки. Для этого привлекаются различные люди и конкретно направленные специалисты. Так как в этой программе присутствует небольшое количество потенциально опасных мест, тестирование будет проводиться собственными силами.

Приложение от многочисленных вылетов помогает спасти конструкция try-catch, благодаря которой даже после очевидных ошибок программное средство может находиться в рабочем состоянии.

Для основного запроса были написаны автотесты с использованием JUnit – популярного фреймворка для модульного тестирования в Java. JUnit позволяет легко писать и запускать тесты, что помогает обеспечивать качество кода и предотвращать ошибки. В нашем проекте тесты были разработаны для проверки корректности работы SOAP-сервиса.

Сначала был создан вспомогательный класс ValidationUtilTestHelper, заполняющий объект класса GetInformationArguments валидными данными. Он приведен в листинге 5.1.

Листинг 5.1. Вспомогательный класс с валидными данными

|  |
| --- |
| public class ValidationUtilTestHelper {  public static GetInformationArguments createValidArguments() {  GetInformationArguments arguments = new GetInformationArguments();  arguments.setPassword("password");  arguments.setUsername("username");  arguments.setServiceId("service123");  Parameters param1 = new Parameters();  param1.setParamKey("clientid");  param1.setParamValue("00000000");  Parameters param2 = new Parameters();  param2.setParamKey("otherKey");  param2.setParamValue("someValue");  List<Parameters> paramsList = new ArrayList<>();  paramsList.add(param1);  paramsList.add(param2);  arguments.setParametersList(paramsList);  return arguments;  }  } |

После этого был создан класс, тестирующий различные кейсы с невалидными данными. Он приведен в приложении В. Для составления отчета о степени покрытия кода тестами была использована библиотека покрытия кода Jacoco.

Для моего SOAP-сервиса степень покрытия кода тестами составила 85%. Результат работы библиотеки приведен на рисунке 5.1.

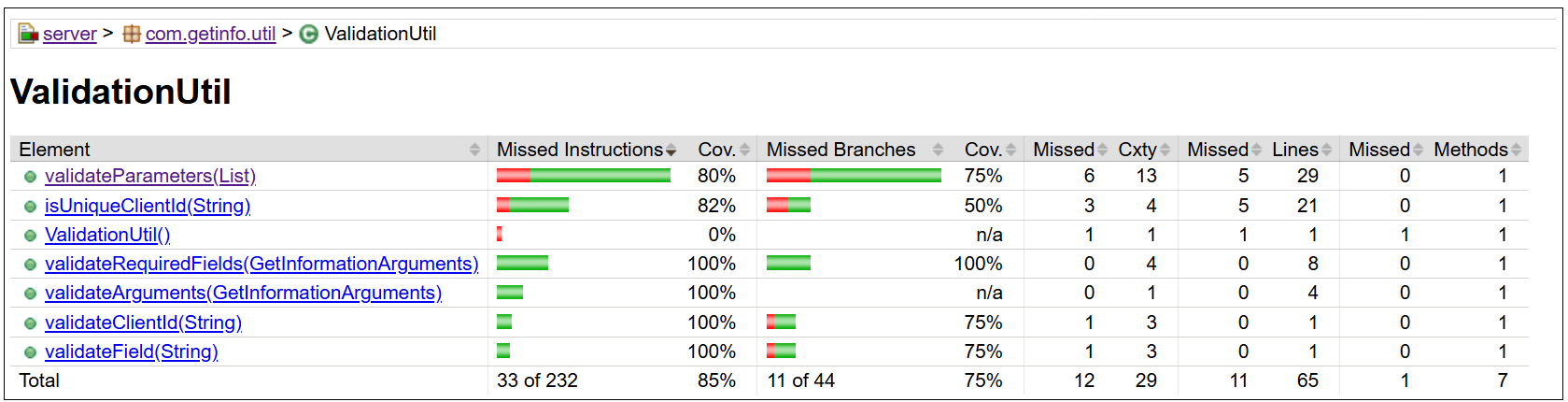


Рисунок 5.1 – Степень покрытия основного запроса тестами

# ЗАКЛЮЧЕНИЕ

Во время прохождения производственной практики подробно изучена деятельность компании. Углубленно изучены методики разработки приложений, изучено ПО, используемое сотрудниками компании для выполнения своих обязанностей.

В качестве индивидуального задания разработано несколько программ, каждое из которых протестировано и одобрено сотрудниками.

За время практики улучшены навыки программирования на Java и другие технологии.

Также получен опыт работы в команде и эффективного взаимодействовать с коллегами и руководителями проектов.

Полученный опыт и знания будут мне полезны в дальнейшей работе в области разработки программного обеспечения. Я благодарен команде компании за возможность пройти практику и за продуктивное сотрудничество.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Introduction to JAX-WS [Электронный ресурс]. Режим доступа: https://www.baeldung.com/jax-ws. Дата доступа: 30.06.2024.
2. A Guide to JUnit 5 [Электронный ресурс]. Режим доступа: https://www.baeldung.com/junit-5. Дата доступа: 01.07.2024.
3. Temenos. Официальный сайт [Электронный ресурс]. Режим доступа: https://www.temenos.com/. Дата доступа: 04.07.2024.
4. WildFly Getting Started Guide [Электронный ресурс]. Режим доступа: https://docs.wildfly.org/28/Getting\_Started\_Guide.html. Дата доступа: 06.07.2024.
5. Intro to JaCoCo [Электронный ресурс]. Режим доступа: https://www.baeldung.com/jacoco. Дата доступа: 06.07.2024.

**ПРИЛОЖЕНИЕ А**

WSDL-схема веб-сервиса

|  |
| --- |
| <wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://service.getinfo.com/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:ns1="http://schemas.xmlsoap.org/soap/http" name="ValidateInformationService" targetNamespace="http://service.getinfo.com/">  <wsdl:types>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://service.getinfo.com/" elementFormDefault="unqualified" targetNamespace="http://service.getinfo.com/" version="1.0">  <xs:element name="GetInformation" type="tns:GetInformation"/>  <xs:element name="GetInformationArguments" type="tns:getInformationArguments"/>  <xs:element name="GetInformationResponse" type="tns:GetInformationResponse"/>  <xs:element name="GetResponse" type="tns:getResponse"/>  <xs:complexType name="GetInformation">  <xs:sequence>  <xs:element minOccurs="0" name="GetInformationArguments" type="tns:getInformationArguments"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="getInformationArguments">  <xs:sequence>  <xs:element maxOccurs="unbounded" minOccurs="0" name="parameters" type="tns:parameters"/>  <xs:element minOccurs="0" name="password" type="xs:string"/>  <xs:element minOccurs="0" name="serviceId" type="xs:string"/>  <xs:element minOccurs="0" name="username" type="xs:string"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="parameters">  <xs:sequence>  <xs:element name="paramKey" type="xs:string"/>  <xs:element name="paramValue" type="xs:string"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="GetInformationResponse">  <xs:sequence>xs:element minOccurs="0" name="return" type="tns:getResponse"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="getResponse">  <xs:sequence>  <xs:element minOccurs="0" name="MSG" type="xs:string"/>  </xs:sequence>  </xs:complexType>  </xs:schema>  </wsdl:types>  <wsdl:message name="GetInformationResponse">  <wsdl:part element="tns:GetInformationResponse" name="parameters"> </wsdl:part>  </wsdl:message>  <wsdl:message name="GetInformation">  <wsdl:part element="tns:GetInformation" name="parameters"> </wsdl:part>  </wsdl:message>  <wsdl:portType name="ValidateInformation">  <wsdl:operation name="GetInformation">  <wsdl:input message="tns:GetInformation" name="GetInformation"> </wsdl:input>  <wsdl:output message="tns:GetInformationResponse" name="GetInformationResponse"> </wsdl:output>  </wsdl:operation>  </wsdl:portType>  <wsdl:binding name="ValidateInformationServiceSoapBinding" type="tns:ValidateInformation">  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>  <wsdl:operation name="GetInformation">  <soap:operation soapAction="" style="document"/>  <wsdl:input name="GetInformation">  <soap:body use="literal"/>  </wsdl:input>  <wsdl:output name="GetInformationResponse">  <soap:body use="literal"/>  </wsdl:output>  </wsdl:operation>  </wsdl:binding>  <wsdl:service name="ValidateInformationService">  <wsdl:port binding="tns:ValidateInformationServiceSoapBinding" name="ValidateInformationPort">  <soap:address location="http://localhost:5443/server-1.0-SNAPSHOT/ValidateInformation"/>  </wsdl:port>  </wsdl:service>  </wsdl:definitions> |

**ПРИЛОЖЕНИЕ Б**

Класс-валидатор параметров запроса

|  |
| --- |
| public class ValidationUtil {  public static List<ErrorDetail> validateArguments(GetInformationArguments arguments) {  List<ErrorDetail> errors = new ArrayList<>();  errors.addAll(validateRequiredFields(arguments));  errors.addAll(validateParameters(arguments.getParametersList()));  return errors;  }  private static List<ErrorDetail> validateRequiredFields(GetInformationArguments arguments) {  List<ErrorDetail> errors = new ArrayList<>();  if (!validateField(arguments.getPassword())) {  errors.add(ErrorRegistry.getError(ErrorKey.EMPTY\_PASSWORD));  }  if (!validateField(arguments.getUsername())) {  errors.add(ErrorRegistry.getError(ErrorKey.EMPTY\_USERNAME));  }  if (!validateField(arguments.getServiceId())) {  errors.add(ErrorRegistry.getError(ErrorKey.EMPTY\_SERVICEID));  }  return errors;  }  private static List<ErrorDetail> validateParameters(List<Parameters> parametersList) {  List<ErrorDetail> errors = new ArrayList<>();  final String CLIENT\_ID = "clientid";  if (parametersList == null || parametersList.size() < 2) {  errors.add(ErrorRegistry.getError(ErrorKey.INSUFFICIENT\_PARAMETERS));  } else {  boolean clientIdFound = false;  boolean clientIdUnique = true;  List<String> clientIdList = new ArrayList<>();  for (Parameters parameters : parametersList) {  if (!validateField(parameters.getParamKey())) {  errors.add(ErrorRegistry.getError(ErrorKey.EMPTY\_PARAM\_KEY));  }  if (!validateField(parameters.getParamValue())) {  errors.add(ErrorRegistry.getError(ErrorKey.EMPTY\_PARAM\_VALUE));  }  if (CLIENT\_ID.equals(parameters.getParamKey())) {  if (clientIdFound) {  clientIdUnique = false;  }  clientIdFound = true;  if (!validateClientId(parameters.getParamValue())) {  errors.add(ErrorRegistry.getError(ErrorKey.INVALID\_CLIENTID));  }  if (!isUniqueClientId(parameters.getParamValue())) {  errors.add(ErrorRegistry.getError(ErrorKey.NON\_UNIQUE\_CLIENT\_ID));  }  if (!clientIdList.contains(parameters.getParamValue())) {  clientIdList.add(parameters.getParamValue());  } else {  clientIdUnique = false;  }  }  }  if (!clientIdFound) {  errors.add(ErrorRegistry.getError(ErrorKey.NO\_CLIENT\_ID));  }  if (!clientIdUnique) {  errors.add(ErrorRegistry.getError(ErrorKey.MULTIPLE\_CLIENT\_IDS));  }  }  return errors;  }  private static boolean validateField(String field) {  return field != null && !field.isEmpty();  }  private static boolean validateClientId(String clientId) {  return clientId != null && clientId.matches("\\d{8}");  }  private static boolean isUniqueClientId(String clientId) {  Connection connection = null;  PreparedStatement statement = null;  ResultSet resultSet = null;  boolean isUnique = true;  try {  connection = SQLiteConnection.getConnection();  String query = "SELECT COUNT(\*) AS count FROM Clients WHERE clientid = ?";  statement = connection.prepareStatement(query);  statement.setString(1, clientId);  resultSet = statement.executeQuery();  if (resultSet.next()) {  int count = resultSet.getInt("count");  if (count > 0) {  isUnique = false;  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  if (resultSet != null) {  try {  resultSet.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  return isUnique;  }  } |

**ПРИЛОЖЕНИЕ В**

Класс с автотестами для основного запроса

|  |
| --- |
| public class ValidationUtilTest {  @Test  public void validateArguments\_AllFieldsValid() {  GetInformationArguments arguments = ValidationUtilTestHelper.createValidArguments();  List<ErrorDetail> errors = ValidationUtil.validateArguments(arguments);  assertTrue(errors.isEmpty());  }  @Test  public void validateArguments\_MissingPassword() {  GetInformationArguments arguments = ValidationUtilTestHelper.createValidArguments();  arguments.setPassword(null);  List<ErrorDetail> errors = ValidationUtil.validateArguments(arguments);  assertErrorCode(ErrorKey.EMPTY\_PASSWORD, errors);  }  @Test  public void validateArguments\_MissingUsername() {  GetInformationArguments arguments = ValidationUtilTestHelper.createValidArguments();  arguments.setUsername(null);  List<ErrorDetail> errors = ValidationUtil.validateArguments(arguments);  assertErrorCode(ErrorKey.EMPTY\_USERNAME, errors);  }  @Test  public void validateArguments\_MissingServiceId() {  GetInformationArguments arguments = ValidationUtilTestHelper.createValidArguments();  arguments.setServiceId(null);  List<ErrorDetail> errors = ValidationUtil.validateArguments(arguments);  assertErrorCode(ErrorKey.EMPTY\_SERVICEID, errors);  }  @Test  public void validateArguments\_InvalidClientId() {  GetInformationArguments arguments = ValidationUtilTestHelper.createValidArguments();  Parameters param1 = new Parameters();  param1.setParamKey("clientid");  param1.setParamValue("123d45678"); // INVALID VALUE (MUST BE 8 DIGITS)  arguments.getParametersList().add(param1);  List<ErrorDetail> errors = ValidationUtil.validateArguments(arguments);  assertErrorCode(ErrorKey.INVALID\_CLIENTID, errors);  }  @Test  public void validateArguments\_AnyParamValueExceptClientId() {  GetInformationArguments arguments = ValidationUtilTestHelper.createValidArguments();  Parameters param1 = new Parameters();  param1.setParamKey("text");  param1.setParamValue("12345dfdfd678987654321");  arguments.getParametersList().add(param1);  List<ErrorDetail> errors = ValidationUtil.validateArguments(arguments);  assertTrue(errors.isEmpty());  }  @Test  public void validateArguments\_MissingBothParametersBlocks() {  GetInformationArguments arguments = ValidationUtilTestHelper.createValidArguments();  arguments.setParametersList(null);  List<ErrorDetail> errors = ValidationUtil.validateArguments(arguments);  assertErrorCode(ErrorKey.INSUFFICIENT\_PARAMETERS, errors);  }  @Test  public void testHandleValidationErrors() {  List<ErrorDetail> errors = new ArrayList<>();  errors.add(ErrorRegistry.getError(ErrorKey.EMPTY\_PARAM\_KEY));  errors.add(ErrorRegistry.getError(ErrorKey.EMPTY\_SERVICEID));  try {  SOAPErrorUtil.handleValidationErrors(errors);  fail("Expected SOAPFaultException was not thrown");  } catch (SOAPFaultException e) {  assertEquals("Errors", e.getFault().getFaultCode());  assertEquals("Validation Errors", e.getFault().getFaultString());  Detail detail = e.getFault().getDetail();  NodeList errorElements = detail.getElementsByTagName("error");  assertEquals(2, errorElements.getLength());  }  }  private void assertErrorCode(ErrorKey expectedKey, List<ErrorDetail> errors) {  String expectedErrorCode = ErrorRegistry.getError(expectedKey).getCode();  assertEquals(expectedErrorCode, errors.get(0).getCode());  }  } |