Министерство науки и высшего образования РФ

**федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

**(Университет ИТМО)**

Факультет Программной инженерии и компьютерной техники

Образовательная программа СППО 2021

Специализация Программная инженерия

О Т Ч Е Т

Об учебной практике

**Тема задания:** Автоматизация проверки шаблона ВКР

**Обучающийся:** *Чжоу Хунсян, Р34131*

Руководитель практики от университета: *Маркина Т.А., к.т.н., старший преподаватель*

Практика пройдена с оценкой \_\_\_\_

Подписи членов комиссии:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.И.О.

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.И.О.

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.И.О.

(подпись)

Дата 04.03.2025

Санкт-Петербург

2025

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc192004587)

[Основная часть 5](#_Toc192004588)

[Задача 2-го этапа **错误!未定义书签。**](#_Toc192004589)

[Задача 3-го этапа **错误!未定义书签。**](#_Toc192004590)

[Задача 4-го этапа **错误!未定义书签。**](#_Toc192004591)

[Заключение 10](#_Toc192004592)

[Примечания 11](#_Toc192004593)

# ВВЕДЕНИЕ

Производственная практика – заключительный и важнейший этап образовательного процесса, направленный на проверку и закрепление компетенций обучающегося, полученных в процессе академического обучение и учебной практики путём работы над настоящим проектом в условиях, не отличающихся от обычного трудового распорядка компании, принимающей практику.

Целью производственной практики является демонстрация учащимся того, что он способен выполнять работу в рамках своей специальности. Проверка навыков осуществляется через выполнение индивидуального задания (Таблица 1).

|  |  |  |
| --- | --- | --- |
| **Порядковый № этапа** | **Наименование этапа** | **Задание этапа** |
| 1 | Исправить существующие ошибки и недочёты | Инструктаж обучающегося по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка |
| 2 | Aнализ требований и проектирование системы | Задание: Изучить требования к шаблону выпускной квалификационной работы, определить перечень параметров для автоматической проверки и составить техническое задание. Результат: Готовый документ, содержащий технические требования, архитектурную схему системы и описание её модулей. |
| 3 | Разработка и реализация системы | Задание: В первую очередь разработать модуль проверки шаблона, который анализирует структуру документа, его форматирование и наличие обязательных элементов. Затем создать модуль формирования отчёта, содержащего список выявленных несоответствий и рекомендации по их исправлению. Дополнительно реализовать модуль сбора статистики (частоту ошибок). Результат: Рабочая система, включающая модули проверки шаблона, формирования отчёта и сбора статистики. Разрfботанный модуль должен быть самостоятельным приложением и работать на Windows 11 |
| 4 | Тестирование и исправление ошибок | Задание: После завершения разработки провести ручное тестирование системы, выявить возможные ошибки, устранить их и оптимизировать работу системы. Результат: Исправленная, оптимизированная и корректно работающая система, полностью соответствующая техническим требованиям. |
| 5 | Оформление отчётных документов в соответствии с требованиями | 1. Должно быть подробное описание выполнения задач по этапам. Результаты задания необходимо разместить в приложения. 2. Оформление отчёта должно быть выполнено в соответствии с методическим пособием (https://books.ifmo.ru/file/pdf/2622.pdf) 3. Структура документа: титульный лист, введение, основная часть, заключение, примечания. 4. В основной части подробно описывается выполнение задач 2-4 этапов, в приложении помещаются результаты данных этапов. 5. Отчёт необходимо подгрузить в модуле практика как "письменный отчёт" |
| 6 | Получить отзыв руководителя практики | Получить отзыв с оценкой у руководителя практики в модуле Практики |

# ОСНОВНАЯ ЧАСТЬ

## Этап 2. Анализ требований и проектирование системы

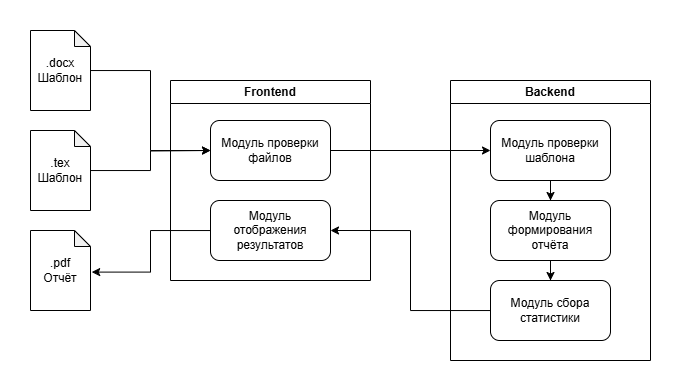
На данном этапе была проведена работа по анализу методических требований к оформлению ВКР. Изучены документы, регламентирующие структуру, форматирование, использование стилей, межстрочный интервал, шрифт и другие параметры. На основе анализа был составлен перечень параметров, подлежащих автоматической проверке, включая:

* Наличие всех обязательных разделов (введение, основная часть, заключение, список литературы и пр.)
* Используемые стили и соответствие их требованиям
* Размер и тип шрифта (например, Times New Roman, 14 пт.)
* Выравнивание, поля, отступы
* Корректное оформление заголовков

На основании проведённого анализа было составлено техническое задание на разработку системы. В нём были описаны цели и функции проекта, сформулированы функциональные и нефункциональные требования. Также была спроектирована архитектура системы, включающая следующие основные модули:

* Модуль проверки шаблона;
* Модуль формирования отчёта;
* Модуль сбора статистики.

Схема архитектуры и техническое задание представлены в Рисунке 1.



**Уровень пользовательского интерфейса:**

* Предоставляет интерфейс загрузки файлов и функцию проверки файлов.
* Отображает результаты проверки, отчеты в формате PDF и статистику ошибок.
* Взаимодействуйте с модулями на сервере для получения отчетов и статистики.

**Уровень внутренней логики:**

* Модуль проверки шаблонов – отвечает за анализ документов и проверку формата.
* Модуль генерации отчетов – создает отчеты с подробной информацией и рекомендациями.
* Модуль сбора статистики – собирает и хранит статистику, такую как частота ошибок и количество предложенных исправлений.

## Этап 3. Разработка и реализация системы

В первую очередь был разработан модуль проверки шаблона. Он реализует алгоритмы анализа структуры документа .docx и .tex, проверяет соответствие форматирования заданным правилам, определяет наличие обязательных элементов и ошибок.

Сначала мы реализовали бэкэнд для анализа файлов .docx с использованием Java Spring Boot.

Во-первых, мы используем Java Spring Boot для реализации бэкэнда для анализа файлов .docx.

Здесь мы используем Apache POI XWPF для разбора и анализа формата .docx следующим образом:

|  |
| --- |
| public ValidationResponse validateTemplate(MultipartFile file) throws IOException {  result.clear();  errorCountMap.clear();  // Initialize result storage  Set<String> checkedParagraphs = new HashSet<>();  // To record reported errors  Set<String> reportedFontErrors = new HashSet<>();  Set<String> reportedFontSizeErrors = new HashSet<>();  Set<String> reportedBoldErrors = new HashSet<>();  Set<String> reportedAlignmentErrors = new HashSet<>();  int totalParagraphs = 0;  try (XWPFDocument doc = new XWPFDocument(file.getInputStream())) {  // check contents  // containsTableOfContents(doc);  // To store text of all paragraphs  StringBuilder allParagraphText = new StringBuilder();  for (XWPFParagraph paragraph : doc.getParagraphs()) {  String paragraphText = paragraph.getText().trim();  if (paragraphText.isEmpty() || checkedParagraphs.contains(paragraphText)) continue;  checkedParagraphs.add(paragraphText);  totalParagraphs++;  allParagraphText.append(paragraphText).append("\n");  checkFont(paragraph, paragraphText, result, errorCountMap, reportedFontErrors);  checkFontSize(paragraph, paragraphText, result, errorCountMap, reportedFontSizeErrors);  checkBoldText(doc, getSectionTextList(), paragraph, paragraphText, result, errorCountMap, reportedBoldErrors);  }  checkIsExitInDocument(allParagraphText.toString(), getSectionTextList(), reportedBoldErrors);  // Statistics of results  ValidationStats stats = new ValidationStats();  stats.setErrorTypeCount(errorCountMap);  stats.setTotalParagraphs(totalParagraphs);  stats.setTotalErrors(result.size());  // If no error  if (result.isEmpty()) {  result.add(new ValidationMessage(  "NoErrors",  "Нет ошибок",  "Документ был проверен, ошибок не обнаружено.",  "OK"  ));  }  return new ValidationResponse(result, stats);  }  } |

Во-вторых, мы используем Python для реализации бэкэнда для анализа файлов .tex.

|  |
| --- |
| @app.route('/api/validate/latex', methods=['POST'])  def validate\_latex():  file = request.files.get('file')  if not file or not file.filename.endswith('.tex'):  return jsonify({"error": "Please upload a .tex file"}), 400  content = file.read().decode('utf-8')  response = validate\_tex(content)  return jsonify(json.loads(response.to\_json())) |

Наконец, мы реализовали графический интерфейс с использованием Java FX, включает следующие компонеты:

1. **Кнопка загрузки файлов пользователя**: используется для загрузки файлов шаблонов.
2. **Список проблем**: Перечисляет проблемы форматирования в шаблоне, включая следующие:

* Тип ошибки,
* Описание ошибки,
* Предложения по модификации,
* Контекст ошибки

1. **Статистика вопросов**: подсчитайте количество различных типов вопросов.
2. **Загрузка отчета в формате PDF**: используется для загрузки созданного отчета о проблемах в формате PDF.

## Этап 4. Тестирование и исправление ошибок

После реализации системы было проведено ручное тестирование на различных шаблонах ВКР. Были выявлены и устранены следующие проблемы:

* При формировании pdf-отчетов возникает ошибка в русской кодировке
* Проблема синтаксиса новой строки в контексте каталога .tex
* Проблема преобразования размера шрифта .docx
* Apache POI XWPF неправильно оценил наследование стилей документов .docx

По итогам оптимизации система стала работать стабильнее, а проверка выполняется быстрее. Все выявленные баги были устранены, а работа модулей приведена в соответствие с техническим заданием.

# ЗАКЛЮЧЕНИЕ

В ходе прохождения практики была успешно реализована система автоматической проверки шаблонов ВКР, включающая три взаимосвязанных модуля в бэкэнде и интерфейс пользователя фронэед на Java. Программа позволяет существенно упростить процесс подготовки ВКР, повысить его точность и соответствие требованиям. Полученные результаты свидетельствуют о практической значимости проекта.

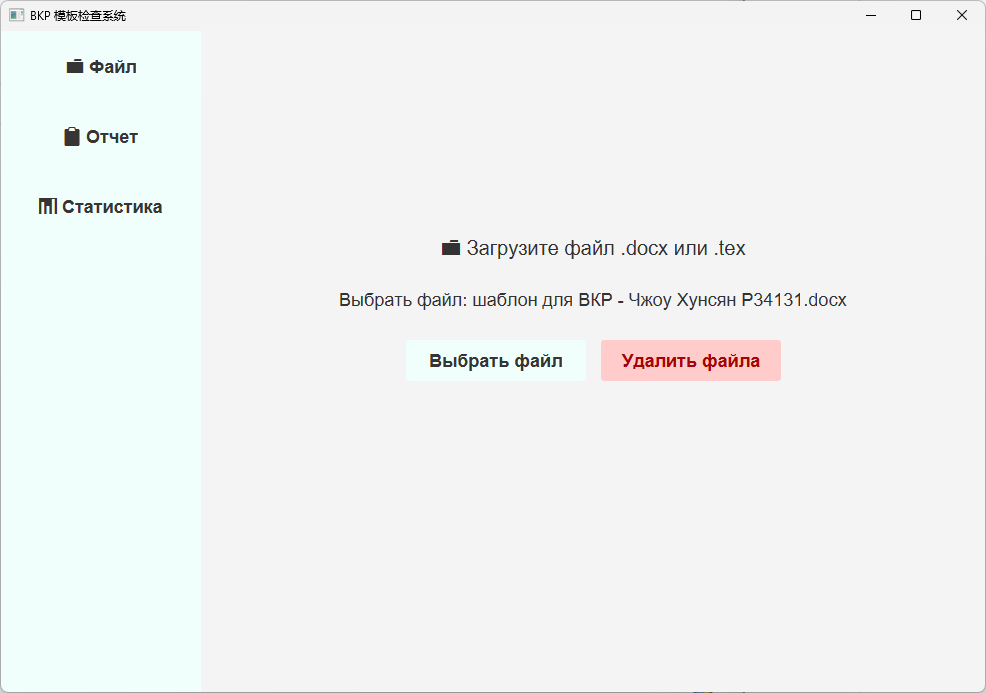
В дальнейшем систему можно расширить за счёт поддержки других форматов документов и интеграции с системами антиплагиата.

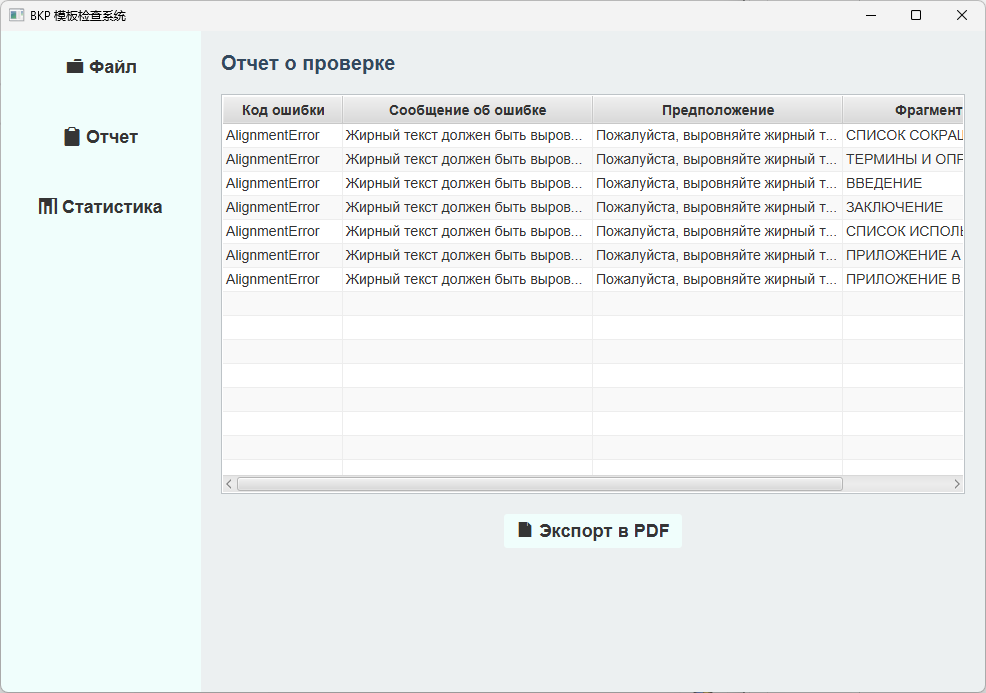
# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

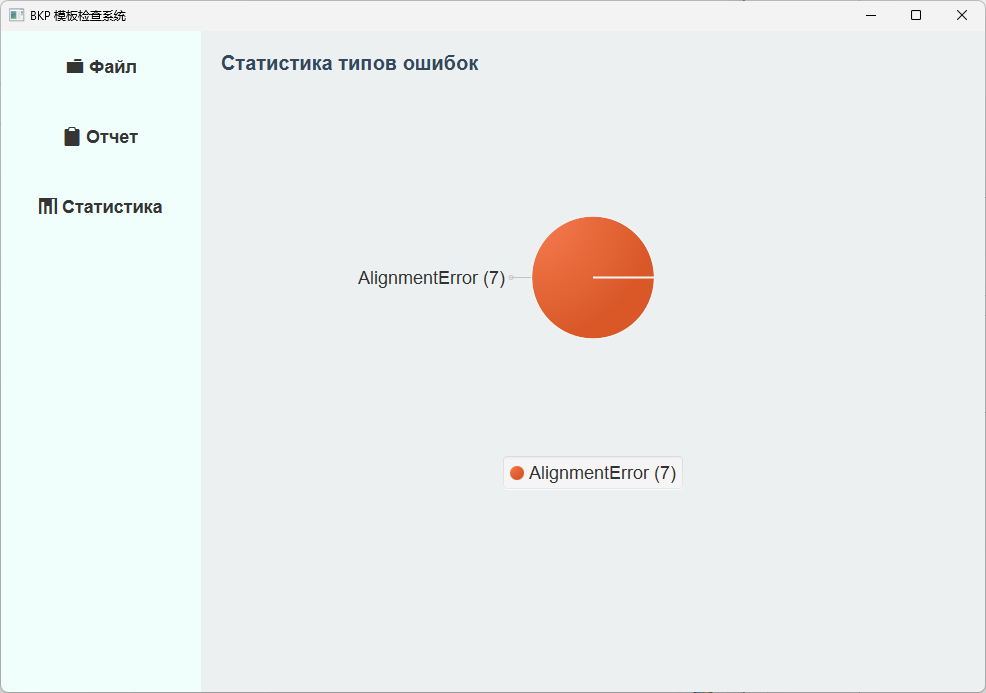
1. Маркина Т.А., Пенской А.В., Штенников Д.Г., Авксентьева Е.Ю., Ильина А.Г. Производственная практика магистрантов: организация и проведение [Электронный ресурс]. – СПб.: Университет ИТМО, 2021. – Режим доступа: <https://books.ifmo.ru/file/pdf/2622.pdf>
2. JavaFX. Официальная документация [Электронный ресурс]. – Режим доступа: <https://openjfx.io>
3. LaTeX Project. Официальный сайт [Электронный ресурс]. – Режим доступа: <https://www.latex-project.org>
4. Apache POI – Java API for Microsoft Documents [Электронный ресурс]. – Режим доступа: <https://poi.apache.org>
5. Spring Boot – Spring Framework Project [Электронный ресурс]. – Режим доступа: <https://spring.io/projects/spring-boot>
6. Python Software Foundation. Официальный сайт Python [Электронный ресурс]. – Режим доступа: <https://www.python.org>
7. Flask – Lightweight Python Web Framework [Электронный ресурс]. – Режим доступа: <https://flask.palletsprojects.com>

# ПРИМЕЧАНИЯ

Скриншоты работающего приложения:







Ссылка на приложение

<https://github.com/mkdirP/Practice_project>

Ссылка на техническое задание

<https://github.com/Tolia-GH/ITMO-PE/blob/main/Practice/%D0%A2%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5%20%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%20Practice%20.pdf>