МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

на тему

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА ДЛЯ ФОРМИРОВАНИЯ ОБРАЗОВАТЕЛЬНЫХ МАТЕРИАЛОВ

Студент _____ 22.05.26 Юмаева Эльмира Яткяровна
(подпись, дата) (ФИО полностью)

Руководитель _____ 22.05.26 Карамышева Н. С.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ 22.05.26 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент Начальник УИТ _____ 22.05.26 Севостьянов В.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 № 14)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 04 июня 2026 № 4)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____ М.А. Митрохин
«20» сентября 2024 г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

1. Студент Юмаева Эльмира Яткяровна гр. 24ВВИм1 факультета ВТ

<Фамилия, имя, отчество полностью>

2.Тема работы Интеллектуальная система для формирования образовательных материалов

Тема утверждена приказом ПГУ № 920/о от "20" сентября 2024 г.

3. Руководитель магистерской диссертации Карамышева Надежда Сергеевна

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать интеллектуальную систему, способную автоматически формировать учебные планы и генерировать рабочие программы дисциплин (РПД) на основе анализа федеральных государственных образовательных стандартов (ФГОС) и профессиональных стандартов.
Исходные данные: тексты ФГОС ВО по направлениям подготовки (09.03.01, 03.04.01, 54.05.02); тексты профессиональных стандартов (06.001 «Программист», 06.003 «Архитектор ПО», 04.002 «Специалист по техническим процессам художественной деятельности»).; API облачной генеративной модели YandexGPT; технологии: Python, Streamlit, библиотеки NLP (pandas, openpyxl, python-docx)

5. Перечень подлежащих разработке вопросов 1.Провести анализ существующих методов автоматизации формирования образовательных программ и обзор аналогов (MMIS, 1С:Университет, LMS). 2. Исследовать методы NLP и возможности больших языковых моделей (LLM) для анализа нормативной документации. 3. Разработать архитектуру интеллектуальной системы, включающую модули анализа ФГОС, обработки профстандартов, генерации дисциплин и формирования учебного плана. 4.Реализовать механизм двухступенчатой генерации образовательных материалов с использованием YandexGPT и JSON-schema-driven prompting. 5. Разработать модуль автоматической генерации рабочих программ дисциплин (РПД) и формирования DOCX-документов. 6. Создать веб-интерфейс на

базе Streamlit для загрузки документов, генерации учебного плана и взаимодействия с ИИ-консультантом.7. Провести экспериментальное исследование и оценку эффективности разработанной системы на различных направлениях подготовки.

6. Календарный график выполнения работы

№ п п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области(ФГОС, профстандарты, обзор существующих систем MMIS, 1С, обзор методов NLP)	15%	20.09.2024 – 15.11.2024	
2	Проектирование архитектуры системы (Разработка структуры модулей, выбор технологического стека)	10%	16.11.2024 - 25.12.2024	
3	Реализация модулей анализа ФГОС и профессиональных стандартов (Извлечение компетенций, обработка трудовых функций, OCR, регулярные выражения)	15%	26.12.2024 – 20.02.2025	
4	Интеграция YandexGPT и разработка механизма генерации дисциплин (Реализация двухступенчатой генерации, prompt-инжиниринг, верификация JSON-ответов)	20%	21.02.2025 – 15.09.2025	
5	Разработка модуля генерации рабочих программ дисциплин (РПД) и формирование DOCX-документов (Интеграция с python-docx, создание титульных листов, таблиц)	20%	16.09.2025 – 20.12.2025	
6	Разработка веб-интерфейса (Streamlit) и ИИ-консультанта (Создание вкладок, обработка пользовательских запросов)	10%	21.12.2025 – 15.02.2026	
7	Экспериментальное исследование и тестирование (Проверка на направлениях 09.03.01, 03.04.01, 54.05.02, сравнение с реальными РПД, оценка эффективности)	10%	16.02.2026 – 30.03.2026	
8	Оформление пояснительной записки, подготовка презентации и доклада	10%	31.03.2026 – 21.05.2026	

Дата выдачи задания "20" сентября 2024г.

Руководитель магистерской диссертации _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Л. Р. Фомина
(подпись, дата) 26.05.2025

РЕФЕРАТ

Магистерская работа содержит 90 листов, 29 рисунков, 10 таблиц, 33 источника, 5 приложений.

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, LLM, YANDEXGPT, АВТОМАТИЗАЦИЯ ОБРАЗОВАТЕЛЬНЫХ ПРОГРАММ, ФГОС ВО 3++, ОПОП, УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ГЕНЕРАЦИЯ ДОКУМЕНТОВ, JSON-СХЕМЫ, PROMPT ENGINEERING, STREAMLIT

Цель работы – разработка и развитие интеллектуальной системы, способной автоматически формировать учебные планы и генерировать РПД на основе ФГОС и профессиональных стандартов.

Объект исследования: процессы проектирования и автоматизации разработки образовательных программ в высшем образовании.

Предмет исследования: методы и программные средства автоматизированного анализа нормативных документов (ФГОС, профстандарты) и формирования учебно-методических материалов.

В работе исследованы возможности применения больших языковых моделей для автоматизации подготовки образовательной документации. Разработана система на основе YandexGPT, обеспечивающая генерацию учебных планов, рабочих программ дисциплин. Для повышения корректности результатов использованы JSON-схемы, механизмы валидации и повторной генерации ответов.

Практическая значимость работы заключается в сокращении времени подготовки образовательной документации и снижении объема рутинной работы методистов. Разработанная система может применяться образовательными организациями ВО для автоматизации учебно-методической деятельности и развития цифровых образовательных технологий.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

ИИ – искусственный интеллект

ОПК – общепрофессиональные компетенции

ПК – профессиональные компетенции

РПД – рабочая программа дисциплины

УК – универсальные компетенции

УММ – учебно-методические материалы

ФГОС – федеральный государственный образовательный стандарт

ФГОС ВО – федеральный государственный образовательный стандарт высшего образования

API – Application Programming Interface (интерфейс программирования приложений)

DOCX – формат электронных документов Microsoft Word

JSON – JavaScript Object Notation (текстовый формат обмена данными)

LMS – Learning Management System (система управления обучением)

MMIS – программный комплекс автоматизации образовательного процесса

NLP – Natural Language Processing (обработка естественного языка)

OCR – Optical Character Recognition (оптическое распознавание текста)

OpenAI – компания-разработчик языковых моделей искусственного интеллекта

Python – язык программирования Python

Streamlit – фреймворк для разработки веб-приложений на Python

YandexGPT – генеративная языковая модель компании Яндекс

ВВЕДЕНИЕ

На фоне цифровизации образования возникает необходимость автоматизации процессов подготовки учебно-методической документации. Формирование учебных планов и рабочих программ дисциплин является трудоёмкой задачей, требующей строгого соответствия федеральным государственным образовательным стандартам высшего образования (ФГОС ВО), а также внутренним нормативным документам образовательной организации. Автоматизация этих процессов способна снизить трудозатраты, уменьшить число ошибок и ускорить актуализацию программ при изменениях в документах.

В рамках данной работы рассматривается разработка интеллектуальной системы, которая автоматически анализирует тексты ФГОС и профессиональных стандартов, формирует перечень учебных дисциплин, распределяет их по семестрам, сопоставляет компетенции и трудовые функции, а также генерирует рабочие программы дисциплин (РПД) в формате DOCX. Система реализована на языке Python с использованием Streamlit, методов NLP, OCR-технологий и облачной модели YandexGPT.

Гипотеза исследования: использование больших языковых моделей и методов обработки естественного языка при анализе ФГОС и профессиональных стандартов позволит сократить время формирования учебных планов и рабочих программ дисциплин не менее чем в 10 раз по сравнению с полностью ручной подготовкой документов при сохранении полноты структуры документов и корректности извлеченных компетенций.

Научная цель исследования: оценить применимость и эффективность методов NLP и генеративных моделей для автоматического проектирования образовательных программ и подготовки рабочих программ дисциплин.

Цель работы – разработка и развитие интеллектуальной системы, способной автоматически формировать учебные планы и генерировать рабочие программы дисциплин (РПД) на основе ФГОС и профессиональных стандартов.

Объект исследования: процессы проектирования и автоматизации разработки образовательных программ в высшем образовании.

Предмет исследования: методы и программные средства автоматизированного анализа нормативных документов (ФГОС, профстандарты) и формирования учебно-методических материалов.

Задачи исследования:

– проанализировать существующие подходы и программные решения для автоматизации формирования учебных планов,

– исследовать методы NLP и возможности облачных языковых моделей для анализа нормативных документов,

– разработать архитектуру интеллектуальной системы, включающей модули анализа ФГОС, анализа профстандартов, генерации дисциплин и формирования учебного плана,

– реализовать модуль автоматической генерации рабочих программ дисциплин (РПД) с использованием YandexGPT и python-docx,

– разработать веб-интерфейс для взаимодействия пользователя с системой,

– провести тестирование и экспериментальную оценку работы системы,

Методы исследования:

– методы обработки естественного языка (предобработка текста, регулярные выражения, морфологический анализ),

– классические алгоритмы машинного обучения для классификации и сопоставления (TF-IDF, SVM),

– генеративные подходы для формирования содержательной части РПД,

– эвристические правила для распределения дисциплин по семестрам, а также экспертная оценка результатов.

Научная новизна заключается в разработке методики двухступенчатой генерации образовательных материалов, при которой облегчённая языковая модель используется для извлечения и первичного структурирования данных,

а полная модель – для формирования содержательной части учебных планов и рабочих программ дисциплин.

Практическая значимость: разработанная система может быть использована методистами и преподавателями для ускорения подготовки учебной документации, повышения её качества и обеспечения соответствия нормативным требованиям; внедрение системы позволяет сократить часы на подготовку РПД и упростить актуализацию учебных планов при изменениях ФГОС.

1 Анализ методов автоматизации формирования образовательных программ

1.1 ФГОС и профстандарты как основа проектирования образовательных программ

Федеральные государственные образовательные стандарты (ФГОС) являются ключевым нормативным документом, определяющим требования к структуре, результатам и условиям реализации образовательных программ высшего образования. ФГОС ВО разрабатываются по профессиям, специальностям и направлениям подготовки по соответствующим уровням профессионального образования и представляют собой совокупность требований, обязательных при реализации основных профессиональных образовательных программ (ОПОП) высшего образования образовательными организациями, имеющими государственную аккредитацию [1]. В ФГОС сформулированы общепрофессиональные, универсальные и профессиональные компетенции, которыми должен обладать выпускник, а также установлены индикаторы их достижения. Далее рассмотрим каждый компонент подробнее.

Профессиональный стандарт – характеристика квалификации, необходимой работнику для осуществления определенного вида профессиональной деятельности, в том числе выполнения определенной трудовой функции [2]. Профессиональные стандарты дополняют ФГОС, описывая трудовые функции, необходимые знания, умения и действия специалиста в реальной профессиональной деятельности. Трудовые функции в процессе проектирования учебных планов позволяют формировать дисциплины, связанные с профессиональными задачами выпускника.

Анализ ФГОС и профстандартов представляет собой трудоёмкий процесс - документы содержат большой объёмы текстовой информации, сложные формулировки компетенций и много взаимосвязей между требованиями. Ручная обработка таких документов требует высокой квалификации методистов и занимает значительное время, особенно при

регулярном обновлении стандартов. Появляется необходимость автоматизации анализа нормативных документов и разработки инструментов, способных извлекать компетенции, сопоставлять их с трудовыми функциями и формировать структуру учебного плана. Взаимосвязь ФГОС, профессиональных стандартов и образовательной программы представлена на рисунке 1.

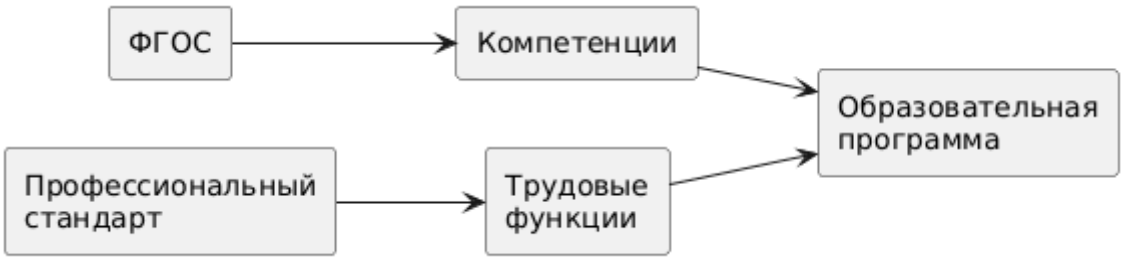


Рисунок 1 – Взаимосвязь ФГОС, профстандартов и образовательной программы

Использование технологий ИИ позволяет существенно упростить этот процесс. Алгоритмы машинного обучения, методы обработки естественного языка (NLP) и современные языковые модели обеспечивают автоматическое извлечение компетенций, классификацию текстов и формирование дисциплин на основе нормативных требований. Интеграция ИИ-моделей в процесс проектирования образовательных программ делает возможным создание интеллектуальных систем, которые анализируют ФГОС и профстандарты, а также формируют учебные планы и обеспечивают их соответствие нормативной базе.

Таким образом, ФГОС и профессиональные стандарты выступают основой для построения образовательных программ, а автоматизация их анализа является ключевым направлением развития современных цифровых инструментов в сфере образования.

1.2 Проблемы ручного формирования учебных планов

Разработка учебных планов в российских вузах до сих пор требует ручной обработки. Методистам приходится просматривать десятки страниц ФГОС, профессиональных стандартов, локальных нормативных документов и

уже существующих рабочих программ дисциплин. Причём задача не сводится к простому переносу данных из одного документа в другой. Нужно сопоставить компетенции с дисциплинами, распределить модули по семестрам, проверить объём часов и формы контроля, а затем согласовать всё это между кафедрой, преподавателями и методическим отделом.

Основные задачи управления процессом обучения, такие как составление учебных планов, формирование учебных поручений, мониторинг качества подготовки специалистов, необходимо решать с учётом требований ФГОС ВО 3++ [3].

Подготовка одного учебного плана может занимать недели, а иногда и месяцы. Особенно это заметно при переходе на обновлённые версии ФГОС или запуске новых образовательных программ. Значительная часть времени уходит именно на работу с нормативными документами: поиск компетенций, проверку индикаторов достижения и сопоставление требований между различными разделами стандарта. Все это напрямую связано с увеличением объема и усложнением системы документооборота, что приводит к дополнительной нагрузке на преподавателей и руководителей подразделений [3].

Проблема ещё и в том, что существующие системы автоматизации помогают в основном хранить и редактировать уже готовые учебные планы. Например, LMS-платформы или решения вроде «1С:Университет» не анализируют сами нормативные документы и не извлекают из них компетенции автоматически. В результате большая часть работы всё равно остаётся ручной. Не всегда эти решения адаптируемы под узкоспециализированные нужды вуза и не все из них адаптированы под новые стандарты [3].

Даже шаблонные решения с готовыми учебными планами помогают ограниченно. Их приходится адаптировать под конкретное направление подготовки, перераспределять дисциплины и повторно проверять соответствие требованиям ФГОС. При этом такие системы работают скорее

как средство хранения документов, чем как инструмент интеллектуального анализа содержания образовательной программы.

Этот процесс является очень трудоемким и для того, чтобы его упростить, необходима система автоматизации процесса подготовки учебно-методических материалов (УММ) [3].

В результате при ручной разработке учебных планов возникают несколько типичных проблем:

– большие временные затраты на подготовку документации,

– постоянная работа с объёмными текстами нормативных документов,

– субъективная интерпретация требований ФГОС;

– отсутствие автоматического извлечения компетенций и структурирования данных;

– ошибки при распределении дисциплин и компетенций;

– многочисленные согласования между участниками образовательного процесса.

Именно эти проблемы стали причиной разработки интеллектуальной системы, способной автоматизировать часть работы с нормативной документацией и сократить объём ручной подготовки учебных планов.

1.3. Обзор существующих систем автоматизации (MMIS, 1C, LMS)

На сегодняшний день в российских вузах используется достаточно много программных решений для работы с учебными планами и учебной документацией. Но почти все они решают в первую очередь организационные и административные задачи. Они помогают хранить данные, формировать отчёты, вести документацию, но не анализируют сами нормативные документы и не помогают проектировать образовательную программу на содержательном уровне.

1.3.1 MMIS «Планы»

Одной из наиболее распространённых систем является MMIS «Планы». Система предназначена для автоматизации планирования учебного процесса и поддерживает формирование учебных планов в формате, совместимом с

процедурами государственной аккредитации. В описании системы указано, что она позволяет создать «единую систему автоматизированного планирования учебного процесса» [4].

Учебные планы (УП), создаваемые в ИС Планы, полностью совместимы с форматом, используемым в процедуре государственной аккредитации [4].

Система включает встроенные шаблоны ФГОС и готовые структуры учебных планов. Это упрощает подготовку документации и сокращает объём ручного ввода данных.

Система также включает в себя комплект формализованных ФГОС, которые можно использовать для создания на их основе учебных планов и проверки их качества [4].

Но на этом возможности системы фактически заканчиваются. MMIS «Планы» не анализирует текст ФГОС, не извлекает компетенции автоматически и не умеет формировать структуру дисциплин на основе содержимого нормативных документов. Методисту всё равно приходится вручную распределять дисциплины, проверять компетенции и корректировать структуру плана.

Кроме того, система работает в основном по шаблонному принципу. Если появляется новое направление подготовки или изменяется структура ФГОС, значительную часть работы приходится переделывать вручную. Диалогового взаимодействия с пользователем или интеллектуальной генерации контента в системе нет.

1.3.2 1С:Университет

Комплекс «1С:Университет» широко используется в российских образовательных организациях для ведения учебной документации и сопровождения образовательного процесса. Система поддерживает хранение учебных планов, формирование отчётности, согласование документов и работу с нагрузкой преподавателей.

Формирование структуры дисциплин, обоснование связей между модулями и их содержательное наполнение остаются трудоемкими операциями [5].

Функционально это уже крупная информационная система уровня университета. Но её основная задача - учёт и управление данными. Анализ содержимого нормативных документов в базовом функционале отсутствует.

Система не поддерживает автоматический семантический анализ ФГОС и профессиональных стандартов, не извлекает компетенции из документов и не генерирует предварительную структуру образовательной программы. Пользователь по-прежнему сам определяет состав дисциплин и их взаимосвязи.

1.3.3 LMS-платформы (Moodle, OpenEdu, СберКласс)

LMS-системы ориентированы уже не на проектирование учебных планов, а на организацию самого учебного процесса. Moodle, OpenEdu, СберКласс и аналогичные платформы позволяют размещать учебные материалы, создавать курсы, проводить тестирование и контролировать успеваемость студентов [16].

Для обучения это удобно. Но такие платформы работают уже с готовыми дисциплинами и готовыми программами.

Если методисту нужно сформировать новый учебный план, LMS здесь почти не помогает. В неё нельзя загрузить ФГОС и получить автоматически сформированный перечень компетенций или дисциплин. Система не предлагает структуру программы и не анализирует связи между предметами.

«Публикация учебных материалов» и «предоставление информации об учебном процессе» являются основными функциями подобных систем [6], однако интеллектуальная генерация образовательного контента в них отсутствует.

По сути, LMS остаётся средой хранения и публикации учебных материалов, а не инструментом интеллектуального проектирования образовательных программ.

Сравнительный анализ функциональных возможностей современных систем, предназначенных для автоматизации образовательного процесса, наглядно систематизирован и представлен в таблице 1. Отмечены различия готовых систем и разработанной в ходе магистерской диссертации системы для формирования образовательных материалов.

Таблица 1 – Сравнение возможностей систем

Функция	MMIS	1C	LMS	Разработанная система
Хранение учебных планов	+	+	+	–
Анализ ФГОС	–	–	–	+
Извлечение компетенций	–	–	–	+
Генерация дисциплин	–	–	–	+
Работа с LLM	–	–	–	+
Генерация РПД	–	–	–	+

1.4 Методы NLP и ИИ для анализа нормативных документов

Методы обработки естественного языка (Natural Language Processing, NLP) позволяют автоматически анализировать текстовые документы, извлекать из них структурированную информацию и находить смысловые связи между различными фрагментами текста. Для образовательной документации это особенно важно, потому что ФГОС и профессиональные стандарты содержат большие объёмы слабо структурированного текста, который методистам приходится разбирать вручную.

Автоматическая обработка текстов на естественном языке и компьютерная лингвистика рассматриваются как одно из направлений интеллектуального анализа текстовой информации. При составлении учебного плана это дает возможность автоматически выделять компетенции, искать ключевые сущности в тексте и преобразовывать нормативные документы для дальнейшей обработки системой.

На первом этапе выполняется предобработка нормативных документов. Система удаляет служебные символы, потом приводит текст к единому регистру, выполняет токенизацию и лемматизацию, это позволяет уменьшить количество различий между словами и упростить последующий анализ текста. Общая схема обработки нормативных документов средствами NLP представлена на рисунке 2.

Рисунок 2 – Схема NLP-обработки нормативных документов

После предварительной обработки выполняется извлечение структурных элементов документа. Для поиска кодов компетенций, трудовых функций и других сущностей используются регулярные выражения (RegEx). С их помощью система автоматически находит конструкции вида УК-1, ОПК-3 или ПК-5 и формирует структурированный список компетенций.

Для повышения точности извлечения сущностей дополнительно применяются методы Named Entity Recognition (NER). Они позволяют автоматически выделять внутри текста коды направлений подготовки, компетенции и трудовые функции.

После извлечения сущностей выполняется сопоставление дисциплин с компетенциями и трудовыми функциями. В качестве базового подхода рассматривались классические методы обработки текста, включая TF-IDF-векторизацию и алгоритмы тематического анализа документов. Такие методы позволяют определять тематическую близость между описанием дисциплины и формулировками компетенций.

Однако классические статистические методы плохо работают со сложным контекстом. Нормативные документы часто содержат длинные формулировки, пересекающиеся определения и смысловые зависимости, которые сложно анализировать только на основе статистического сходства текста. В ходе исследования было установлено, что методы, основанные

исключительно на TF-IDF и регулярных выражениях, недостаточно эффективно учитывают взаимосвязи между компетенциями, трудовыми функциями и дисциплинами.

Именно поэтому в последние годы активно используются большие языковые модели (Large Language Models, LLM). Они способны учитывать контекст документа и формировать связный текст на естественном языке. Развитие подходов к обработке образовательных документов и переход от шаблонных алгоритмов к большим языковым моделям представлены на рисунке 3.

Рисунок 3 – Эволюция методов обработки образовательных документов

Современные методы обработки текста используют различные подходы к представлению и анализу семантики текста. Большую часть подходов можно разделить на три подмножества. В одном используют гипотезу дистрибутивной семантики, в другом - информацию из графовых баз знаний, а в третьем - анализируют лексико-синтаксические шаблоны в документах [7].

В рамках данной работы используется YandexGPT. Модель применяется для генерации дисциплин, тематических планов и рабочих программ дисциплин на основе текста нормативной документации.

Одной из важных особенностей современных LLM является возможность получения структурированного ответа в формате JSON. Это позволяет автоматически передавать результаты генерации между программными модулями системы без дополнительной ручной обработки.

Для повышения точности извлечения сущностей в системе также применяются методы Named Entity Recognition (NER). Они позволяют автоматически выделять внутри текста компетенции, коды направлений подготовки и трудовые функции.

Классические методы обработки текста, основанные на TF-IDF, регулярных выражениях и статистических моделях, хорошо подходят для извлечения структурированных элементов документа, однако плохо учитывают сложные смысловые связи между различными разделами нормативной документации. Большие языковые модели способны анализировать контекст документа целиком и формировать более связные результаты генерации. Это делает LLM более подходящими для задач автоматического формирования дисциплин, тематических планов и рабочих программ дисциплин.

Несмотря на высокое качество генерации, большие языковые модели не гарантируют абсолютную корректность результата, модель может нарушать структуру ответа, генерировать нерелевантные дисциплины или некорректно интерпретировать отдельные формулировки нормативных документов. Поэтому результаты генерации требуют дополнительной проверки и экспертного контроля со стороны методистов.

1.5 Эволюция подходов и этапов разработки системы

Разработка интеллектуальной системы автоматизации образовательных материалов происходила в несколько этапов. Система постепенно менялась от одной научно-исследовательской работы к другой, сначала это были простые алгоритмы обработки текста, позже появились методы NLP, а затем генеративные модели и автоматическая генерация РПД.

На первом этапе основное внимание уделялось анализу структуры ФГОС и профессиональных стандартов. Нужно было понять, как вообще устроены нормативные документы: где располагаются компетенции, как записываются трудовые функции, какие элементы повторяются и что можно извлекать автоматически. Использовались в основном регулярные выражения и эвристические алгоритмы. Такой подход позволял находить отдельные элементы текста, например коды компетенций вида УК-1 или ПК-5, но работал нестабильно. Малейшее изменение структуры документа могло нарушить обработку. Кроме того, система требовала большого количества ручной

настройки. Стало понятно, что обычного поиска по шаблонам недостаточно, нормативные документы содержат слишком много смысловых связей и сложных формулировок.

Использование методов обработки естественного языка и классических алгоритмов машинного обучения, стало следующим этапом разработки. Для сопоставления дисциплин с компетенциями использовались TF-IDF векторизация и алгоритмы классификации. Система начала не просто искать совпадения по шаблону, а анализировать тематическую близость текстов. Дополнительно исследовались методы извлечения сущностей из документов - компетенций, трудовых функций и других структурных элементов ФГОС. После этого система стала заметно устойчивее. Уменьшилось количество ручной обработки, а качество извлечения данных повысилось.

В систему была интегрирована генеративная модель YandexGPT, это стало слеющим этапом разработки. Именно после этого появилась возможность автоматически формировать дисциплины, тематические планы и содержательные разделы рабочих программ дисциплин. Разница с предыдущими версиями системы была довольно заметной. Если раньше алгоритмы в основном искали и сопоставляли текст, то большие языковые модели уже работали с контекстом документа целиком. Система начала формировать более связные и содержательные результаты, близкие по структуре к реальным учебно-методическим материалам.

На заключительном этапе разработки появился модуль генерации рабочих программ дисциплин. Система начала автоматически формировать DOCX-документы с титульным листом, таблицами компетенций, тематическим планированием, образовательными технологиями и оценочными средствами.

Для передачи данных между модулями использовался JSON-формат. Это позволило связать языковую модель, обработку данных и генерацию документов в единую систему.

В итоге разработка прошла путь от простых эвристических алгоритмов до полноценной интеллектуальной системы, объединяющей NLP, генеративные модели и автоматическую подготовку учебно-методических материалов. При этом каждая следующая версия системы не заменяла предыдущую полностью, а расширяла и дополняла уже реализованные механизмы.

1.6 Постановка задачи исследования

Проведённый анализ показал, что подготовка учебно-методической документации в вузах до сих пор остаётся довольно тяжёлым и во многом ручным процессом. Учебные планы, рабочие программы дисциплин, распределение компетенций - всё это требует большого количества времени и постоянной проверки нормативных документов.

Этот процесс является очень трудоемким. Особенно когда приходится одновременно работать с ФГОС, профессиональными стандартами и внутренними требованиями университета.

Дополнительная проблема связана с объёмом самих документов. ФГОС и профстандарты содержат десятки страниц текста с пересекающимися формулировками, компетенциями и требованиями. При ручной обработке легко пропустить связь между дисциплиной и компетенцией или допустить ошибку при распределении учебной нагрузки.

Существующие программные решения в основном помогают хранить документацию и формировать отчёты, но почти не работают с содержанием документов.

Не все решения адаптированы под новые стандарты, а интеллектуальная обработка нормативных документов в большинстве систем вообще отсутствует.

Из-за этого появилась задача разработки системы, которая могла бы автоматически анализировать нормативную документацию, формировать учебные планы и генерировать рабочие программы дисциплин с использованием NLP и больших языковых моделей.

Цель исследования: разработка интеллектуальной системы автоматизации формирования образовательных материалов на основе анализа нормативных документов с использованием технологий обработки естественного языка и генеративных моделей.

Для достижения этой цели в работе решаются следующие задачи:

– анализ существующих подходов к автоматизации учебно-методической документации,

– исследование методов NLP и генеративного искусственного интеллекта,

– разработка архитектуры системы,

– реализация модуля анализа ФГОС и профессиональных стандартов,

– разработка модуля генерации дисциплин и распределения их по семестрам,

– реализация генерации рабочих программ дисциплин,

– формирование DOCX-документов,

– проведение экспериментального исследования и оценка эффективности системы.

Объектом исследования являются процессы формирования учебно-методической документации в образовательных организациях высшего образования.

Предмет исследования: методы и программные средства интеллектуальной обработки нормативных документов и автоматизации формирования образовательных материалов.

Практическая значимость работы связана с разработкой системы, которая позволяет сократить время подготовки учебной документации и уменьшить объём ручной работы методистов при создании образовательных программ.

1.7 Выводы

В первой главе были рассмотрены особенности формирования образовательных программ на основе ФГОС и профессиональных стандартов. Установлено, что ручная подготовка учебно-методической документации требует значительных временных затрат и сопровождается высокой нагрузкой на методистов.

Проведённый анализ существующих систем автоматизации показал, что современные программные решения в основном ориентированы на хранение и сопровождение документации.

Исследование методов NLP и больших языковых моделей показало перспективность их применения для автоматизации анализа нормативной документации, извлечения компетенций и генерации структуры образовательных программ.

На основе проведённого анализа была сформулирована задача разработки интеллектуальной системы автоматизации формирования учебно-методических материалов с использованием технологий обработки естественного языка и генеративных моделей.

2 Проектирование интеллектуальной системы

2.1. Архитектура системы

Разрабатываемая система предназначена для автоматизации формирования учебных планов и рабочих программ дисциплин на основе нормативной документации. Архитектура построена по модульному принципу: отдельные компоненты отвечают за обработку документов, генерацию дисциплин, работу с компетенциями и формирование итоговых файлов. Такой подход упростил дальнейшую доработку проекта, потому что изменения в одном модуле не затрагивают остальные части системы.

Система реализована как веб-приложение на Streamlit. Пользователь загружает ФГОС и профессиональные стандарты через интерфейс, после чего может запускать генерацию учебного плана и рабочих программ дисциплин. Общая схема взаимодействия модулей показана на рисунке 4.

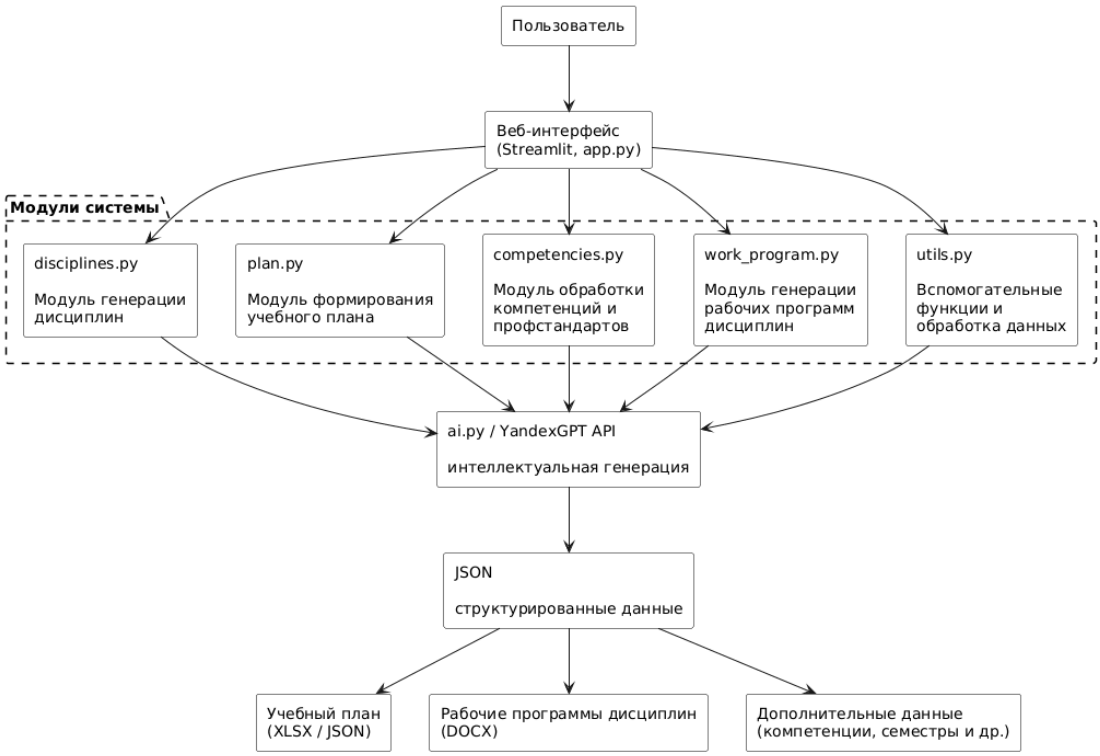


Рисунок 4 – Архитектура интеллектуальной системы

Внутри система разделена на несколько программных модулей. Отдельный модуль отвечает за работу с YandexGPT. Через него выполняется генерация дисциплин, тематических планов и содержательной части РПД.

Перед генерацией система сначала обрабатывает входные документы. Текст очищается от служебных символов, затем выполняется извлечение компетенций и других структурных элементов ФГОС. После обработки данные переводятся в JSON-формат и передаются другим компонентам системы.

Модуль генерации учебного плана формирует перечень дисциплин и распределяет их по семестрам с учётом компетенций и профиля подготовки. Дополнительно реализована обработка связей между дисциплинами, компетенциями ФГОС и трудовыми функциями профессиональных стандартов. Для сохранения учебных планов в формате Excel используется библиотека openpyxl.

Отдельно реализован модуль генерации рабочих программ дисциплин. На основе данных о дисциплине и учебном плане формируется запрос к YandexGPT, после чего результат преобразуется в структурированный JSON и используется при создании DOCX-документа.

Для генерации итоговых файлов применяется библиотека python-docx. С её помощью формируются таблицы, тематические разделы, заголовки и другие элементы РПД.

Обмен данными между модулями выполняется через JSON-структуры. Это упростило взаимодействие компонентов и позволило не привязывать отдельные части системы друг к другу слишком жёстко.

Архитура системы позволяет постепенно расширять функциональность проекта. В дальнейшем в систему можно добавить интеграцию с LMS, анализ успеваемости студентов и поддержку дополнительных образовательных стандартов.

2.2 Структура программных модулей

Разрабатываемая система построена по модульному принципу. Это упростило разработку проекта и позволило разделить обработку документов, генерацию учебных планов и создание РПД на отдельные части. При

необходимости любой компонент можно доработать отдельно, не изменяя остальные модули системы.

Основная точка взаимодействия пользователя с системой реализована в файле app.py на базе Streamlit. Через интерфейс пользователь загружает ФГОС и профессиональные стандарты, запускает генерацию дисциплин, просматривает УП и формирует РП дисциплин. Интерфейс разделён на несколько вкладок: генерация учебного плана, чат-взаимодействие с ИИ и генерация РПД.

Работа с YandexGPT вынесена в файл ai.py. Здесь формируются запросы к API, выполняется отправка HTTP-запросов и обработка ответов модели. Дополнительно реализована проверка JSON-структур и обработка ошибок генерации. Фрагмент программной структуры модулей системы приведён в приложении В, листинг В.1–В.4

Генерация дисциплин выполняется в модуле disciplines.py. Система анализирует профиль подготовки, компетенции ФГОС и текст нормативных документов, после чего формирует перечень дисциплин для учебного плана.

После этого данные передаются в модуль plan.py, где дисциплины распределяются по семестрам. При распределении используются эвристические правила и анализ содержимого дисциплин. Базовые предметы размещаются в первых семестрах, более специализированные - ближе к старшим курсам.

Обработка компетенций и профессиональных стандартов реализована в файле competencies.py. Модуль сопоставляет дисциплины с компетенциями ФГОС и трудовыми функциями профстандартов. Для анализа текста используются регулярные выражения и методы обработки текстовых данных.

Отдельно был реализован модуль work_program.py, отвечающий за генерацию рабочих программ дисциплин. Внутри используются функции generate_work_program_content() и create_work_program_docx(). Первая формирует запрос к YandexGPT и получает JSON-структуру РПД, а вторая

преобразует полученные данные в DOCX-документ с таблицами, разделами и заголовками.

Дополнительно в work_program.py реализованы вспомогательные функции обработки данных: _safe_str(), _safe_int() и _normalize_list(). Они используются при работе с неполными или некорректными данными учебного плана.

Вспомогательные функции вынесены в utils.py. Здесь находятся процедуры проверки данных и сервисные методы, используемые в разных частях проекта.

Передача данных между модулями выполняется через JSON-структуры. Такой подход позволил упростить взаимодействие компонентов системы и сделать обработку данных более последовательной - от анализа ФГОС до генерации итогового DOCX-документа.

Структура программных модулей системы представлена на рисунке 5

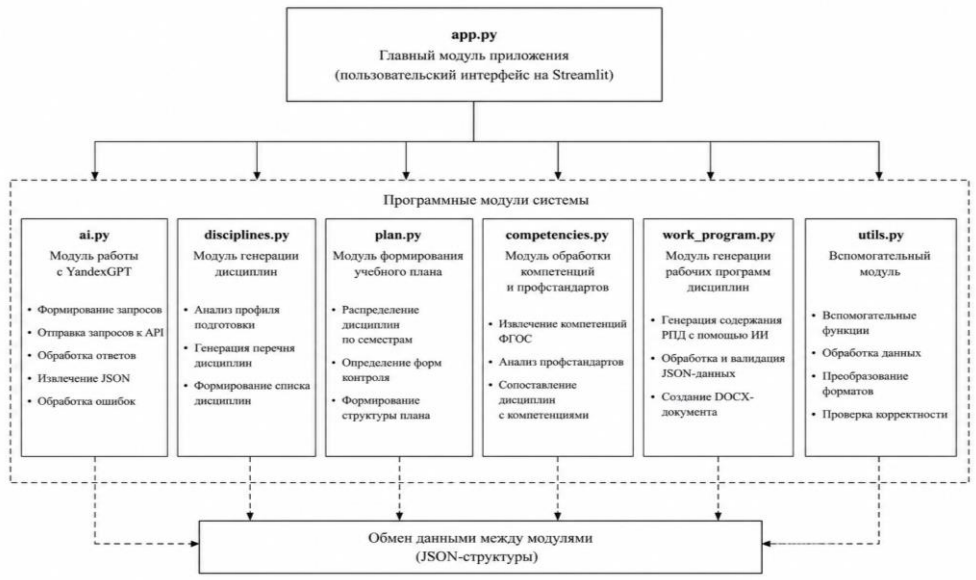


Рисунок 5 – Структура программных модулей системы

2.3 Выбор технологий и инструментов

При разработке системы подбирались технологии, которые позволяли работать с нормативными документами, взаимодействовать с языковой моделью и автоматически формировать учебно-методические материалы. Основным языком разработки был выбран Python. Такой выбор связан не

только с простым синтаксисом, но и с большим количеством библиотек для обработки текста, анализа данных и работы с API.

Python является одним из наиболее популярных языков программирования для задач анализа данных и машинного обучения [10].

Пользовательский интерфейс системы реализован с использованием Streamlit. Этот фреймворк позволил достаточно быстро собрать веб-приложение без разработки отдельной клиентской части. Через интерфейс пользователь загружает ФГОС, запускает генерацию дисциплин и формирует рабочие программы дисциплин.

Streamlit позволяет создавать интерактивные приложения для Data Science и машинного обучения непосредственно на Python [8].

Для генерации дисциплин и содержательной части РПД использовалась языковая модель YandexGPT. Взаимодействие с моделью выполняется через API Yandex Cloud с использованием HTTP-запросов.

В проекте YandexGPT применяется для генерации дисциплин, тематических планов и отдельных разделов рабочих программ дисциплин на основе текста нормативной документации.

Для хранения и передачи данных между программными модулями используется формат JSON. Это упростило взаимодействие компонентов системы и позволило передавать результаты генерации в структурированном виде.

Обработка таблиц реализована с помощью библиотеки pandas. Она используется при формировании структуры учебного плана, распределении дисциплин по семестрам и обработке данных генерации.

Для создания документов применяются библиотеки python-docx и openpyxl. С их помощью система формирует DOCX-файлы рабочих программ дисциплин и Excel-документы учебных планов.

Python-docx - библиотека для создания, редактирования и чтения документов Word в формате .docx [9].

Для работы с Excel-файлами используется openpyxl.

Модуль openpyxl - это библиотека Python для чтения и записи файлов Excel формата XLSX [11].

Дополнительно при обработке нормативной документации используются регулярные выражения RegEx. Они применяются для извлечения кодов компетенций, трудовых функций и других структурных элементов текста. Для отправки запросов к API используется библиотека requests, а для распознавания текста на изображениях - pytesseract.

В результате выбранный набор технологий позволил объединить обработку нормативных документов, генерацию образовательных материалов и автоматическое формирование DOCX-документов в рамках одной системы.

Таблица 2 – Используемые технологии и инструменты

Технология / инструмент	Назначение
Python	Основной язык разработки системы
Streamlit	Реализация веб-интерфейса
YandexGPT API	Генерация дисциплин и РПД
pandas	Обработка табличных данных
openpyxl	Формирование и сохранение Excel-документов
python-docx	Генерация рабочих программ дисциплин в формате DOCX
requests	Отправка HTTP-запросов к API
pytesseract	OCR-распознавание текста
RegEx	Извлечение компетенций и структурных элементов
JSON	Обмен структурированными данными между модулями

2.4 Интеграция YandexGPT

Одним из самых важных компонентов системы стала интеграция YandexGPT. В этой архитектуре модель используется не просто как внешний сервис для генерации текста. Через неё проходит формирование дисциплин, тематических планов и рабочих программ дисциплин. Модель получает контекст из нормативной документации и на его основе формирует

структурированный результат, который затем обрабатывается остальными модулями системы.

Большие языковые модели способны генерировать осмысленный текст с учётом контекста запроса [6]. Именно это и стало причиной использования LLM в проекте.

Связь с YandexGPT организована через API Yandex Cloud. Для отправки запросов используется библиотека requests и HTTP-запросы. Вся логика взаимодействия вынесена в модуль ai.py. Здесь формируются запросы к модели, передаются параметры генерации и обрабатываются ответы API.

Каждый запрос состоит из двух частей: системного промпта и пользовательского сообщения. В системном промпте задаётся роль модели и ограничения на структуру ответа. Во втором сообщении передаются уже конкретные данные: дисциплина, профиль подготовки, компетенции и трудовые функции [12].

Во время разработки отдельной проблемой стала структура ответов модели. Если отправлять запрос без ограничений, LLM начинает добавлять пояснения, комментарии и лишний текст. Для автоматической обработки это неудобно. Поэтому система требует возвращать результат строго в формате JSON. JSON представляет собой простой формат обмена данными, удобный для машинной обработки.

Использование JSON позволило передавать результаты генерации между программными модулями без дополнительной обработки текста. Реализация обращения к API YandexGPT представлена в приложении В, листинг В.1

Фрагмент обращения к API YandexGPT представлен ниже:

```
response = requests.post(
    YANDEX_COMPLETION_URL,
    headers=get_yandex_headers(),
    json=data,
    timeout=60
)
```

Во время разработки стало понятно, что одной модели для всех задач недостаточно. В системе реализована двухступенчатая схема генерации. На первом этапе используется облегчённая модель yandexgpt-lite, которая выполняет предварительное извлечение и структурирование данных: выделение дисциплин, компетенций и параметров учебного плана. На втором этапе применяется полная модель yandexgpt, предназначенная для генерации более сложных и содержательных фрагментов, включая тематические планы и рабочие программы дисциплин.

Промпты внутри системы тоже разделены по задачам. Отдельные шаблоны используются для генерации дисциплин, обработки компетенций и формирования РПД. В запросах задаются ограничения на количество тем, структуру таблиц и формат итогового ответа. Без этого модель начинает слишком свободно интерпретировать запрос, а при работе с учебно-методической документацией это быстро приводит к ошибкам.

Отдельно была реализована обработка ошибок API. Система проверяет HTTP-ответы, корректность JSON-структур и сетевые исключения. Если модель возвращает некорректный ответ или API недоступен, пользователь получает сообщение с описанием причины ошибки.

Интеграция YandexGPT изменила сам подход к работе системы. Ранние версии проекта в основном опирались на шаблоны, регулярные выражения и эвристики. После перехода к LLM система начала учитывать контекст нормативных документов и формировать результаты, которые по структуре уже близки к реальным вузовским материалам.

2.5 Механизм JSON-schema-driven prompting и верификации ответов LLM

Одной из основных проблем использования больших языковых моделей в прикладных системах является нестабильность структуры генерируемого ответа. При одинаковых запросах модель может возвращать различный формат данных, добавлять пояснения или нарушать структуру результата. Для

решения данной проблемы в разработанной системе реализован механизм JSON-schema-driven prompting и последующей верификации ответа LLM.

Для каждого типа генерации задаётся собственная JSON-схема. При генерации дисциплин модель должна вернуть перечень обязательных и вариативных дисциплин. При формировании рабочей программы дисциплины модель возвращает структурированные разделы РПД. При работе ИИ-консультанта ответ преобразуется в команду изменения учебного плана: добавление дисциплины, изменение поля, удаление записи или сообщение об ошибке.

После получения ответа выполняется его программная верификация. Система извлекает JSON-фрагмент, выполняет его разбор и проверяет наличие обязательных полей. Если ответ не соответствует ожидаемой структуре, система не передает его в следующие модули, а возвращает диагностическое сообщение или использует резервный механизм обработки. Такой подход снижает вероятность передачи невалидных данных в модули формирования учебного плана и генерации DOCX-документов.

При обнаружении невалидного JSON-ответа система блокирует передачу данных в последующие модули обработки. Пользователю выводится диагностическое сообщение с описанием ошибки генерации. Дополнительно реализован механизм повторной генерации запроса, позволяющий повторно обратиться к языковой модели при возникновении ошибок структуры ответа или сетевых исключений. Такой подход повышает устойчивость системы к нестабильности LLM-генерации.

Общая последовательность обработки ответа языковой модели и проверки JSON-структуры представлена на рисунке 6.

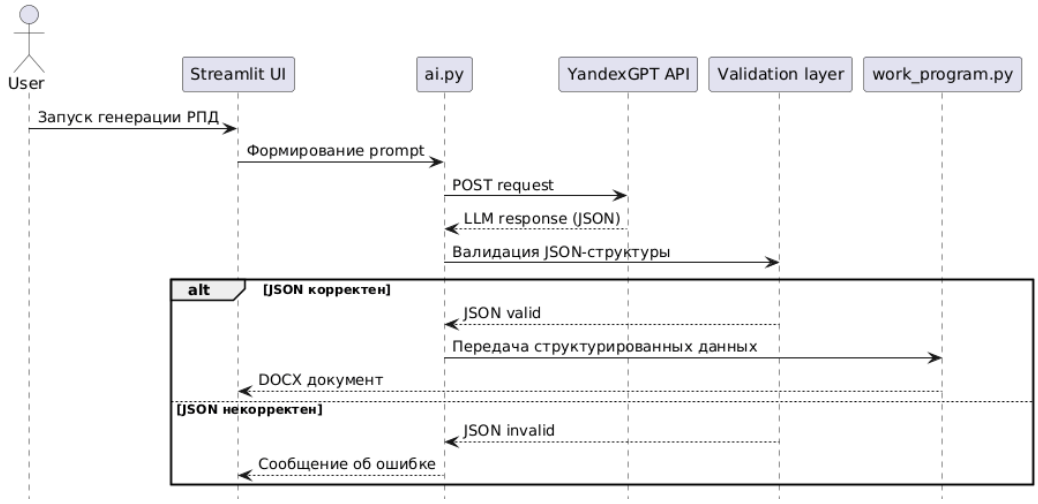


Рисунок 6 – UML-диаграмма последовательности обработки и валидации ответа языковой модели

Как показано на рисунке 6, после формирования prompt-конструкции система отправляет запрос к API YandexGPT и получает ответ в формате JSON. Далее выполняется проверка корректности полученных данных. При успешной валидации структурированные данные передаются в модуль генерации DOCX-документов. В случае нарушения структуры JSON система блокирует дальнейшую обработку и возвращает сообщение об ошибке пользователю. Фрагмент обработки JSON-ответов модели приведён в приложении В, листинг В.5

Данный механизм можно рассматривать как формализованную модель повышения надёжности генерации. Он ограничивает произвольность ответа LLM и переводит результат генерации в машиночитаемый вид. Это особенно важно при работе с учебно-методической документацией, где ошибка в структуре данных может привести к некорректному распределению дисциплин, неправильному заполнению компетенций или сбою при создании DOCX-файла.

Применение JSON-schema-driven prompting позволило связать генеративную модель с программными модулями системы и использовать LLM не как обычный чат-инструмент, а как компонент автоматизированного конвейера обработки данных.

2.6 Формат хранения и обработки данных

В системе используется несколько форматов хранения данных, потому что на разных этапах обработки информация выглядит по-разному. Часть данных приходит в виде нормативных документов, часть генерируется языковой моделью, а итоговые результаты сохраняются в форматах, с которыми потом работают методисты и преподаватели.

На вход система получает ФГОС и профессиональные стандарты в формате PDF. После загрузки документы проходят предварительную обработку: текст очищается от служебных символов, выполняется нормализация регистра, извлекаются компетенции, трудовые функции и другие структурные элементы.

Если документ загружен в виде скана или изображения, дополнительно используется OCR-распознавание с помощью pytesseract. Это позволяет извлекать текст даже из неструктурированных копий документов.

Для внутреннего обмена данными между программными модулями используется JSON. По сути, это основной формат взаимодействия компонентов системы. Практически все результаты генерации - дисциплины, тематические планы, компетенции и структура РПД - сначала переводятся в JSON и только потом передаются другим модулям.

Работать с JSON оказалось удобнее, чем с обычным текстом. Система может отдельно получить список компетенций, количество часов или темы дисциплины без повторного анализа ответа модели.

Пример структуры данных при генерации рабочей программы дисциплины приведён ниже:

```
json
{
  "discipline_name": "Информационные технологии",
  "total_hours": 108,
  "competencies": ["УК-1", "ОПК-2"],
  "lecture_topics": [
    "Основы информационных технологий",
    "Информационные системы"
  ]
}
```

Во время разработки пришлось отдельно решать проблему нестабильности ответов языковой модели. Даже при одинаковом запросе структура ответа может меняться: модель добавляет пояснения, нарушает форматирование или возвращает лишний текст. Из-за этого в системе была реализована дополнительная проверка JSON-структур и функции безопасной обработки данных.

Для хранения учебных планов используется табличный формат. После генерации данные преобразуются в pandas DataFrame, где каждая строка соответствует отдельной дисциплине. Это упростило сортировку, фильтрацию и распределение дисциплин по семестрам.

Итоговые учебные планы сохраняются в формате Excel. Такой формат удобен для дальнейшего редактирования и согласования документов внутри вуза. Для формирования Excel-файлов используется библиотека openpyxl.

Рабочие программы дисциплин формируются в формате DOCX с использованием библиотеки python-docx. Система автоматически создаёт таблицы, тематические разделы, заголовки и другие элементы оформления.

Такой формат был выбран не случайно. В большинстве вузов учебно-методическая документация продолжает оформляться именно в DOCX, поэтому преподаватель может при необходимости вручную отредактировать уже сгенерированный документ.

Использование нескольких форматов хранения позволило разделить внутреннее представление данных и итоговые пользовательские документы. JSON применяется для обмена данными между модулями, Excel используется для хранения структуры учебного плана, а DOCX - для формирования готовых рабочих программ дисциплин.

Таблица 3 – Используемые форматы данных

Формат	Назначение
PDF	Загрузка ФГОС и профстандартов
DOCX	Формирование рабочих программ дисциплин

Продолжение таблицы 3

JSON	Внутренний обмен данными между модулями
XLSX	Сохранение учебных планов
DataFrame	Обработка табличных данных

2.7 Выводы

Во второй главе была разработана архитектура интеллектуальной системы автоматизации формирования учебно-методической документации. Система реализована по модульному принципу, что позволило разделить обработку нормативных документов, генерацию дисциплин, формирование УП и создание рабочих программ дисциплин на независимые компоненты.

В ходе проектирования были выбраны основные технологии и инструменты разработки: Python, Streamlit, YandexGPT API, pandas, openpyxl и python-docx. Их совместное использование позволило реализовать обработку нормативных документов, генерацию образовательных материалов и автоматическое формирование итоговых DOCX- и XLSX-документов.

Особое внимание было уделено интеграции YandexGPT и организации взаимодействия с языковой моделью. В системе реализована двухступенчатая схема генерации, а также механизм JSON-schema-driven prompting и верификации ответов LLM, обеспечивающий передачу структурированных данных между программными модулями и повышение устойчивости генерации.

Дополнительно были рассмотрены используемые форматы хранения и обработки данных. В качестве внутреннего формата обмена данными применяется JSON, для хранения учебных планов используется XLSX, а для формирования рабочих программ дисциплин – DOCX.

Таким образом, результаты проектирования подтвердили возможность построения интеллектуальной системы, способной автоматизировать обработку нормативной документации и подготовку образовательных материалов с использованием технологий NLP и генеративных моделей.

3 Реализация и тестирование системы

3.1 Реализация пользовательского интерфейса

Пользовательский интерфейс системы реализован с использованием фреймворка Streamlit. Выбор данной технологии связан с необходимостью быстро организовать веб-интерфейс для работы с учебными планами, нормативными документами и генерацией рабочих программ дисциплин без разработки отдельной клиентской части. Streamlit -это открытая библиотека на языке программирования Python, разработанная для быстрого создания интерактивных веб-приложений, ориентированных на анализ данных, машинное обучение и научные вычисления [14].

Интерфейс приложения построен по принципу последовательной работы пользователя с системой. Основные функции распределены по отдельным вкладкам: генерация учебного плана, взаимодействие с ИИ и формирование рабочих программ дисциплин. Общий вид интерфейса представлен на рисунке 7.

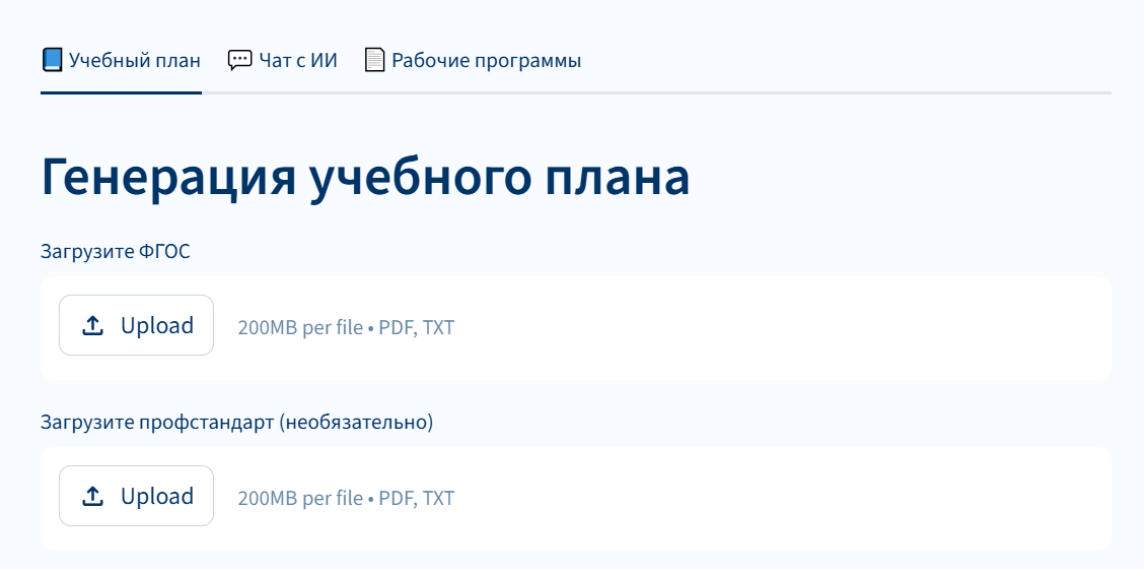
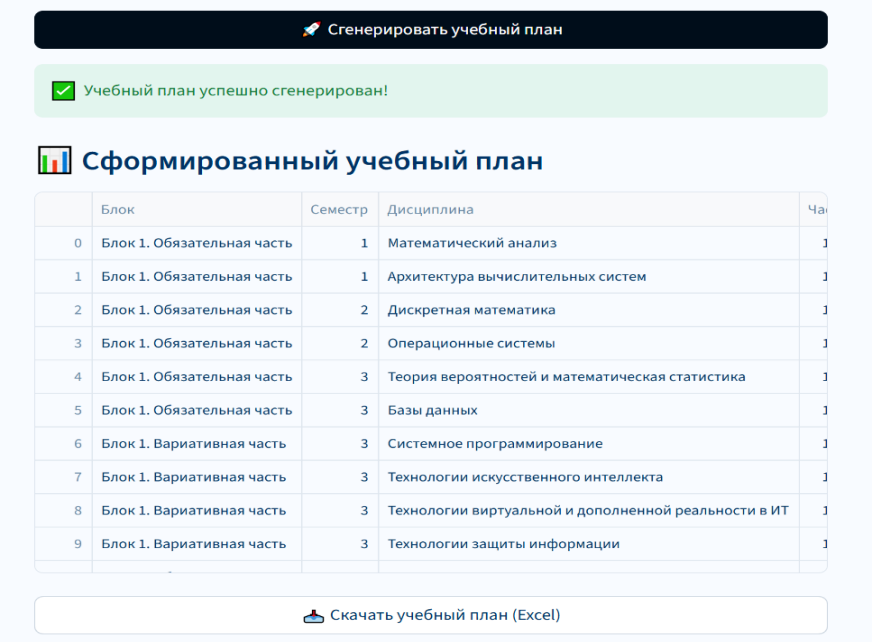


Рисунок 7 – Главный интерфейс Streamlit-приложения

Через интерфейс пользователь может загрузить федеральный государственный образовательный стандарт, профессиональный стандарт и дополнительные документы, необходимые для формирования учебного плана. После загрузки файлов система выполняет обработку данных и отображает результаты в табличном виде.

Одной из ключевых частей интерфейса является таблица учебного плана, отображающая результаты генерации дисциплин, распределение семестров, формы контроля и компетенции. Изменение структуры учебного плана выполняется не напрямую через таблицу, а посредством взаимодействия с ИИ-консультантом. Пользователь может отправлять текстовые команды на естественном языке, например изменять количество часов, добавлять дисциплины или корректировать параметры учебного плана. Полученные команды преобразуются системой в JSON-структуры и применяются к данным учебного плана автоматически. Пример отображения учебного плана представлен на рисунке 8.



	Блок	Семестр	Дисциплина	Ча
0	Блок 1. Обязательная часть	1	Математический анализ	1
1	Блок 1. Обязательная часть	1	Архитектура вычислительных систем	1
2	Блок 1. Обязательная часть	2	Дискретная математика	1
3	Блок 1. Обязательная часть	2	Операционные системы	1
4	Блок 1. Обязательная часть	3	Теория вероятностей и математическая статистика	1
5	Блок 1. Обязательная часть	3	Базы данных	1
6	Блок 1. Вариативная часть	3	Системное программирование	1
7	Блок 1. Вариативная часть	3	Технологии искусственного интеллекта	1
8	Блок 1. Вариативная часть	3	Технологии виртуальной и дополненной реальности в ИТ	1
9	Блок 1. Вариативная часть	3	Технологии защиты информации	1

Рисунок 8 – Сгенерированный учебный план

Отдельная вкладка предназначена для генерации рабочих программ дисциплин. Пользователь выбирает дисциплину из учебного плана, после чего система автоматически формирует РПД и сохраняет документ в формате DOCX. Интерфейс генерации рабочих программ представлен на рисунке 9.

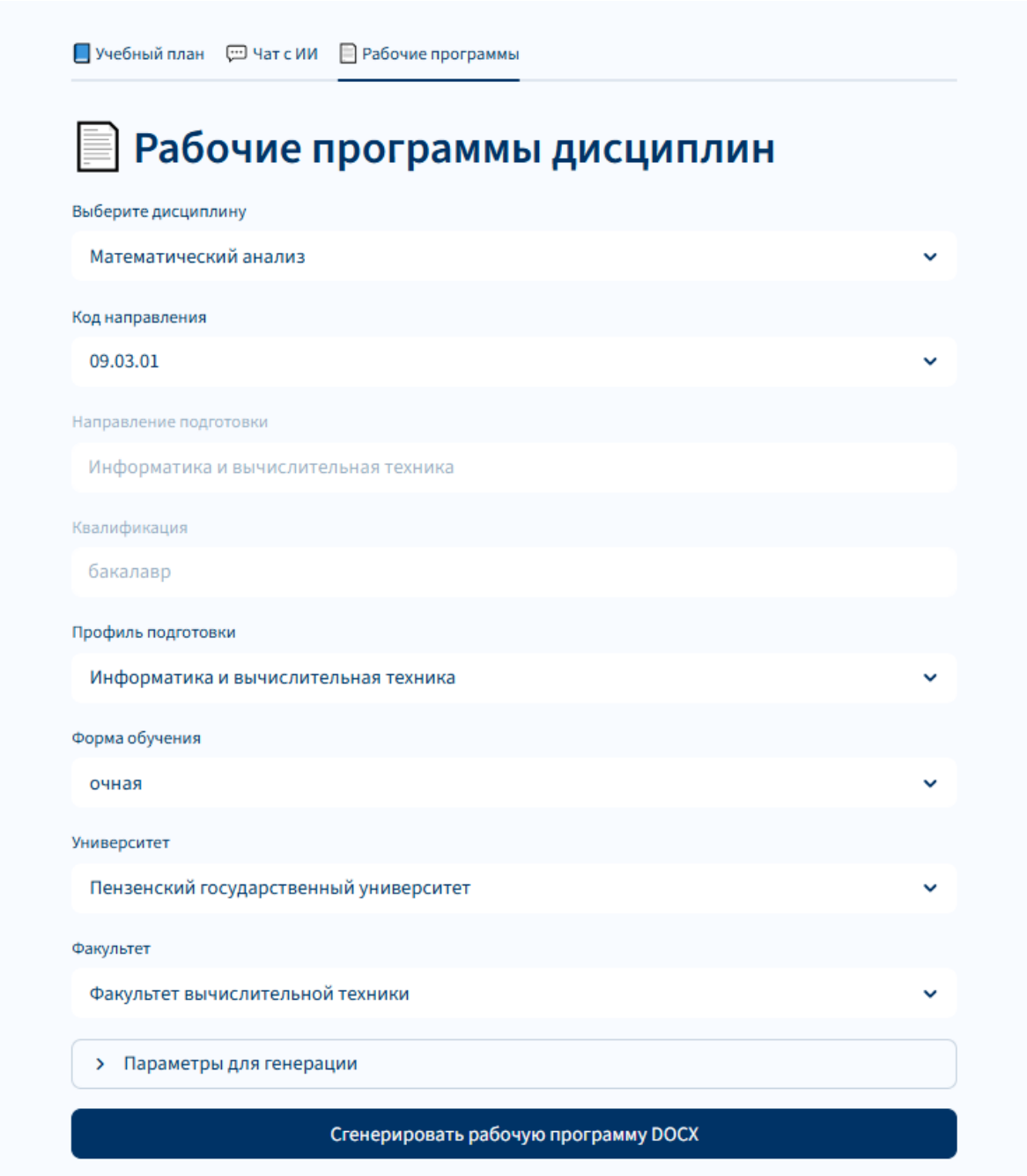


Рисунок 9 – Интерфейс генерации рабочей программы дисциплины

Во время разработки интерфейса особое внимание уделялось возможности ручного редактирования результатов генерации. Несмотря на использование ИИ, система не предполагает полностью автоматическую подготовку документов без участия пользователя. Поэтому большинство элементов учебного плана могут изменяться вручную непосредственно через интерфейс приложения.

Для хранения промежуточных данных используются объекты session_state Streamlit. Это позволяет сохранять состояние учебного плана,

историю взаимодействия с ИИ и результаты генерации между действиями пользователя без повторной загрузки документов.

Фрагмент программной реализации создания вкладок интерфейса представлен ниже.

```
tab1, tab2, tab3 = st.tabs([
    "Учебный план",
    "ИИ-консультант",
    "Рабочие программы"
])
```

Интерфейс приложения разрабатывался с учётом того, что пользователями системы будут преподаватели и методисты, не имеющие опыта программирования. По этой причине большая часть взаимодействия реализована через таблицы, формы загрузки файлов и кнопки запуска генерации, а не через настройку параметров вручную.

3.2 Реализация генерации дисциплин

Генерация дисциплин является одним из основных этапов работы системы, поскольку именно на этом этапе формируется предварительная структура будущего учебного плана. В качестве исходных данных используются федеральный государственный образовательный стандарт, профессиональный стандарт и профиль подготовки.

Работа начинается с загрузки нормативных документов через интерфейс приложения. После загрузки система выполняет предварительную обработку текста и извлекает основные структурные элементы ФГОС: компетенции, коды направлений подготовки и описания образовательных результатов. Результат обработки ФГОС представлен на рисунке 10.

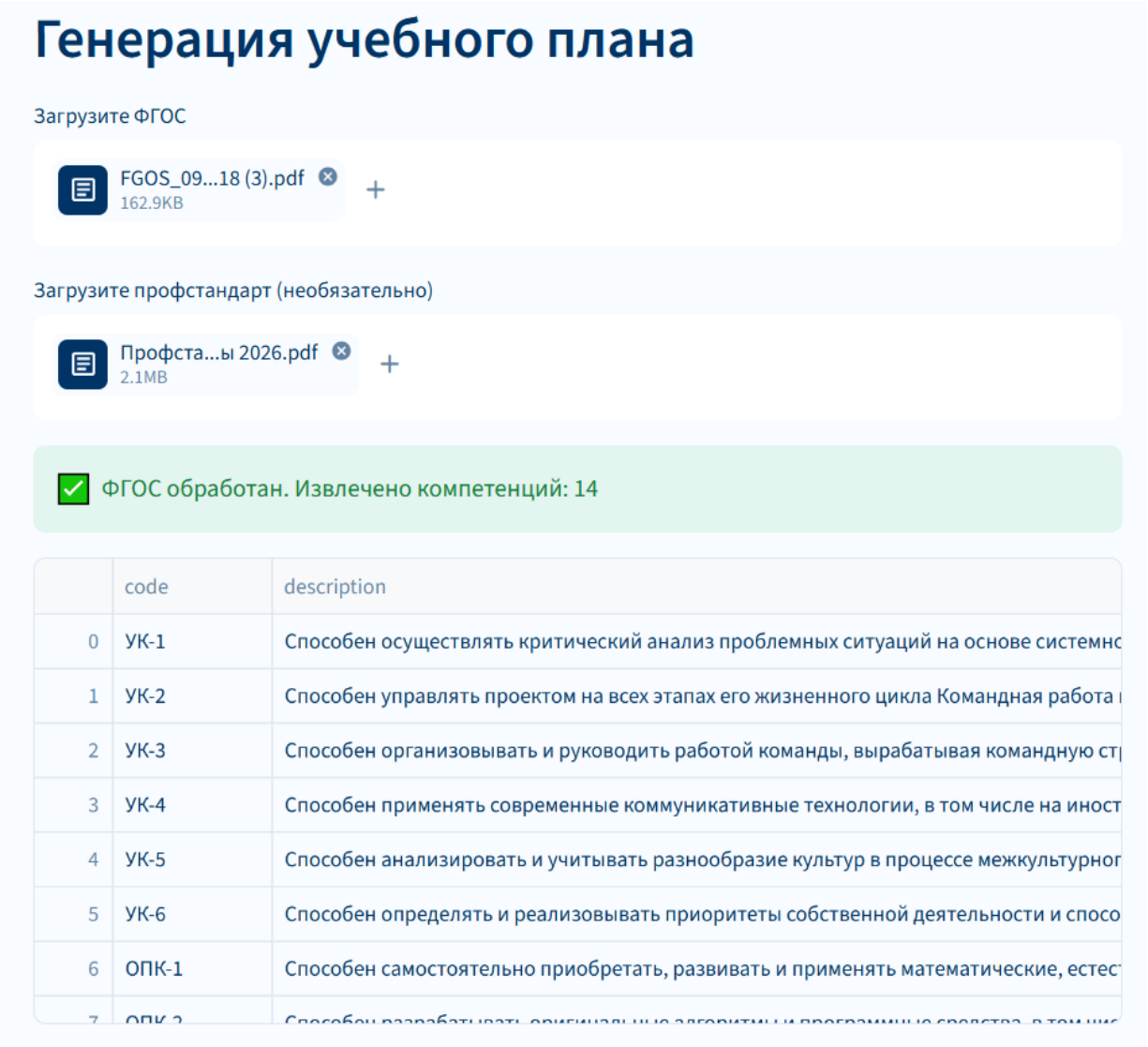


Рисунок 10 – Обработка ФГОС и извлечение компетенций

Для извлечения компетенций используются регулярные выражения и дополнительная обработка текста. После анализа документа система формирует структурированный список компетенций, который затем используется при генерации дисциплин и сопоставлении образовательных результатов.

Дополнительно пользователь может загрузить профессиональный стандарт. В этом случае система выполняет анализ трудовых функций и направлений профессиональной деятельности. Полученные данные используются как дополнительный контекст при формировании учебного плана. Результат обработки профессионального стандарта представлен на рисунке 11.

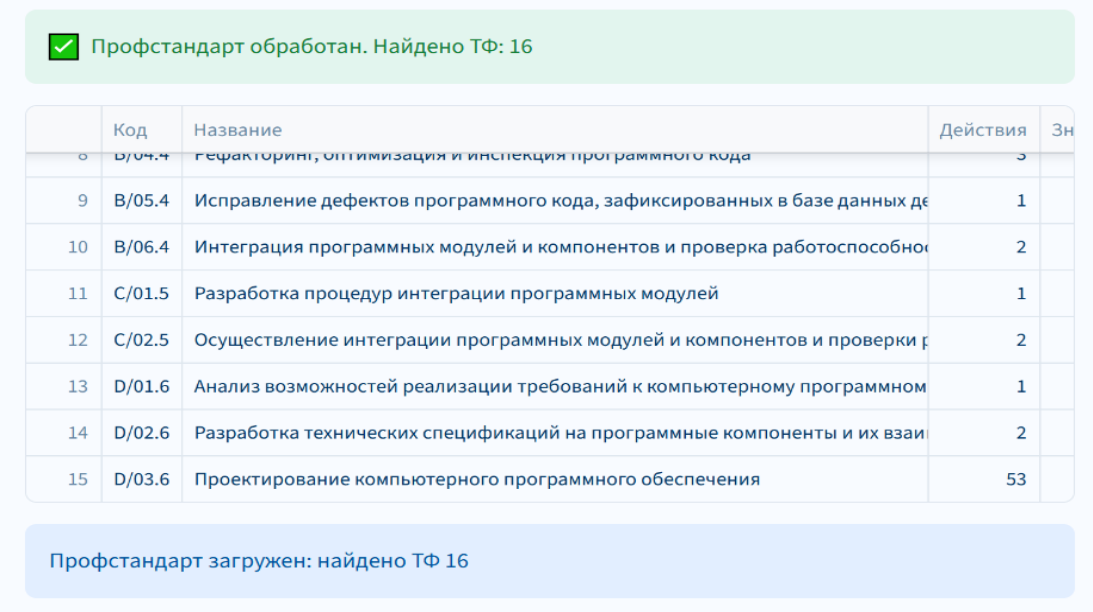


Рисунок 11 – Обработка профессионального стандарта

Общая схема обработки нормативных документов и генерации дисциплин представлена на рисунке 12.

Рисунок 12 – Схема обработки нормативных документов и генерации дисциплин

После извлечения компетенций ФГОС и трудовых функций профессионального стандарта система выполняет их контекстное сопоставление при формировании учебного плана. Полученные данные используются как дополнительный контекст при генерации дисциплин языковой моделью.

В prompt-конструкцию передаются профиль подготовки, перечень компетенций, трудовые функции и параметры образовательной программы. На основе этих данных модель формирует перечень дисциплин, содержание которых соответствует профессиональной области подготовки.

В отличие от жёсткого сопоставления по ключевым словам система использует контекстно-ориентированный подход, позволяющий учитывать смысловую близость между компетенциями, трудовыми функциями и

дисциплинами даже при различии формулировок. Например, для направления «Информатика и вычислительная техника» система генерирует дисциплины, связанные с программированием, архитектурой вычислительных систем, базами данных и алгоритмами, а для педагогических направлений – дисциплины, связанные с педагогикой, методикой обучения и возрастной психологией.

После обработки нормативных документов система формирует запрос к YandexGPT. В запрос передаются профиль подготовки, перечень компетенций, трудовые функции и параметры образовательной программы. На основе этих данных языковая модель генерирует перечень дисциплин, соответствующих направлению подготовки. Для уменьшения нагрузки на систему генерация выполняется по двухступенчатой схеме. Сначала облегчённая модель формирует предварительный структурированный список дисциплин в формате JSON. Затем полученные данные передаются в последующие модули, где используются при формировании учебного плана и рабочих программ дисциплин.

Во время разработки стало понятно, что генерация обычного текстового списка дисциплин неудобна для дальнейшей обработки. По этой причине система требует от модели возвращать результат в формате JSON. Такой подход позволяет автоматически извлекать названия дисциплин, количество часов, формы контроля и другие параметры учебного плана. Ключевой фрагмент алгоритма генерации дисциплин представлен в приложении В, листинг В.3

Пример JSON-структуры, используемой при генерации дисциплин, представлен ниже.

```
{
  "discipline": "Информационные технологии",
  "semester": 2,
  "hours": 108,
  "control": "экзамен"
}
```

После получения ответа выполняется проверка структуры данных. Система контролирует корректность JSON, наличие обязательных полей и типы значений. Если модель возвращает некорректный ответ, пользователю выводится сообщение об ошибке.

Полученные данные преобразуются в pandas DataFrame и используются для дальнейшего формирования учебного плана. Табличный формат позволяет выполнять дополнительную обработку дисциплин, распределение по семестрам и сохранение результатов в Excel.

Фрагмент преобразования данных в DataFrame представлен ниже.

```
df = pd.DataFrame(disciplines)
df["Семестр"] = df["Семестр"].fillna(1)
df["Часы"] = df["Часы"].astype(int)
```

После генерации перечня дисциплин система выполняет их распределение по семестрам. Для этого используется эвристический алгоритм, учитывающий тип дисциплины, уровень её фундаментальности и предполагаемую последовательность изучения.

Базовые и фундаментальные дисциплины распределяются преимущественно в ранние семестры образовательной программы. К ним относятся математические дисциплины, алгоритмы, основы программирования, педагогика, психология и другие дисциплины, формирующие базовые компетенции.

Профессионально-ориентированные дисциплины размещаются в средних семестрах, когда у обучающихся уже сформирована необходимая теоретическая база. Практико-ориентированные дисциплины, проектная деятельность, производственные практики и государственная итоговая аттестация переносятся в старшие семестры.

Для определения фундаментальности дисциплины используется эвристическая проверка ключевых терминов, связанных с профилем подготовки. Например, для ИТ-направлений учитываются термины, связанные с алгоритмами, архитектурой вычислительных систем, программированием и базами данных.

Общая схема распределения дисциплин по семестрам представлена на рисунке 13.

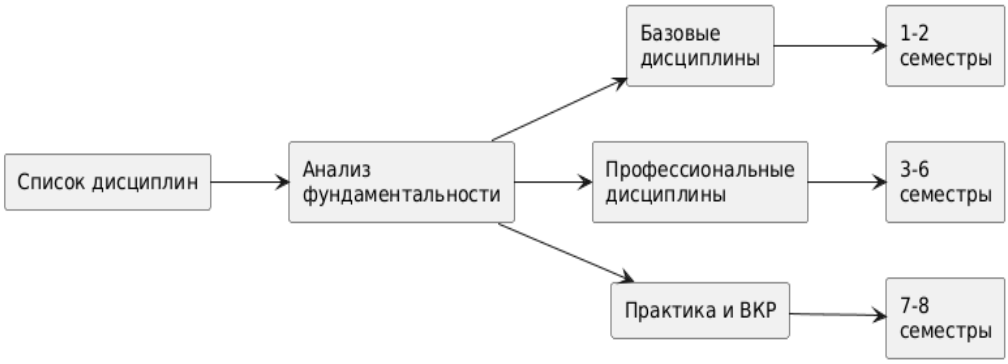


Рисунок 13 – Эвристическая схема распределения дисциплин по семестрам

После генерации пользователь может изменять структуру учебного плана через взаимодействие с ИИ-консультантом. Система поддерживает текстовые команды на естественном языке, позволяющие добавлять дисциплины, изменять количество часов или корректировать параметры учебного плана без прямого редактирования таблицы.

Реализованный механизм генерации дисциплин позволил автоматизировать один из наиболее трудоёмких этапов проектирования образовательных программ и сократить время подготовки предварительной структуры учебного плана.

3.3 Реализация модуля генерации РПД

Одним из наиболее крупных компонентов системы является модуль генерации рабочих программ дисциплин. Данный модуль предназначен для автоматического формирования РПД на основе параметров учебного плана, выбранной дисциплины и данных, полученных из нормативных документов.

Генерация РПД выполняется в отдельной вкладке пользовательского интерфейса. Пользователь выбирает дисциплину, код направления подготовки, профиль, форму обучения, университет и факультет. Для большинства полей реализованы выпадающие списки с возможностью ввода собственного значения.

Во время разработки модуля особое внимание уделялось удобству заполнения параметров. Изначально часть полей вводилась вручную, однако при тестировании стало понятно, что большое количество однотипных данных усложняет работу пользователя. По этой причине были добавлены выпадающие списки с возможностью поиска значений и выбора готовых вариантов. Пример выбора профиля подготовки представлен на рисунке 14.

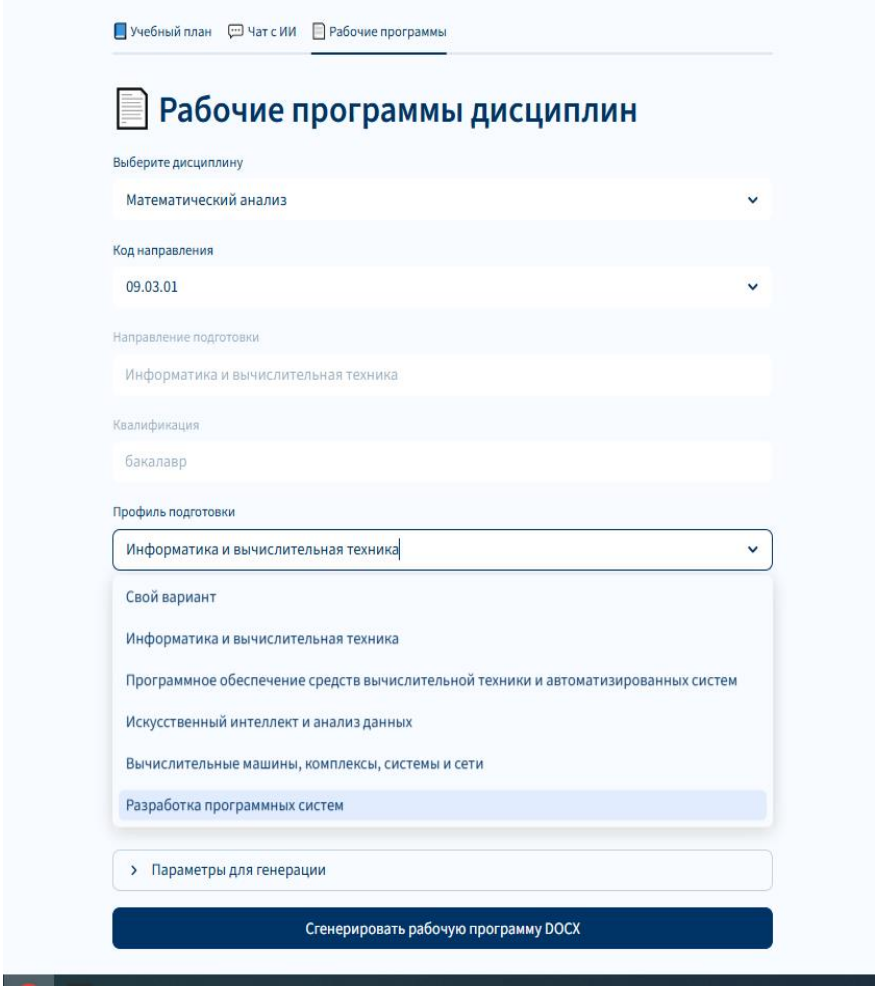


Рисунок 14 – Выпадающий список профиля подготовки

Аналогичный механизм используется для выбора формы обучения, университета и факультета. При этом пользователь может как выбрать готовое значение, так и указать собственный вариант. Это позволило сделать интерфейс более универсальным и пригодным для различных образовательных организаций.

После заполнения параметров система формирует JSON-структуру, содержащую данные, необходимые для генерации рабочей программы

дисциплины. Пример структуры параметров генерации представлен на рисунке 15.

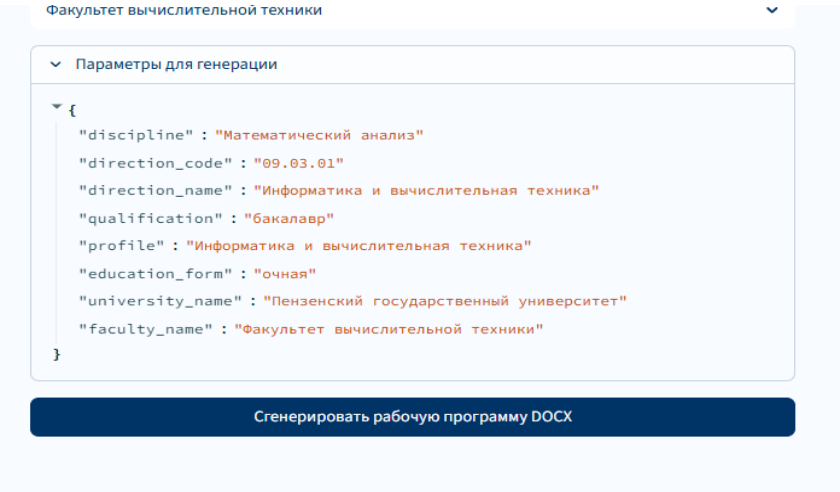


Рисунок 15 – Параметры генерации в формате JSON

На основе полученных данных формируется запрос к YandexGPT. В запрос передаются название дисциплины, направление подготовки, профиль, компетенции, количество часов и форма контроля. Дополнительно в промпте задаются ограничения на структуру ответа и содержание разделов РПД [17].

Генерация РПД относится ко второму этапу двухступенчатой схемы, так как требует более длинного контекста и формирования связного учебно-методического текста. Поэтому для данного модуля используется полная модель YandexGPT.

Во время разработки возникла проблема нестабильности генерации. При отсутствии ограничений языковая модель могла формировать неполные таблицы, изменять структуру документа или добавлять лишний текст. Для решения этой проблемы система требует от модели возвращать строго структурированный JSON без дополнительных комментариев. Фрагмент функции генерации РПД приведён в приложении В, листинг В.6

Фрагмент функции генерации рабочей программы представлен ниже.

```
def generate_work_program_content(
    discipline_row,
    profile,
    direction_code,
    direction_name,
    qualification,
    education_form,
```

```
        university_name,
        faculty_name,
    )
```

После получения ответа выполняется проверка структуры данных и извлечение необходимых разделов документа. Далее информация передаётся в модуль формирования DOCX-файла. Последовательность состояний системы при обработке нормативных документов и генерации данных представлена на рисунке 16.

Рисунок 16 – UML-диаграмма состояний процесса генерации данных

Как показано на рисунке 10, после загрузки нормативных документов система выполняет обработку ФГОС и переходит к этапу генерации данных. В случае успешной генерации формируется итоговый результат, а при возникновении ошибок система переходит в состояние обработки ошибки с возможностью повторной генерации.

Генерация документа реализована с использованием библиотеки python-docx. Система автоматически создает титульный лист, таблицы результатов освоения дисциплины, тематический план, перечень литературы, самостоятельную работу и другие разделы РПД. Сформированный документ сохраняется в формате DOCX и становится доступным для скачивания через интерфейс приложения. Результат генерации представлен на рисунке 17.

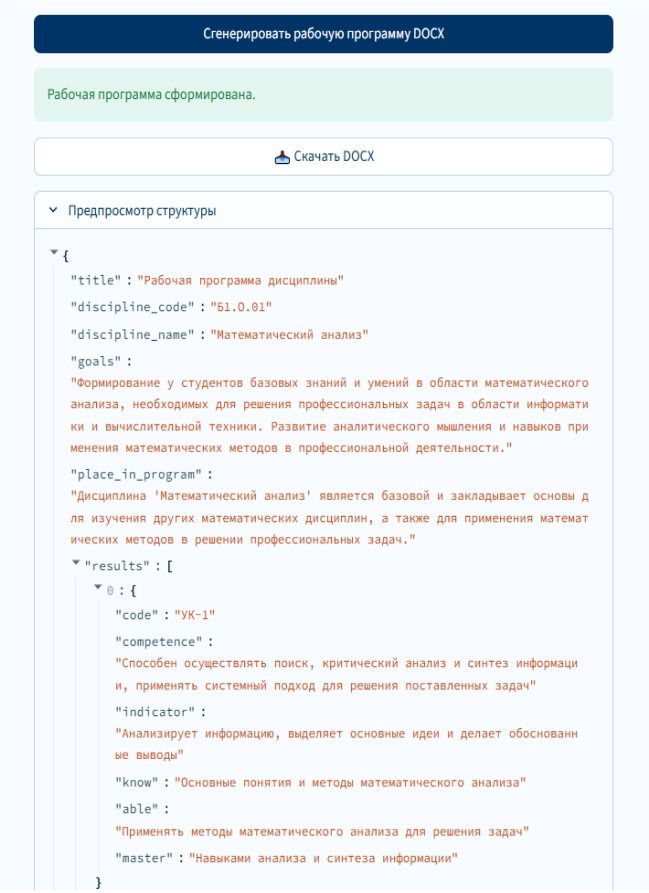


Рисунок 17 – Результат генерации РПД и предпросмотр структуры

Пример сформированной рабочей программы дисциплины представлен в приложении А.

Во время тестирования было установлено, что автоматическая генерация позволяет существенно сократить время подготовки РПД по сравнению с ручным заполнением документа. При этом пользователь сохраняет возможность редактирования итогового DOCX-файла после генерации.

3.4 Генерация DOCX-документов

Одним из завершающих этапов работы системы является формирование документов в формате DOCX. В разработанном приложении DOCX используется для автоматического создания рабочих программ дисциплин, пригодных для дальнейшего редактирования и использования в учебно-методической работе.

Во время разработки было важно не просто сформировать текстовый файл, а приблизить структуру документа к реальным вузовским шаблонам

РПД. По этой причине система генерирует титульный лист, таблицы результатов освоения дисциплины, тематический план, перечень литературы, разделы самостоятельной работы и другие элементы оформления.

Генерация документов реализована с использованием библиотеки python-docx. Формирование файла выполняется поэтапно: сначала создаётся структура документа, затем добавляются заголовки и таблицы, после чего в документ вставляются данные, полученные от языковой модели. Фрагмент формирования DOCX-документа представлен в приложении В, листинг В.6

Фрагмент создания DOCX-документа представлен ниже.

```
doc = Document()
table = doc.add_table(rows=1, cols=6)
table.style = "Table Grid"
```

Для оформления документа используются встроенные средства python-docx: настройка шрифтов, выравнивание текста, создание таблиц и управление отступами. В качестве основного шрифта используется Times New Roman, соответствующий требованиям оформления большинства учебно-методических документов.

Особое внимание уделялось автоматическому формированию таблиц. Система создаёт таблицу результатов освоения дисциплины, таблицу структуры дисциплины, перечень лабораторных или практических работ и раздел самостоятельной работы студентов. Пример сформированной таблицы результатов освоения дисциплины представлен на рисунке 18.

При генерации документа система дополнительно выполняет обработку компетенций. В итоговую таблицу включаются только компетенции ФГОС с кодами УК, ОПК и ПК. Трудовые функции профессионального стандарта в таблицы результатов не добавляются, поскольку они используются только как дополнительный контекст при генерации содержания дисциплины.

3. Результаты освоения дисциплины «Математический анализ»

Код	Компетенция	Индикатор	Знать	Уметь	Владеть
УК-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	Анализирует информацию, выделяет основные идеи и делает обоснованные выводы	Основные понятия и методы математического анализа	Применять методы математического анализа для решения задач	Навыками анализа и синтеза информации
ОПК-1	Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования в профессиональной деятельности	Применяет методы математического анализа для решения профессиональных задач	Методы математического анализа	Решать задачи с применением методов математического анализа	Навыками применения методов математического анализа в профессиональной деятельности
ОПК-2	Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности	Использует математические методы для решения задач с помощью информационных технологий	Основы работы с информационными технологиями	Применять математические методы с использованием информационных технологий	Навыками работы с информационными технологиями для решения математических задач
ПК-3	Способен осуществлять выбор и применение методов и инструментов для решения профессиональных задач	Выбирает методы математического анализа для решения профессиональных задач	Методы математического анализа	Выбирать методы математического анализа для решения конкретных задач	Навыками выбора методов математического анализа для решения профессиональных задач

Рисунок 18 – Таблица компетенций в DOCX-документе

Для формирования титульного листа используются данные, введённые пользователем через интерфейс приложения: университет, факультет, направление подготовки, профиль, форма обучения и название дисциплины. Пример титульного листа сформированной рабочей программы представлен на рисунке 19.

После завершения генерации документ сохраняется во временный буфер и передаётся пользователю для скачивания через интерфейс приложения. Такой подход позволил отказаться от хранения файлов на сервере и упростить работу приложения. Пример сформированной рабочей программы дисциплины представлен в приложении А.

УТВЕРЖДАЮ

Декан факультета

_____ /_____/

"____" _____ 2026 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Пензенский государственный университет

Факультет вычислительной техники

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.О.01 Математический анализ

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Направленность подготовки: «Информатика и вычислительная техника»

Квалификация выпускника: бакалавр

Форма обучения: очная

Пенза, 2026

Рисунок 19 – Титульный лист РПД

Фрагмент сохранения DOCX-документа представлен ниже.

```
buffer = io.BytesIO()
doc.save(buffer)
buffer.seek(0)
```

Реализованный механизм генерации DOCX-документов позволил автоматизировать подготовку РПД и сократить объем ручного оформления документов. Итоговый файл остаётся полностью редактируемым и может использоваться в дальнейшей учебно-методической работе без дополнительного преобразования форматов.

3.5 Обработка компетенций и профстандартов

Одним из ключевых этапов работы системы является обработка ФГОС и профессиональных стандартов. На основе этих документов формируется контекст для генерации дисциплин, учебного плана и рабочих программ дисциплин. После загрузки ФГОС система выполняет анализ текстового содержимого документа и извлекает компетенции, относящиеся к образовательной программе. В процессе обработки выделяются коды компетенций, их описания и структурные элементы, необходимые для дальнейшего формирования УП.

Для извлечения компетенций используются регулярные выражения и дополнительная обработка текста. Система анализирует строки документа и определяет конструкции, соответствующие шаблонам УК, ОПК и ПК. После извлечения данные преобразуются в структурированный формат и сохраняются для дальнейшего использования.

Фрагмент обработки компетенций представлен ниже.

```
if upper.startswith("УК-") or \
   upper.startswith("ОПК-") or \
   upper.startswith("ПК-"):
    result.append(item)
```

Во время разработки было установлено, что структура нормативных документов может существенно различаться. В разных версиях ФГОС компетенции располагаются в разных разделах и могут иметь отличия в форматировании. По этой причине система дополнительно выполняет нормализацию текста и удаление лишних символов перед анализом документа.« Фрагмент извлечения компетенций из ФГОС приведён в приложении В, листинг В.2

Помимо ФГОС система поддерживает обработку профессиональных стандартов. Из профстандартов извлекаются трудовые функции, направления профессиональной деятельности и дополнительные характеристики, связанные с профессиональной подготовкой выпускника.

Результат обработки профессионального стандарта представлен на рисунке 11.

Извлеченные трудовые функции не включаются напрямую в таблицы результатов освоения дисциплины. Они используются как дополнительный контекст при генерации дисциплин и рабочих программ, позволяя языковой модели учитывать особенности профессиональной деятельности выпускника.

После извлечения компетенций и трудовых функций система формирует структурированные JSON-данные, которые затем используются другими модулями проекта. Такой подход позволил отказаться от повторного анализа документов на последующих этапах обработки.

Пример JSON-структуры компетенций представлен ниже.

```
{
  "code": "УК-1",
  "description": "Способен осуществлять поиск, критический анализ и синтез информации"
}
```

Дополнительно в системе реализована проверка дублирующихся компетенций и очистка некорректных значений. Это оказалось особенно важно при обработке документов, содержащих повторяющиеся разделы или ошибки форматирования.

Автоматическая обработка нормативных документов позволяет существенно сократить время подготовки учебно-методических материалов. Вместо ручного поиска компетенций и трудовых функций система автоматически извлекает необходимые данные и подготавливает их для дальнейшей генерации учебного плана и РПД.

3.6 Выводы

В третьей главе была рассмотрена практическая реализация интеллектуальной системы автоматизации формирования учебно-методической документации. Был реализован пользовательский интерфейс на базе Streamlit, обеспечивающий загрузку нормативных документов, генерацию учебных планов, взаимодействие с ИИ-консультантом и формирование рабочих программ дисциплин.

В ходе реализации была разработана система генерации дисциплин на основе анализа ФГОС, профессиональных стандартов, компетенций и трудовых функций. Для повышения устойчивости генерации использовались JSON-структуры, механизмы верификации данных и двухступенчатая схема взаимодействия с YandexGPT.

Дополнительно был реализован модуль автоматической генерации рабочих программ дисциплин и формирования DOCX-документов с использованием библиотеки python-docx. Система автоматически создаёт структуру РПД, таблицы компетенций, тематическое планирование и другие элементы документа.

Также была реализована обработка компетенций и профессиональных стандартов, включая извлечение кодов компетенций, трудовых функций и их преобразование в структурированный формат для последующей генерации образовательных материалов.

Результаты реализации показали, что использование NLP-технологий и генеративных моделей позволяет автоматизировать значительную часть процессов подготовки учебно-методической документации и сократить объём рутинной работы при проектировании образовательных программ.

4 Экспериментальное исследование системы

4.1 Цель и методика тестирования

После завершения разработки системы было проведено экспериментальное исследование, направленное на проверку корректности работы основных программных модулей и оценку качества формируемых результатов. Тестирование выполнялось на различных вариантах федеральных государственных образовательных стандартов, профессиональных стандартов и параметров образовательных программ.

Основной целью тестирования являлась проверка возможности автоматического формирования учебных планов и рабочих программ дисциплин на основе нормативной документации. Дополнительно оценивалась устойчивость работы системы при изменении входных данных, корректность обработки JSON-структур и качество сформированных DOCX-документов.

В процессе исследования проверялись следующие функции системы:

– загрузка и обработка ФГОС,

– извлечение компетенций и трудовых функций,

– генерация дисциплин учебного плана,

– взаимодействие с YandexGPT,

– формирование рабочих программ дисциплин,

– экспорт документов в форматы Excel и DOCX.

Тестирование проводилось поэтапно. На первом этапе проверялась корректность обработки нормативных документов. Для этого в систему загружались различные версии ФГОС и профессиональных стандартов, содержащие отличия в структуре и форматировании текста. Оценивалась способность системы корректно извлекать компетенции, коды направлений подготовки и трудовые функции.

На втором этапе тестировалась генерация учебного плана. Проверялось соответствие дисциплин профилю подготовки, корректность распределения

часов и форм контроля, а также устойчивость генерации при изменении параметров запроса.

Отдельное внимание уделялось тестированию модуля генерации рабочих программ дисциплин. Проверялась полнота структуры РПД, корректность формирования таблиц, наличие обязательных разделов и соответствие итогового документа типовой структуре вузовских рабочих программ.

Во время тестирования использовались как автоматически сформированные данные, так и реальные примеры нормативных документов. Один из примеров сформированной рабочей программы дисциплины представлен в приложении А.

Для оценки результатов использовались следующие критерии:

– корректность обработки входных документов,

– полнота извлечения компетенций,

– отсутствие ошибок JSON-структуры,

– корректность формирования DOCX-документов,

– соответствие структуры РПД требованиям учебно-методической документации,

– устойчивость работы интерфейса при изменении параметров генерации.

Тестирование выполнялось в ручном режиме через пользовательский интерфейс системы. Дополнительно проводилась проверка промежуточных JSON-структур и результатов обработки данных на каждом этапе генерации.

Результаты проведённого экспериментального исследования представлены в последующих разделах.

4.2 Тестирование генерации учебных планов

Для проверки работоспособности разработанной системы было проведено тестирование генерации учебных планов на основе федеральных государственных образовательных стандартов и профессиональных стандартов. В качестве источников ФГОС использовался официальный портал

ФГОС России. Федеральный государственный образовательный стандарт (ФГОС) - совокупность обязательных требований к образованию определённого уровня.

В систему загружались нормативные документы в формате PDF, после чего выполнялся их автоматический анализ. Тестирование проводилось на нескольких направлениях подготовки и профессиональных стандартах:

– Бакалавриат 09.03.01 «Информатика и вычислительная техника»[13],

– Специалитет 54.05.02 «Живопись»,

– Магистратура 03.04.01 «Прикладные математика и физика»,

– Профессиональный стандарт 06.001 «Программист» [15],

– Профессиональный стандарт 06.003 «Архитектор программного обеспечения»,

– Профессиональный стандарт 04.002 «Специалист по техническим процессам художественной деятельности».

Тестирование охватывало как технические, так и гуманитарные направления подготовки, а также различные профессиональные области деятельности.

Для тестирования генерации учебного плана бакалавриата использовалось направление 09.03.01 «Информатика и вычислительная техника» и профессиональный стандарт 06.001 «Программист».

На рисунке 20 показано, как система корректно выполнила обработку ФГОС и профессионального стандарта. Были автоматически извлечены компетенции и трудовые функции, используемые при дальнейшей генерации учебного плана.

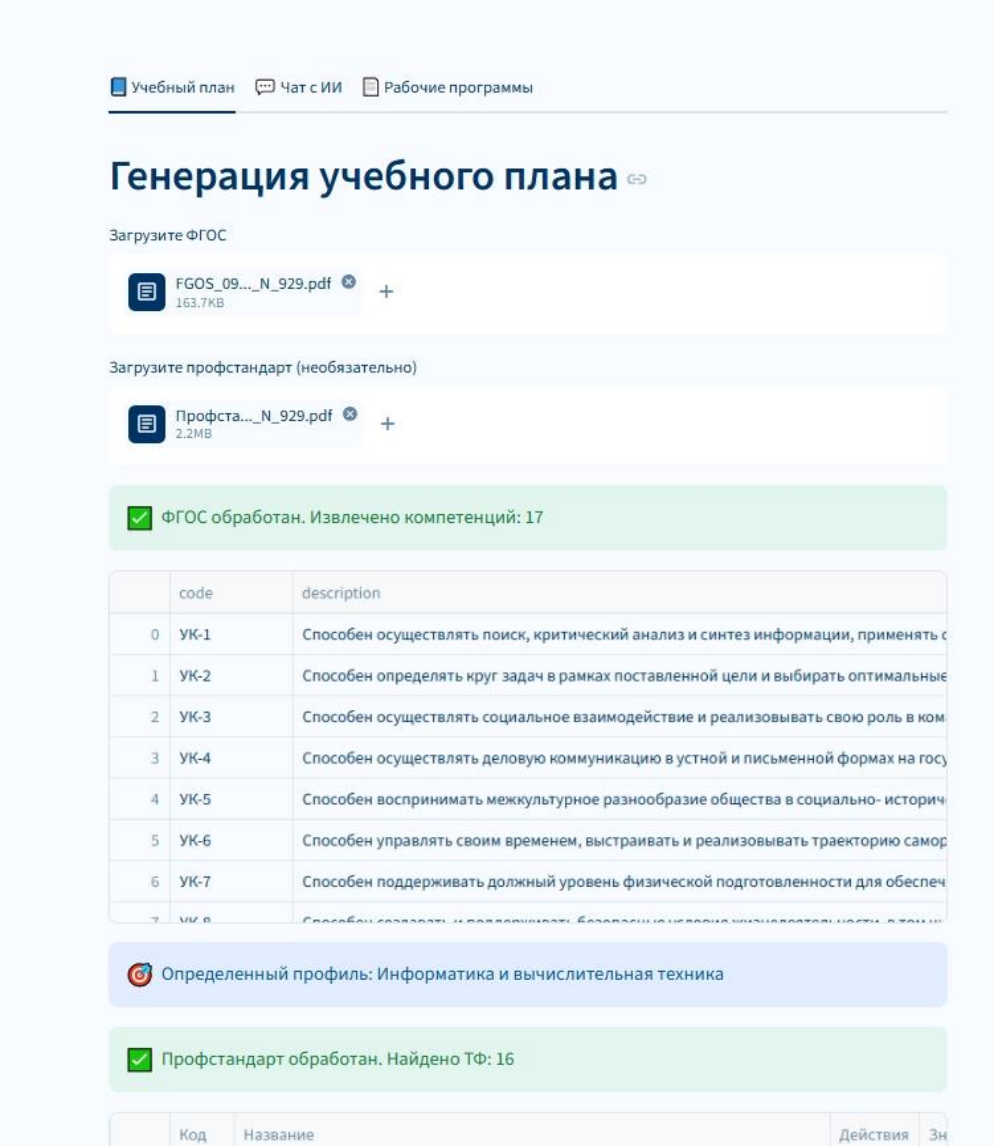


Рисунок 20 – Обработка ФГОС и профессионального стандарта для направления 09.03.01

В процессе обработки ФГОС направления 09.03.01 «Информатика и вычислительная техника» система автоматически извлекла 17 компетенций, включая универсальные (УК) и общепрофессиональные компетенции (ОПК). Дополнительно при анализе профессионального стандарта 06.001 «Программист» было выделено 16 трудовых функций, связанных с разработкой программного обеспечения, тестированием, интеграцией модулей и проектированием программных компонентов.

На основе извлечённых компетенций система формировала перечень дисциплин, соответствующих профилю подготовки в области информационных технологий и программной инженерии.

В сгенерированном учебном плане присутствовали дисциплины, характерные для ИТ-направлений подготовки: «Архитектура вычислительных систем», «Операционные системы», «Базы данных», «Системное программирование», «Теория вероятностей и математическая статистика». Это показывает, что система корректно учитывает специфику технического направления подготовки. Извлечённые системой компетенции ФГОС представлены в таблице 4.

Таблица 4 – Извлечённые компетенции ФГОС бакалавриата

code	description
УК-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач Разработка и реализация проектов
УК-2	Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений Командная работа и лидерство
УК-3	Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде Коммуникация
УК-4	Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(ых) языке(ах) Межкультурное взаимодействие
УК-5	Способен воспринимать межкультурное разнообразие общества в социально-историческом, этическом и философском контекстах Самоорганизация и саморазвитие (в том числе здоровьесбережение)
УК-6	Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни
УК-7	Способен поддерживать должный уровень физической подготовленности для обеспечения полноценной социальной и профессиональной деятельности Безопасность жизнедеятельности
УК-8	Способен создавать и поддерживать безопасные условия жизнедеятельности, в том числе при возникновении чрезвычайных ситуаций 3.3. Программа бакалавриата должна устанавливать следующие общепрофессиональные компетенции:
ОПК-1	Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности;

Продолжение таблицы 4

ОПК-2	Способен использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности;
ОПК-3	Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности;
ОПК-4	Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью; fgos.ru 14.05.2026
ОПК-5	Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем;
ОПК-6	Способен разрабатывать бизнес-планы и технические задания на оснащение отделов, лабораторий, офисов компьютерным и сетевым оборудованием;
ОПК-7	Способен участвовать в настройке и наладке программно-аппаратных комплексов;
ОПК-8	Способен разрабатывать алгоритмы и программы, пригодные для практического применения;
ОПК-9	Способен осваивать методики использования программных средств для решения практических задач. 3.4. Профессиональные компетенции определяются Организацией самостоятельно на основе профессиональных стандартов, соответствующих профессиональной деятельности выпускников (при наличии). При определении профессиональных компетенций на основе профессиональных стандартов Организация осуществляет выбор профессиональных стандартов, соответствующих профессиональной деятельности выпускников, из числа указанных в приложении к ФГОС ВО и (или) иных профессиональных стандартов, соответствующих профессиональной деятельности выпускников, из реестра профессиональных стандартов (перечня видов профессиональной деятельности), размещенного на специализированном сайте Министерства труда и социальной защиты Российской Федерации "Профессиональные стандарты" (http://profstandart.rosmintrud.ru) [3] (при наличии соответствующих профессиональных стандартов). [3] Пункт 1 приказа Министерства труда и социальной защиты Российской Федерации ...

Как видно из таблицы 4, система корректно извлекает как универсальные, так и общепрофессиональные компетенции из текста ФГОС. При этом были успешно обработаны длинные описания компетенций, содержащие сложные формулировки и дополнительные пояснения.

Особое значение имеет автоматическое выделение кодов компетенций (УК, ОПК, ПК), поскольку именно они используются при дальнейшем сопоставлении дисциплин и образовательных результатов. Результаты обработки профессионального стандарта представлены в таблице 5.

Таблица 5 – Извлечённые трудовые функции профстандарта

Код	Название	Действия	Знания	Умения
A/01.3	Формализация и алгоритмизация поставленных задач для разработки программного кода	2	0	0
A/02.3	Оформление программного кода в соответствии с установленными требованиями	1	1	1
A/03.3	Оформление программного кода в соответствии с установленными требованиями	1	1	1
A/04.3	Работа с системой управления версиями программного кода	1	1	1
A/05.3	Проверка работоспособности и рефакторинг кода программного обеспечения	2	2	3
B/01.4	Разработка процедур проверки работоспособности и измерения характеристик компьютерного программного	2	2	2
B/02.4	Проверка работоспособности компьютерного программного обеспечения	1	1	1
B/03.4	Проверка работоспособности и рефакторинг кода программного обеспечения	1	0	0
B/04.4	Рефакторинг, оптимизация и инспекция программного кода	3	3	3
B/05.4	Исправление дефектов программного кода, зафиксированных в базе данных дефектов	1	1	0
B/06.4	Интеграция программных модулей и компонентов и проверка работоспособности выпусков программного прод	2	2	3
C/01.5	Разработка процедур интеграции программных модулей	1	1	1
C/02.5	Осуществление интеграции программных модулей и компонентов и проверки работоспособности выпусков про	2	1	2

Продолжение таблицы 5

D/01.6	Анализ возможностей реализации требований к компьютерному программному обеспечению	1	1	1
D/02.6	Разработка технических спецификаций на программные компоненты и их взаимодействие	1	1	1
D/03.6	Проектирование компьютерного программного обеспечения	53	0	0

Согласно данным таблицы 5, система автоматически выделяет трудовые функции, знания, умения и действия, связанные с профессиональной деятельностью программиста. Наиболее часто встречались функции, связанные с разработкой программного кода, интеграцией программных модулей и проектированием программного обеспечения.

Полученные трудовые функции использовались не напрямую в учебном плане, а как дополнительный контекст для YandexGPT при генерации дисциплин и обосновании их включения в образовательную программу. Фрагмент автоматически сформированного учебного плана представлен в таблице 6.

Таблица 6 – Фрагмент сгенерированного учебного плана бакалавриата

Блок	Семестр	Дисциплина	Часы	Форма контроля	Компетенции ФГОС	Трудовые функции	Обоснование
Блок 1. Обязательная часть	1	Математический анализ	144	экзамен	['УК-1', 'ОПК-1', 'ОПК-2', 'ПК-3']	A/01.3	Дисциплина 'Математический анализ' развивает аналитические навыки и умение применять математические методы для решения задач, что важно для формализации и алгоритмизации задач при разработке программного кода.
Блок 1. Обязательная часть	1	Архитектура вычислительных систем	144	экзамен	['УК-1', 'УК-2', 'ОПК-2', 'ПК-5']	A/01.3	Дисциплина 'Архитектура вычислительных систем' формирует понимание принципов работы вычислительных систем, что необходимо для применения естественнонаучных и общеинженерных знаний (ОПК-1). Также она обучает использованию современных информационных технологий и программных средств (ОПК-2)

Продолжение таблицы 6

Блок 1. Обязательная часть	2	Дискретная математика	144	экзамен	['УК-1', 'ОПК-1', 'ПК-3', 'ПК-4']	A/01.3	Дисциплина 'Дискретная математика' развивает навыки анализа и синтеза информации, необходимые для формализации задач, а также умение использовать математические методы в профессиональной деятельности.
Блок 1. Обязательная часть	2	Операционные системы	144	зачёт	['УК-1', 'УК-2', 'ОПК-2', 'ОПК-3', 'ПК-6', 'ПК-5']	A/01.3	Дисциплина 'Операционные системы' формирует умение анализировать информацию и использовать современные технологии для решения задач, что соответствует УК-1 и ОПК-2. Трудовая функция A/01.3 поддерживает эти компетенции, так как требует формализации задач для разработки программного кода.
Блок 1. Обязательная часть	3	Теория вероятностей и математическая статистика	144	экзамен	['УК-1', 'ОПК-1', 'ПК-10', 'ПК-13']	A/01.3	Дисциплина 'Теория вероятностей и математическая статистика' развивает аналитические навыки и умение работать с данными, что важно для формализации задач и разработки алгоритмов.
Блок 1. Обязательная часть	3	Базы данных	144	экзамен	['УК-1', 'УК-2', 'ОПК-2', 'ОПК-3', 'ПК-8', 'ПК-9']	A/01.3	Дисциплина 'Базы данных' формирует ОПК-2 через изучение современных информационных технологий и программных средств, а также ОПК-1 через применение методов математического моделирования и анализа в контексте работы с базами данных. Трудовая функция A/01.3 поддерживает эти компетенции, так как требует формализации и алгоритмизации задач для разработки программного кода, связанного с базами данных.
Блок 1. Вариативная часть	3	Системное программирование	108	экзамен	['УК-1', 'УК-2', 'ОПК-2', 'ПК-6', 'ПК-5']	A/01.3	Дисциплина 'Системное программирование' формирует ОПК-2 через использование современных инструментов и технологий, а ОПК-1 – через применение математических методов и моделирования в программировании. Трудовая функция A/01.3 поддерживает эти компетенции, обучая алгоритмизации задач.

Как видно из таблицы 6, система способна автоматически формировать логически связанный учебный план с распределением дисциплин по семестрам, указанием форм контроля, часов и компетенций.

Базовые математические и инженерные дисциплины были размещены в первых семестрах, тогда как специализированные дисциплины, связанные с программированием и информационными системами, располагались на более поздних этапах обучения. Такое распределение соответствует типовой структуре образовательных программ технических направлений подготовки.

Дополнительно система автоматически формировала обоснование включения дисциплины в учебный план. В обосновании учитывались как компетенции ФГОС, так и трудовые функции профессионального стандарта. Это позволило показать связь между образовательной программой и профессиональной деятельностью выпускника.

Для проверки генерации учебного плана магистратуры использовалось направление 03.04.01 «Прикладные математика и физика». Был взят профстандарт Профессиональный стандарт 04.002 «Специалист по техническим процессам художественной деятельности». Результат генерации представлен на рисунке 21.

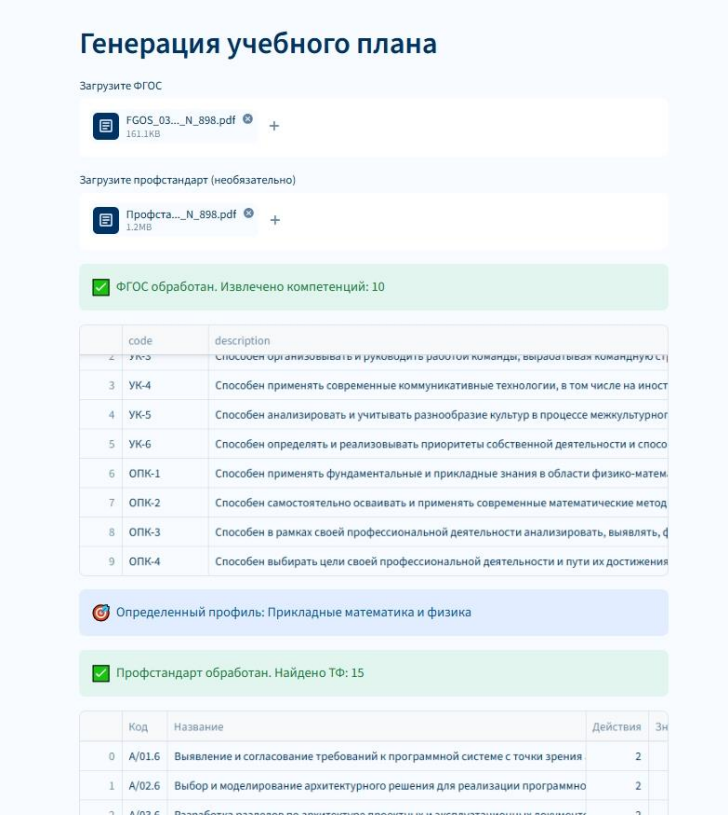


Рисунок 21 – Обработка ФГОС и профессионального стандарта для направления 03.04.01

При тестировании магистратуры система сформировала учебный план с более исследовательской и научной направленностью. В отличие от бакалавриата, в учебном плане преобладали дисциплины, связанные с математическим моделированием, анализом данных, вычислительными методами и специализированными направлениями прикладной физики.

Количество компетенций оказалось меньше по сравнению с бакалавриатом, однако сами формулировки были значительно сложнее и содержали более высокий уровень профессиональной специализации.

Для тестирования специалитета использовалось направление 54.05.02 «Живопись» и профессиональный стандарт 06.003 «Архитектор программного обеспечения». Результат обработки нормативной документации для направления представлен на рисунке 22.

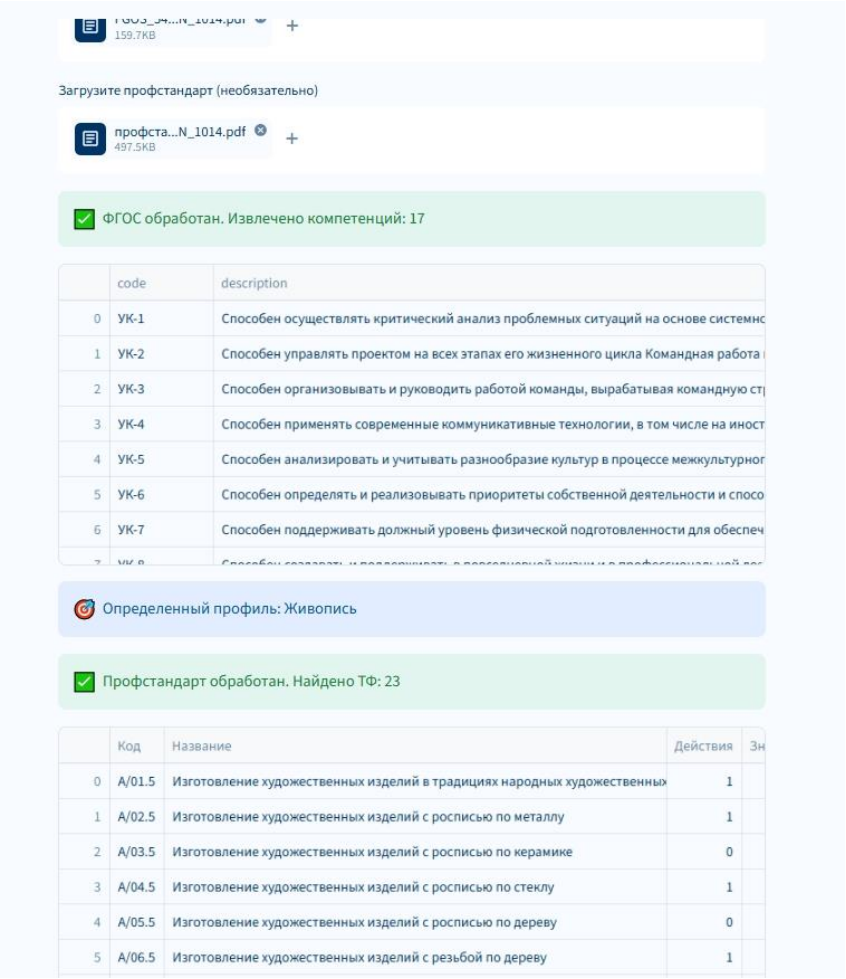


Рисунок 22 – Обработка ФГОС и профессионального стандарта для направления 54.05.02

Для направления 54.05.02 «Живопись» система сформировала существенно отличающийся набор дисциплин. В учебном плане преобладали дисциплины художественного и творческого профиля, связанные с академическим рисунком, живописью, композицией, историей искусства и художественными технологиями.

Это показывает, что система учитывает особенности гуманитарных и творческих направлений подготовки и не использует одинаковые шаблоны дисциплин для разных образовательных областей.

Дополнительно было установлено, что при обработке гуманитарных направлений структура компетенций и профессиональных стандартов отличается от технических специальностей, однако система успешно адаптируется к различным типам нормативной документации.

Таблица 7 – Результаты тестирования системы

Направление	Уровень	Компетенций	Трудовых функций	Сгенерировано дисциплин
09.03.01	Бакалавриат	17	16	48
03.04.01	Магистратура	10	15	35
54.05.02	Специалитет	17	23	35

Как видно из таблицы 7, система успешно обрабатывает нормативные документы различных уровней подготовки: бакалавриата, магистратуры и специалитета.

Наибольшее количество дисциплин было сформировано для направления 09.03.01 «Информатика и вычислительная техника», что связано с большим количеством профессиональных компетенций и трудовых функций в области разработки программного обеспечения.

Для творческого направления количество дисциплин оказалось меньше, однако структура учебного плана сохранила соответствие профилю подготовки.

Полученные результаты подтверждают возможность использования системы для автоматизированной подготовки предварительных учебных планов в различных предметных областях.

4.3 Тестирование генерации РПД

На рисунке 23 представлен интерфейс системы генерации рабочих программ дисциплин. Пользователь выбирает дисциплину, направление подготовки, профиль, форму обучения, университет и факультет. После заполнения параметров система формирует JSON-структуру, содержащую сведения, необходимые для генерации рабочей программы дисциплины.

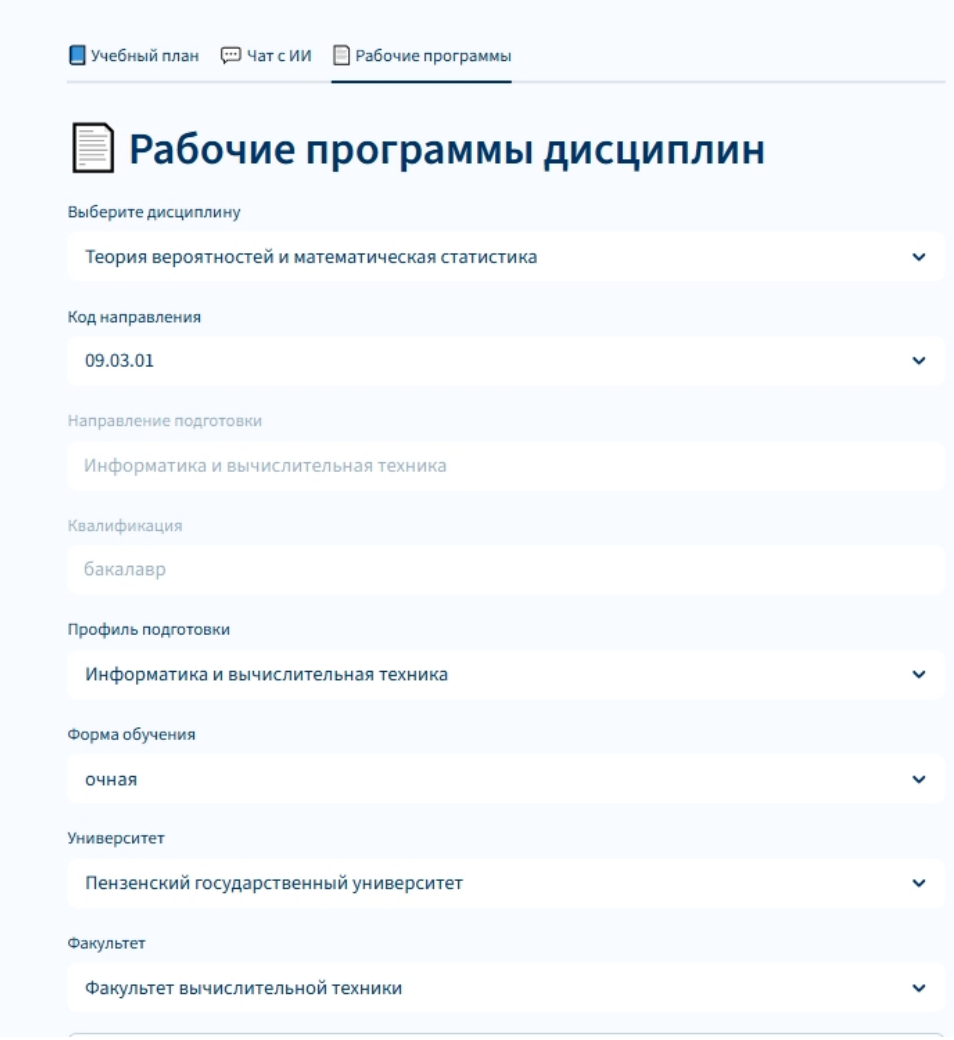


Рисунок 23 – Интерфейс генерации рабочей программы дисциплины.

В рассматриваемом примере выполнялась генерация рабочей программы дисциплины «Теория вероятностей и математическая статистика» для направления подготовки 09.03.01 «Информатика и вычислительная

техника». В качестве образовательной организации был указан Пензенский государственный университет.

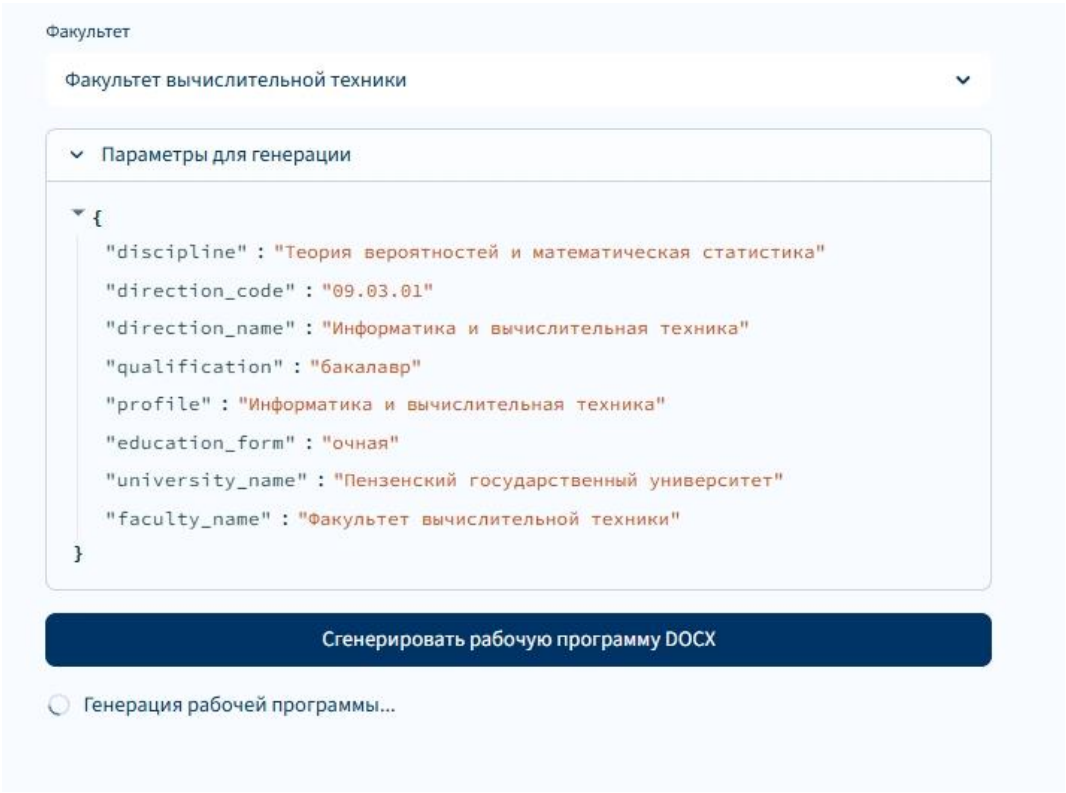


Рисунок 24 – Формирование параметров генерации рабочей программы дисциплины

После заполнения параметров система автоматически сформировала рабочую программу дисциплины в формате DOCX. Полный пример автоматически сформированной рабочей программы дисциплины представлен в приложении А. Сгенерированный документ включает основные структурные элементы РПД:

– титульный лист (представлен на рисунке 25),

– цели освоения дисциплины,

– место дисциплины в структуре ОПОП,

– компетенции,

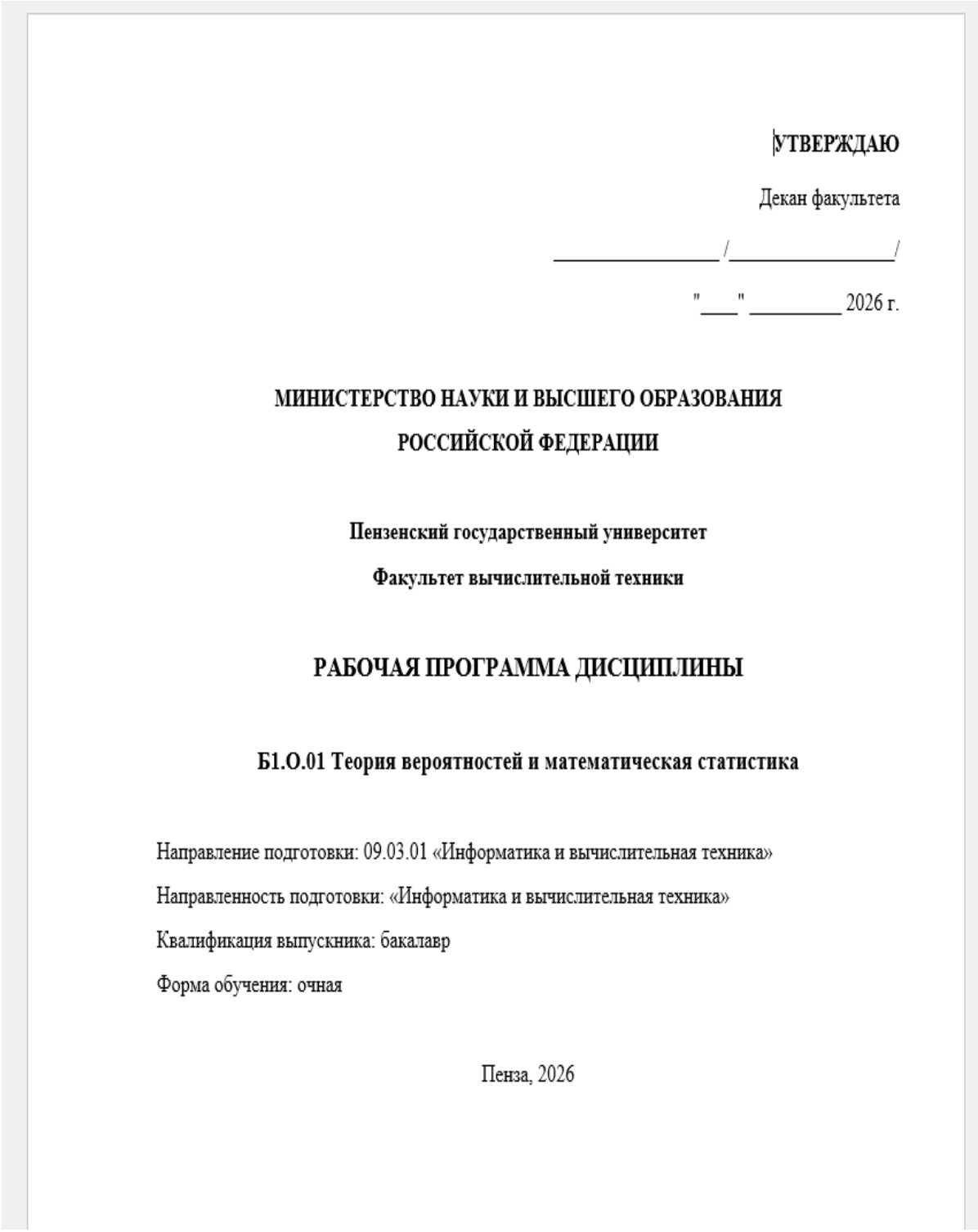
– тематический план (представлен на рисунке 26),

– содержание дисциплины,

– перечень практических занятий,

– формы контроля,

– учебно-методическое обеспечение.



УТВЕРЖДАЮ

Декан факультета

_____ /_____/

"____" _____ 2026 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Пензенский государственный университет

Факультет вычислительной техники

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.О.01 Теория вероятностей и математическая статистика

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Направленность подготовки: «Информатика и вычислительная техника»

Квалификация выпускника: бакалавр

Форма обучения: очная

Пенза, 2026

Рисунок 25 – Титульный лист автоматически сгенерированной рабочей программы дисциплины

4. Структура и содержание дисциплины «Теория вероятностей и математическая статистика»

Общая трудоемкость дисциплины составляет 4 зачетных единиц, 144 часов.

Раздел	Тема	Семестр	Недели	Лекции	Лаб./практ.	Иное	СРС	Текущий контроль	Промежуточный контроль
Раздел 1. Основы теории вероятностей	Тема 1.1. Основные понятия теории вероятностей	3	1-2	2	2	0	6	Опрос	экзамен
Раздел 2. Случайные величины	Тема 2.1. Дискретные случайные величины	3	3-4	2	2	0	6	Проверка практических работ	экзамен
Раздел 3. Математическая статистика	Тема 3.1. Выборочный метод	3	5-6	2	2	0	6	Опрос	экзамен
Раздел 4. Статистические методы анализа данных	Тема 4.1. Проверка статистических гипотез	3	7-8	2	2	0	6	Проверка практических работ	экзамен

Рисунок 26 – Структура автоматически сгенерированной рабочей программы дисциплины

4. Структура и содержание дисциплины «Теория вероятностей и математическая статистика»

4.1. Структура дисциплины

Общая трудоемкость дисциплины составляет 3 зачетных единиц, 108 часа.

№ п/п	Наименование разделов и тем дисциплины (модуля)	Семестр	Недели семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)									Формы текущего контроля успеваемости *(по неделям семестра)*							
				Контактная работа					Самостоятельная работа											
				Всего	Лекции	Практические занятия	Лабораторные занятия	Др. виды контакт. работы	Всего	Курсовая работа (проект)	Подготовка к экзамену	Другие виды самостоятельной работы	Собеседование	Коллоквиум	Проверка тестов	Проверка контрольн. работ	Проверка реферата	Проверка эссе и иных творческих работ	курсовая работа (проект)	др.
1.	Раздел 1. *Случайные события*	**3**	**1-4**	**15**	**4**	**8**		**3**	**16**		**12**	**4**				5				
1.1.	Тема 1.1. Основные понятия комбинаторики. Основные понятия и теоремы теории вероятностей	3	1-2	7	3	4			8		6	2								
1.2.	Тема 1.2. Схема повторных независимых испытаний	3	3-4	5	1	4			8		6	2								
2.	Раздел 2. *Случайные величины*	**3**	**5-11**	**23**	**9**	**14**			**18**		**12**	**6**				12				
2.1.	Тема 2.1. Случайные величины и их числовые характеристики	3	5-6	7	3	4			4		3	1								
2.2.	Тема 2.2. Основные законы распределения	3	7-8	7	3	4			4		3	1								
2.3.	Тема 2.3. Многомерные случайные величины	3	9-10	5	1	4			6		3	3								
2.4.	Тема 2.4. Закон больших чисел и предельные теоремы.	3	11	4	2	2			4		3	1								
3.	Раздел 3. *Математическая статистика*	**3**	**12-17**	**16**	**4**	**12**			**20**		**12**	**8**								
3.1.	Тема 3.1. Эмпирические характеристики и выборки	3	12-13	6	2	4			6		4	2								
3.2.	Тема 3.2. Точечные и интервальные оценки	3	14-15	6	2	4			6		4	2								
3.3.	Тема 3.3. Статистическая проверка гипотез	3	16-17	4		4			8		4	4								
	Подготовка к экзамену										**36**									
	Общая трудоемкость, в часах			**54**	**17**	**34**		**3**	**54**		**36**	**18**	Промежуточная аттестация							
													Форма		Семестр					
													Экзамен		3					

Рисунок 27 – Структура реальной рабочей программы дисциплины ПГУ

Сгенерированная рабочая программа была сформирована автоматически на основе данных ФГОС, профессионального стандарта и параметров учебного направления.

Для оценки качества генерации была выполнена сравнительная проверка автоматически сформированной рабочей программы дисциплины с реальной рабочей программой Пензенского государственного университета по дисциплине «Теория вероятностей и математическая статистика».

Сравнение показало, что структура автоматически сформированного документа соответствует структуре реальной рабочей программы и включает аналогичные разделы:

– цели освоения дисциплины,

– результаты освоения дисциплины,

– тематическое планирование,

– содержание лекционных и практических занятий,

– формы текущего и промежуточного контроля,

– перечень литературы и образовательных технологий.

В сгенерированном документе были автоматически сформированы компетенции, тематический план дисциплины, перечень практических занятий и структура самостоятельной работы студентов. При этом система обеспечила корректное заполнение сведений о направлении подготовки, квалификации и образовательной организации.

Анализ показал, что автоматически сформированная РПД имеет упрощённую структуру по сравнению с официальным документом университета. В реальной рабочей программе присутствуют дополнительные элементы:

– подробное тематическое планирование,

– расширенный перечень литературы,

– интернет-ресурсы,

– методические указания,

– листы согласования и переутверждения программы.

Разработанная система успешно формирует основное содержательное наполнение РП дисциплины и обеспечивает существенное сокращение времени подготовки документации, хоть и имеются недостатки.

Таблица 8 – Сравнение реальной и сгенерированной РПД

Критерий	Реальная РПД	Сгенерированная РПД
Титульный лист	Да	Да
Компетенции	Да	Да
Тематический план	Да	Да
Практические занятия	Да	Да
Самостоятельная работа	Да	Да
Методические указания	Да	Частично
Интернет-ресурсы	Да	Нет
Листы согласования	Да	Нет
Автоматическая генерация	Нет	Да
Время подготовки	Высокое	Низкое

В таблице 8 можно заметить, что разработанная система обеспечивает автоматическое формирование основных структурных элементов рабочей программы дисциплины. Качество получаемого документа является достаточным для дальнейшей доработки и использования в образовательной организации. Автоматически сформированная рабочая программа дисциплины по своей структуре и содержанию близка к реальным вузовским документам. Система успешно формирует основные разделы РПД, включая компетенции, тематический план, формы контроля и учебно-методическое обеспечение. Несмотря на наличие отдельных упрощений по сравнению с официальными документами университета, система позволяет сократить время подготовки РПД и уменьшить объем ручной работы методиста.

При тестировании выяснилось, что разработанная система способна автоматически формировать рабочие программы дисциплин на основе параметров образовательной программы и нормативной документации.

Система обеспечивает:

– автоматизацию подготовки РПД,

– снижение трудоёмкости оформления документации,

– унификацию структуры рабочих программ,

– ускорение процесса разработки учебно-методических материалов.

Проведённое сравнение с реальной рабочей программой университета подтвердило корректность формирования основных разделов документа и возможность практического применения разработанного решения в образовательных организациях.

4.4 Тестирование ИИ-консультанта

Дополнительно было проведено тестирование встроенного ИИ-консультанта, предназначенного для взаимодействия пользователя с системой на естественном языке. Целью тестирования являлась проверка корректности обработки текстовых запросов, устойчивости генерации и возможности автоматического изменения параметров учебного плана без ручного редактирования таблиц.

В отличие от стандартного редактирования Excel-таблиц, пользователь может взаимодействовать с системой через текстовые запросы. ИИ-консультант принимает команды на естественном языке, после чего система пытается преобразовать их в структурированные изменения учебного плана.

Во время тестирования проверялись следующие типы запросов:

– добавление новых дисциплин,

– удаление дисциплин,

– изменение формы контроля,

– изменение количества часов,

– перенос дисциплин между семестрами,

– комбинированное изменение параметров учебного плана.

Для оценки качества работы консультанта использовались тестовые пользовательские запросы. Результаты тестирования представлены в таблице 9.

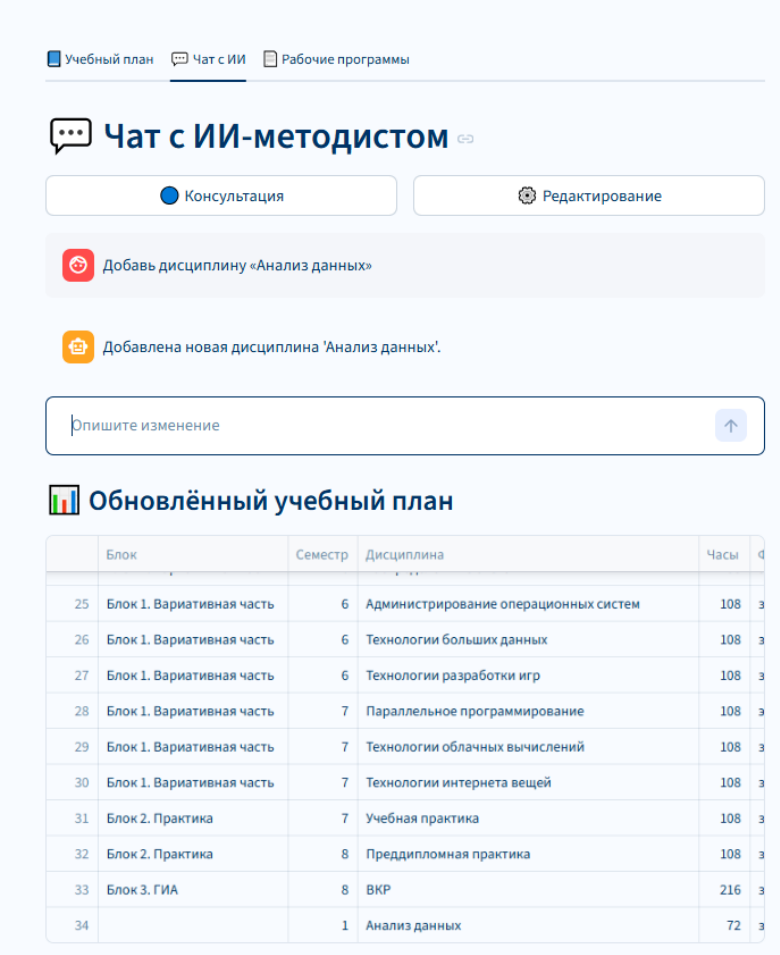
Таблица 9 – Запросы

Запрос пользователя	Ожидаемый результат	Фактический результат	Статус
Добавь дисциплину «Анализ данных»	Добавление новой дисциплины	Дисциплина добавлена	Успешно
Удали дисциплину «Разработка игр»	Удаление дисциплины	Дисциплина удалена	Успешно
Измени форму контроля по дисциплине «Базы данных» на экзамен	Изменение формы контроля	Форма контроля изменена	Успешно
Установи 108 часов для дисциплины «Системное программирование»	Изменение часов	Часы изменены	Успешно
Перенеси «Архитектура вычислительных систем» на 4 семестр	Изменение семестра	Семестр не изменён	Ошибка
Добавь экзамен по дисциплине «Операционные системы»	Изменение формы контроля	Добавлена новая дисциплина	Ошибка
Увеличь часы по математическому анализу	Требуется уточнение значения	Часы обновлены некорректно	Ошибка
Удали несуществующую дисциплину «Квантовая биология»	Сообщение об ошибке	Система сообщила, что дисциплина не найдена	Успешно
Добавь дисциплину «Машинное обучение» на 5 семестр с зачётом	Добавление дисциплины с параметрами	Дисциплина добавлена с параметрами	Успешно
Перенеси «Базы данных» на 4 семестр и поставь экзамен	Комбинированное изменение	Выполнено частично	Ошибка

По результатам тестирования корректно были обработаны 6 из 10 запросов, что составляет около 60 % от общего количества тестовых сценариев. Наиболее успешно система справлялась с простыми операциями

добавления и удаления дисциплин, а также с изменением отдельных параметров учебного плана.

Пример взаимодействия пользователя с ИИ-консультантом представлен на рисунке 28.



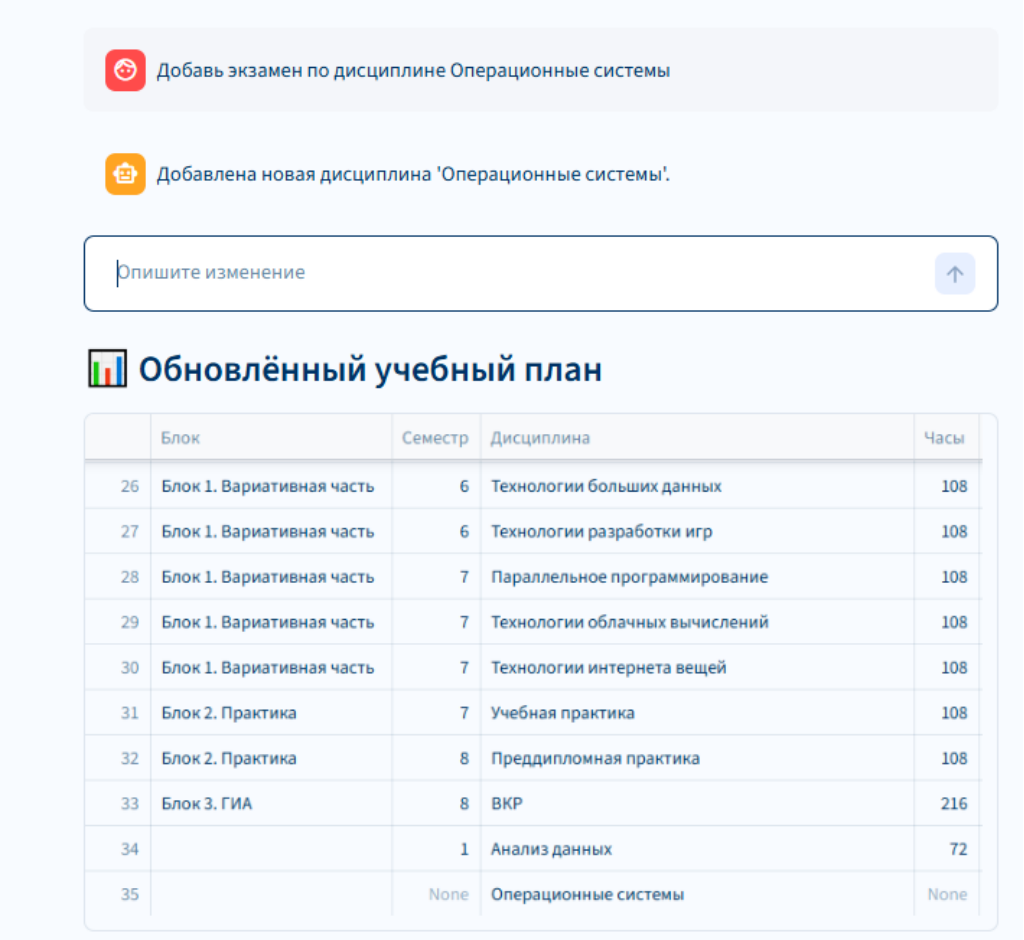
	Блок	Семестр	Дисциплина	Часы
25	Блок 1. Вариативная часть	6	Администрирование операционных систем	108
26	Блок 1. Вариативная часть	6	Технологии больших данных	108
27	Блок 1. Вариативная часть	6	Технологии разработки игр	108
28	Блок 1. Вариативная часть	7	Параллельное программирование	108
29	Блок 1. Вариативная часть	7	Технологии облачных вычислений	108
30	Блок 1. Вариативная часть	7	Технологии интернета вещей	108
31	Блок 2. Практика	7	Учебная практика	108
32	Блок 2. Практика	8	Преддипломная практика	108
33	Блок 3. ГИА	8	ВКР	216
34		1	Анализ данных	72

Рисунок 28 – Взаимодействие пользователя с ИИ-консультантом

Однако тестирование также выявило ряд ограничений, связанных с использованием генеративной языковой модели. Наибольшее количество ошибок возникало при обработке неоднозначных и комбинированных запросов. В некоторых случаях модель некорректно интерпретировала намерение пользователя и формировала неправильное изменение структуры учебного плана.

Например, при попытке изменить форму контроля дисциплины «Операционные системы» система не изменила существующую запись, а создала новую дисциплину с аналогичным названием. При этом часть

параметров новой записи осталась незаполненной. Результат представлен на рисунке 29.



	Блок	Семестр	Дисциплина	Часы
26	Блок 1. Вариативная часть	6	Технологии больших данных	108
27	Блок 1. Вариативная часть	6	Технологии разработки игр	108
28	Блок 1. Вариативная часть	7	Параллельное программирование	108
29	Блок 1. Вариативная часть	7	Технологии облачных вычислений	108
30	Блок 1. Вариативная часть	7	Технологии интернета вещей	108
31	Блок 2. Практика	7	Учебная практика	108
32	Блок 2. Практика	8	Преддипломная практика	108
33	Блок 3. ГИА	8	ВКР	216
34		1	Анализ данных	72
35		None	Операционные системы	None

Рисунок 29 – Попытка изменить форму контроля дисциплины «Операционные системы».

Дополнительные ошибки были выявлены при изменении количества часов и переносе дисциплин между семестрами. В отдельных случаях система изменяла неверный элемент учебного плана либо выполняла запрос частично.

Причиной подобных ошибок является неоднозначность интерпретации текстовых команд языковой моделью. Несмотря на использование JSON-структур и дополнительных ограничений генерации, модель всё ещё может некорректно определять контекст запроса или формировать ошибочные изменения данных.

Во время тестирования также было установлено, что обработка запросов через API языковой модели занимает сравнительно большое время. При выполнении сложных запросов генерация ответа происходила заметно

медленнее по сравнению с ручным редактированием таблицы. В ряде случаев прямое редактирование учебного плана оказывалось более эффективным, чем использование ИИ-консультанта.

Тем не менее реализованный механизм взаимодействия на естественном языке продемонстрировал возможность применения генеративных моделей для упрощения работы с образовательной документацией. Использование ИИ-консультанта позволяет сократить объём рутинного редактирования и упростить взаимодействие пользователя с системой.

Полученные результаты показывают, что генеративные модели могут эффективно использоваться как вспомогательный инструмент при проектировании учебных планов, однако полностью заменять ручное редактирование на текущем этапе преждевременно. Система требует дальнейшей доработки механизмов обработки пользовательских запросов, повышения устойчивости генерации и улучшения контроля корректности изменений учебного плана.

4.5 Оценка эффективности системы

После завершения тестирования была выполнена оценка эффективности разработанной интеллектуальной системы. Основной задачей исследования являлось определение того, насколько использование автоматизированной генерации учебных материалов позволяет сократить трудозатраты методистов и ускорить подготовку документации по сравнению с традиционным ручным подходом.

В рамках оценки сравнивались:

– ручное формирование учебного плана,

– автоматическая генерация учебного плана системой,

– ручная подготовка рабочей программы дисциплины,

– автоматическая генерация РПД,

– изменение учебного плана вручную и через ИИ-консультанта.

При традиционном подходе подготовка учебного плана требует анализа ФГОС, профессиональных стандартов, распределения дисциплин по

семестрам, сопоставления компетенций и проверки форм контроля. Значительная часть времени уходит на ручную обработку нормативных документов и заполнение таблиц.

Использование разработанной системы позволило автоматизировать большую часть этих операций. После загрузки ФГОС и профессиональных стандартов система автоматически:

– извлекает компетенции,

– анализирует трудовые функции,

– формирует перечень дисциплин,

– распределяет дисциплины по семестрам,

– генерирует формы контроля;

– создаёт предварительную структуру учебного плана;

– формирует рабочие программы дисциплин в формате DOCX.

Во время экспериментального исследования было установлено, что автоматическая генерация учебного плана занимает значительно меньше времени по сравнению с ручной подготовкой. В среднем генерация предварительного варианта учебного плана выполнялась за несколько минут, тогда как ручная подготовка аналогичной структуры может занимать от нескольких дней до нескольких недель в зависимости от сложности направления подготовки.

Аналогичная ситуация наблюдалась при генерации рабочих программ дисциплин. Система автоматически формировала титульный лист, компетенции, тематическое планирование, перечень практических занятий и учебно-методическое обеспечение. Подготовка РПД вручную обычно требует значительных временных затрат, связанных с оформлением таблиц, копированием структуры документа и заполнением однотипных разделов.

Результаты сравнительной оценки представлены в таблице 10.

Таблица 10 – Сравнение ручной и автоматизированной подготовки документов

Критерий	**Ручная подготовка**	**Разработанная система**
Анализ ФГОС	полностью вручную	автоматически
Извлечение компетенций	вручную	автоматически
Формирование дисциплин	вручную	автоматически
Распределение по семестрам	вручную	автоматически
Формирование РПД	вручную	автоматически
Генерация DOCX	вручную	автоматически
Время подготовки учебного плана	от 60 дней	3–7 минут
Время подготовки РПД	от 14 дней	3–7 минут
Необходимость ручного редактирования	высокая	частичная
Вероятность ошибок	высокая	снижена

Полученные результаты показывают, что разработанная система позволяет существенно сократить объём рутинных операций при подготовке учебно-методической документации. Наиболее заметный эффект достигается за счёт автоматизации обработки нормативных документов, извлечения компетенций и формирования предварительной структуры рабочих программ дисциплин.

При этом результаты тестирования показали, что полностью исключить участие методиста на текущем этапе невозможно. Отдельные элементы документа требуют дополнительной проверки и корректировки со стороны пользователя, включая уточнение формулировок дисциплин, проверку распределения часов, редактирование тематического плана и контроль корректности структуры РПД.

Дополнительно было установлено, что использование ИИ-консультанта для редактирования учебного плана не всегда оказывается эффективнее ручного изменения таблицы. В отдельных случаях обработка текстовых

команд занимала больше времени, а результаты генерации требовали дополнительной проверки. Это связано с ограничениями генеративных языковых моделей и возможностью неоднозначной интерпретации пользовательских запросов.

Тем не менее использование NLP-технологий и генеративных моделей позволило значительно ускорить процесс формирования учебных планов и рабочих программ дисциплин, уменьшить количество рутинных операций и упростить работу с нормативной документацией. Таким образом, результаты экспериментального исследования подтверждают возможность практического применения разработанной интеллектуальной системы в образовательных организациях в качестве инструмента поддержки проектирования образовательных программ и подготовки учебно-методической документации

4.6 Выводы

В четвёртой главе было проведено экспериментальное исследование разработанной интеллектуальной системы автоматизации формирования учебно-методической документации. В ходе тестирования проверялась корректность обработки нормативных документов, генерации учебных планов, формирования рабочих программ дисциплин и работы ИИ-консультанта.

Результаты тестирования показали, что система успешно обрабатывает ФГОС и профессиональные стандарты различных направлений подготовки, автоматически извлекает компетенции и трудовые функции, а также формирует предварительную структуру учебного плана с учётом профиля подготовки.

Проведённое сравнение автоматически сформированных рабочих программ дисциплин с реальными вузовскими РПД подтвердило корректность формирования основных структурных элементов документа, включая компетенции, тематическое планирование и формы контроля.

Дополнительно было установлено, что использование ИИ-консультанта позволяет выполнять часть изменений учебного плана на естественном языке,

однако качество обработки сложных и комбинированных запросов пока остаётся ограниченным.

Экспериментальное исследование показало, что применение NLP-технологий и генеративных моделей позволяет существенно сократить время подготовки учебно-методической документации и уменьшить объём рутинной работы методистов. При этом результаты генерации требуют дополнительного экспертного контроля со стороны пользователя.

ЗАКЛЮЧЕНИЕ

В ходе выполнения магистерской диссертации была разработана интеллектуальная система автоматизированного формирования образовательных материалов на основе ФГОС и профессиональных стандартов с использованием технологий обработки естественного языка и генеративных моделей.

Актуальность работы обусловлена высокой трудоёмкостью подготовки учебно-методической документации в образовательных организациях. Формирование УП и рабочих программ дисциплин требует анализа большого объема нормативных документов, сопоставления компетенций, трудовых функций и распределения дисциплин по семестрам. Ручное выполнение данных задач занимает значительное количество времени и повышает вероятность возникновения ошибок.

В рамках исследования были проанализированы существующие подходы к автоматизации образовательных программ, методы обработки нормативных документов, а также возможности применения NLP и больших языковых моделей для решения задач проектирования образовательных программ.

В ходе работы были решены задачи анализа существующих подходов к автоматизации формирования учебно-методической документации, исследования методов NLP и генеративных моделей, разработки архитектуры интеллектуальной системы, реализации модулей генерации учебных планов и рабочих программ дисциплин, создания пользовательского веб-интерфейса, а также проведения экспериментального тестирования разработанной системы.

В результате работы была разработана архитектура интеллектуальной системы, включающая:

– модуль анализа ФГОС,

– модуль обработки профессиональных стандартов,

– модуль извлечения компетенций,

– модуль генерации дисциплин,

– модуль формирования учебного плана,

– модуль генерации рабочих программ дисциплин,

– пользовательский веб-интерфейс,

– модуль взаимодействия с ИИ-консультантом.

Система реализована на языке Python с использованием Streamlit, pandas, openpyxl, python-docx, requests и YandexGPT API. Для обработки нормативных документов применялись методы NLP, регулярные выражения, JSON-структуры и эвристические алгоритмы распределения дисциплин.

Научная новизна работы заключается в разработке и реализации двухступенчатой методики генерации образовательных материалов, при которой облегчённая языковая модель используется для извлечения и первичного структурирования данных, а полная модель – для формирования содержательной части учебных планов и рабочих программ дисциплин. Использование данного подхода позволило повысить устойчивость генерации и сократить количество ошибок обработки данных.

В ходе экспериментального исследования было проведено тестирование системы на различных направлениях подготовки и профессиональных стандартах. Проверялась обработка ФГОС бакалавриата, магистратуры и специалитета, а также генерация учебных планов и рабочих программ дисциплин. Результаты тестирования показали, что система:

– автоматически формирует перечень дисциплин,

– корректно извлекает компетенции и трудовые функции,

– распределяет дисциплины по семестрам,

– генерирует структуры учебных планов,

– формирует рабочие программы дисциплин в формате DOCX,

– поддерживает взаимодействие с пользователем через ИИ-консультанта.

Дополнительно было проведено сравнение автоматически сформированной рабочей программы дисциплины с реальной рабочей

программой Пензенского государственного университета. Сравнение показало, что структура сгенерированного документа соответствует основным требованиям учебно-методической документации и включает ключевые разделы РПД.

При оценке эффективности системы было установлено существенное сокращение времени подготовки образовательных материалов по сравнению с полностью ручным подходом. Автоматическая генерация учебного плана и рабочих программ дисциплин выполнялась за несколько минут, тогда как подготовка аналогичной документации вручную может занимать от нескольких часов до нескольких дней.

При этом результаты тестирования ИИ-консультанта показали, что использование генеративных моделей для изменения структуры учебного плана пока требует дополнительной доработки. В отдельных случаях наблюдалась некорректная интерпретация пользовательских запросов, ошибки изменения параметров дисциплин и увеличение времени обработки запросов. Это подтверждает необходимость сохранения экспертного контроля со стороны методиста при использовании подобных систем.

Таким образом, поставленная цель исследования была достигнута, а все задачи, сформулированные во введении, успешно решены. Результаты экспериментального исследования подтвердили гипотезу о том, что использование технологий обработки естественного языка и больших языковых моделей позволяет существенно сократить время подготовки учебно-методической документации при сохранении структуры и содержательной полноты формируемых документов.

Практическая значимость работы заключается в возможности использования разработанной системы в образовательных организациях для сокращения трудозатрат методистов и ускорения подготовки учебно-методической документации.

Перспективы дальнейшего развития системы связаны:

– с улучшением алгоритмов обработки пользовательских запросов,

– повышением точности работы ИИ-консультанта,

– расширением поддержки различных образовательных стандартов,

– автоматической проверкой соответствия учебных планов требованиям ФГОС,

– развитием механизмов валидации и контроля корректности генерации образовательных материалов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дзусова Б. Т. Федеральный государственный образовательный стандарт как гарантия качества национального образования / Б. Т. Дзусова // Личность в меняющемся мире: здоровье, адаптация, развитие. – 2020. – Т. 8, № 2 (29). – С. 172–181. – DOI: 10.23888/humJ20202172-181.

2. Красноперова Л. Б. Методические рекомендации по сопоставлению ФГОС и профессиональных стандартов / Л. Б. Красноперова.

3. Киреева Д. Д. Автоматизированное формирование учебно-методических материалов кафедры по ФГОС 3++ / Д. Д. Киреева, А. Г. Кравец // Системный анализ в науке и образовании. – 2021. – № 1. – С. 1–7.

4. MMIS «Планы» [Электронный ресурс]. – Режим доступа: https://www.mmis.ru/programs/plany (дата обращения: 20.05.2026).

5. 1С:Университет ПРОФ [Электронный ресурс]. – Режим доступа: [https://solutions.1c.ru/catalog/universityprof](https://solutions.1c.ru/catalog/university-prof)(дата обращения: 20.05.2026).

6. Сафонов Е. И. Автоматизация составления рабочих программ учебных дисциплин / Е. И. Сафонов, А. Ю. Чернавский // Вестник Югорского государственного университета. – 2019. – Т. 15, № 2. – С. 33–40.

7. Мельников А. Об использовании машинного обучения в задачах обработки естественного языка на примере анализа образовательного контента / А. Мельников, Д. Ботов, Ю. Кленин // Онтология проектирования. – 2017. – Т. 7, № 1 (23). – С. 34–47.

8. Streamlit для Data Science: создание интерактивных приложений на Python [Электронный ресурс]. – Режим доступа: https://znanium.ru/catalog/document?id=462900 (дата обращения: 20.05.2026).

9. Python-docx: библиотека для работы с документами Word в Python [Электронный ресурс]. – Режим доступа: [https://docs-python.ru/packages/modul-python-docx-python/brief-description/](https://docs-

python.ru/packages/modul-python-docx-python/brief-description/) (дата обращения: 20.05.2026).

10. Работа с файлами XLSX при помощи модуля openpyxl [Электронный ресурс]. – Режим доступа: https://docs-python.ru/packages/modul-openpyxl/ (дата обращения: 20.05.2026).

11. Python и анализ данных [Электронный ресурс] // Habr. – Режим доступа: https://habr.com/ru/articles/415725/ (дата обращения: 20.05.2026).

12. Феникс Д. Промпт-инжиниринг для GenAI. Паттерны надежных запросов для качественных результатов / Д. Феникс, М. Тейлор. – [Б. м.] : O'Reilly, 2026. – 432 с. – ISBN 978-601-08-4722-4.

13. Федеральный государственный образовательный стандарт высшего образования по направлению подготовки 09.04.01 Информатика и вычислительная техника (уровень магистратуры) [Электронный ресурс] // Федеральный реестр государственных образовательных стандартов. – 2024. – Режим доступа: https://fgos.ru/ (дата обращения: 03.04.2026).

14. Радковский С. А. Использование фреймворка Streamlit для создания виртуальных лабораторных работ в виде веб-приложений / С. А. Радковский, Г. В. Коваленко // CyberLeninka. – Режим доступа: https://cyberleninka.ru/article/n/ispolzovanie-freymvorka-streamlit... (дата обращения: 03.04.2026).

15. Профессиональный стандарт «Программист» [Электронный ресурс] // Classinform. – Режим доступа: https://classinform.ru/profstandarty/06.001-programmist.html (дата обращения: 03.04.2026).

16. Moodle LMS [Электронный ресурс]. – Режим доступа: https://moodle.org/ (дата обращения: 20.05.2026).

17. Кузьменко О. Промптология. Искусство диалога с нейросетями [Электронный ресурс] / Олег Кузьменко. – [Б. м. : б. и.], 2025. – 1 с. – ISBN 978-5-9729-2715-9. – Режим доступа: по подписке.

18. Андреев А. А. Дистанционное обучение: сущность, технология, организация / А. А. Андреев, В. И. Солдаткин. – Москва : МЭСИ, 1999. – 196 с.

19. Вымятнин В. М. Мультимедиа курсы: методология и технология разработки / В. М. Вымятнин, В. П. Демкин, Г. В. Можаева, Т. В. Руденко // Открытое и дистанционное образование. – 2002. – № 3 (7). – С. 34–61.

20. Денисов Д. В. Автоматизация управления качеством образования в вузе [Электронный ресурс] / Д. В. Денисов. – Москва : Синергия, 2020. – Режим доступа: https://www.litres.ru/book/d-v-denisov/avtomatizaciya-upravleniya-kachestvomobrazovaniya-v-vuze-4991115/ (дата обращения: 05.06.2025).

21. Захарова И. Г. Информационные технологии в образовании : учебное пособие для студентов высших педагогических учебных заведений. – Москва : Академия, 2003. – 192 с.

22. Карамышева Н. С. Интеграция искусственного интеллекта в образовательные процессы: обучения для анализа и обработки текстов стандартов ФГОС в образовательных системах / Н. С. Карамышева, Э. Я. Юмаева. – Текст : электронный // Электронная библиотека ПГУ. – Пенза : Изд-во ПГУ, 2025. – URL: http://elib.pnzgu.ru/library/1761371533 (дата обращения: 20.05.2026).

23. Карамышева Н. С. Интеграция искусственного интеллекта в образовательные процессы для анализа и обработки текстов стандартов ФГОС в образовательных системах / Н. С. Карамышева, Э. Я. Юмаева. – Текст : электронный // Электронная библиотека ПГУ. – Рязань : ФГБОУ ВО «РГРТУ», 2024. – URL:

http://elib.pnzgu.ru/library/1761285331 (дата обращения: 20.05.2026).

24. Концепция информатизации сферы образования Российской Федерации: Проблемы информатизации высшей школы. – Москва, 1998.

25. Леонов А. Г. Области применения больших языковых моделей для цифровых образовательных платформ / А. Г. Леонов, Н. С. Мартынов, К. А. Мащенко, М. С. Паремузов, К. К. Пчелин, А. В. Шляхов // Труды НИИСИ. – 2025. – Т. 15, № 2. – С. 9–15. – DOI: 10.25682/NIISI.2025.2.0001.

26. Макаров И. М. Автоматизация синтеза и обучение интеллектуальных систем автоматического управления / И. М. Макаров, В. М. Лохин. – Москва, 2009. – 228 с.

27. Мастяев Ф. А. Автоматизация формирования учебных программ и контроля их исполнения в системе высшего профессионального образования [Электронный ресурс] / Ф. А. Мастяев. – 2025. – Режим доступа: https://www.litres.ru/book/f-amastyaev/avtomatizaciya-formirovaniya-uchebnyh-programm-i-kontrolya-ih5616269/ (дата обращения: 05.06.2025).

28. Сафиуллин Р. К. Основы автоматики и автоматизация процессов : учебное пособие для среднего профессионального образования / Р. К. Сафиуллин. – 2-е изд., испр. и доп. – Москва : Юрайт, 2021. – 146 с. – (Профессиональное образование). – ISBN 978-5-534-08256-2. – Режим доступа: https://urait.ru/bcode/473108 (дата обращения: 31.05.2025).

29. Семенов А. Л. Технологии искусственного интеллекта в образовании // Образовательные технологии и общество. – 2022. – Т. 25, № 2. – С. 43–54.

30. Юмаева Э. Я. Автоматизация генерации учебных планов на основе ФГОС и профессиональных стандартов с использованием искусственного интеллекта / Э. Я. Юмаева, Н. С. Карамышева // Информационные технологии в науке и образовании. Проблемы и перспективы : сборник статей XIII

Всероссийской научно-практической конференции (г. Пенза, 11 марта 2026 г.). – Пенза : Изд-во ПГУ, 2026. – С. 1–19.

31. Lo K. M. A closer look into mixture-of-experts in large language models / K. M. Lo, Z. Huang, Z. Qiu, Z. Wang, J. Fu // arXiv preprint. – 2024. – arXiv:2406.18219. – Режим доступа: https://arxiv.org/abs/2406.18219 (дата обращения: 03.04.2026).

32. Xunyu Zhu A Survey on Model Compression for Large Language Models / Zhu Xunyu, Li Jian, Liu Yong, Ma Can, Wang Weiping // Transactions of the Association for Computational Linguistics. – 2024. – Vol. 12. – P. 1556–1577. – DOI: 10.1162/tacl_a_00704.

ПРИЛОЖЕНИЕ

ПРИЛОЖЕНИЕ А

Сгенерированная РПД

УТВЕРЖДАЮ

Декан факультета

_____ /_____/

"____" _____ 2026 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Пензенский государственный университет

Факультет вычислительной техники

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.О.01 Теория вероятностей и математическая статистика

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Направленность подготовки: «Информатика и вычислительная техника»

Квалификация выпускника: бакалавр

Форма обучения: очная

Пенза, 2026

1. Цели освоения дисциплины

Формирование у студентов теоретических знаний и практических навыков в области теории вероятностей и математической статистики, развитие аналитических способностей и умения работать с данными. Подготовка к применению методов теории вероятностей и математической

статистики для решения профессиональных задач в области информатики и вычислительной техники.

2. Место дисциплины в структуре ОПОП

Дисциплина 'Теория вероятностей и математическая статистика' является базовой и входит в обязательную часть образовательной программы по направлению подготовки 09.03.01 'Информатика и вычислительная техника'.

3. Результаты освоения дисциплины «Теория вероятностей и математическая статистика»

Код	Компетенция	Индикатор	Знать	Уметь	Владеть
УК-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	Анализирует и синтезирует информацию, применяет системный подход	Основные понятия и методы теории вероятностей и математической статистики	Применять методы теории вероятностей и математической статистики для решения задач	Навыками анализа и синтеза информации, системного подхода
ОПК-1	Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования в профессиональной деятельности	Применяет естественнонаучные и общеинженерные знания, методы математического анализа и моделирования	Методы математического анализа и моделирования	Применять естественнонаучные и общеинженерные знания для решения профессиональных задач	Навыками применения естественнонаучных и общеинженерных знаний
ПК-10	Способен разрабатывать алгоритмы и программы для решения профессиональных задач	Разрабатывает алгоритмы и программы для решения профессиональных задач	Методы разработки алгоритмов и программ	Разрабатывать алгоритмы и программы для решения профессиональных задач	Навыками разработки алгоритмов и программ

4. Структура и содержание дисциплины «Теория вероятностей и математическая статистика»

Общая трудоемкость дисциплины составляет 4 зачетных единиц, 144 часов.

Раздел	Тема	Семестр	Недели	Лекции	Лаб./практ.	Иное	СРС	Текущий контроль	Промежуточный контроль
Раздел 1. Основы теории вероятностей	Тема 1.1. Основные понятия теории вероятностей	3	1-2	2	2	0	6	Опрос	экзамен
Раздел 2. Случайные величины	Тема 2.1. Дискретные случайные величины	3	3-4	2	2	0	6	Проверка практических работ	экзамен
Раздел 3. Математическая статистика	Тема 3.1. Выборочный метод	3	5-6	2	2	0	6	Опрос	экзамен
Раздел 4. Статистические методы анализа данных	Тема 4.1. Проверка статистических гипотез	3	7-8	2	2	0	6	Проверка практических работ	экзамен

4.2.1. Содержание лекционных занятий

• Тема 1.1. Основные понятия теории вероятностей

• Тема 2.1. Дискретные случайные величины

• Тема 3.1. Выборочный метод

• Тема 4.1. Проверка статистических гипотез

4.2.2. Темы лабораторных работ

Наименование лабораторной / практической работы	Часы
Практическое занятие 1. Решение задач на основные понятия теории вероятностей	2
Практическое занятие 2. Решение задач на дискретные случайные величины	2
Практическое занятие 3. Решение задач на выборочный метод	2
Практическое занятие 4. Решение задач на проверку статистических гипотез	2

5. Образовательные технологии

• Проблемное обучение

6. Учебно-методическое обеспечение самостоятельной работы студентов. Оценочные средства для текущего контроля успеваемости и промежуточной аттестации.

Недели	Тема	Вид работы	Задание	Литература	Часы
1-2	Тема 1.1. Основные понятия теории вероятностей	Подготовка к занятиям	Изучение основных понятий теории вероятностей	1-3	6
3-4	Тема 2.1. Дискретные случайные величины	Анализ практики	Решение задач на дискретные случайные величины	4-5	6
5-6	Тема 3.1. Выборочный метод	Реферат	Подготовка реферата на тему 'Выборочный метод'	1-3	6
7-8	Тема 4.1. Проверка статистических гипотез	Решение кейсов	Решение кейсов на проверку статистических гипотез	4-5	6

6.3. Оценочные средства

• Опрос

• Проверка практических работ

• экзамен

7. Учебно-методическое и материально-техническое обеспечение дисциплины

а) Литература

1. Гмурман В. Е. Теория вероятностей и математическая статистика. – М.: Высшая школа, 2003.

2. Вентцель Е. С. Теория вероятностей. — М.: Высшая школа, 1999.

3. Кремер Н. Ш. Теория вероятностей и математическая статистика. — М.: ЮНИТИ-ДАНА, 2004.

4. Гнеденко Б. В. Курс теории вероятностей. — М.: УРСС, 2005.

5. Чистяков В. П. Курс теории вероятностей. — М.: Наука, 1987.

б) Программное обеспечение

• КонсультантПлюс

в) Материально-техническое обеспечение

• Компьютерный класс

8. Лист согласования

Разработчик рабочей программы	_____	_____
Заведующий кафедрой	_____	_____
Председатель методической комиссии	_____	_____

ПРИЛОЖЕНИЕ Б

Демонстрационная версия системы

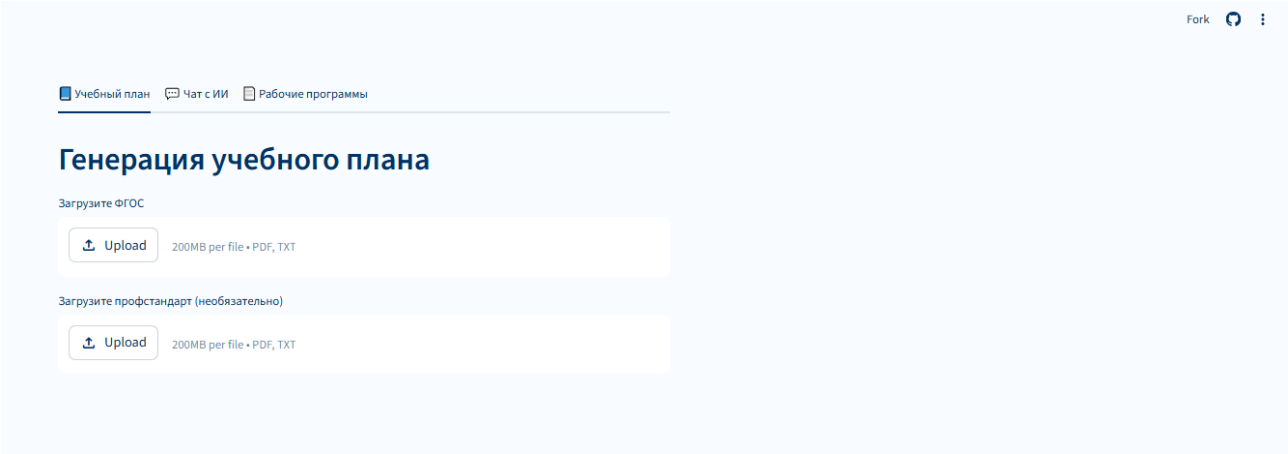


Рис. Б.1. Скриншот демонстрационного сайта проекта; в правом верхнем углу - ссылка на репозиторий

Демонстрационная версия системы доступна онлайн по адресу: https://my-study-plan-app-p2kmlkgrgksagptgnsuaqg.streamlit.app/ (дата обращения: 03.04.2026).

Репозиторий проекта: https://github.com/Elimirage/my-study-plan-app.git

ПРИЛОЖЕНИЕ В

Листинг

Листинг В.1 — Интеграция с YandexGPT

```
def post_to_yandex(messages, model_name: str, temperature: float,
max_tokens: int) -> dict:
    data = {
        "modelUri": f"gpt://{FOLDER_ID}/{model_name}",
        "completionOptions": {
            "stream": False,
            "temperature": temperature,
            "maxTokens": str(max_tokens)
        },
        "messages": messages
    }

    response = requests.post(
        YANDEX_COMPLETION_URL,
        headers=get_yandex_headers(),
        json=data,
        timeout=60
    )

    if not response.ok:
        raise RuntimeError(
            f"Yandex API вернул {response.status_code}:
{response.text}"
        )

    return response.json()


tesseract_path = os.getenv("TESSERACT_CMD")
if tesseract_path:
    pytesseract.pytesseract.tesseract_cmd = tesseract_path


def call_yandex_lite(messages, temperature=0.3, max_tokens=1500):
    result = post_to_yandex(
        messages=messages,
        model_name=YANDEX_MODEL_LITE,
        temperature=temperature,
        max_tokens=max_tokens
    )

    return result["result"]["alternatives"][0]["message"]["text"]


def completion_with_ai(prompt: str) -> str:
    system_prompt = """
Ты — система редактирования учебного плана.

ТВОЯ ЗАДАЧА:
Преобразовать команду пользователя в один JSON-объект.
```

```
ТЫ НЕ ДОЛЖЕН:
- писать пояснения;
- писать текст до JSON;
- писать текст после JSON;
- добавлять новую дисциплину, если пользователь просит изменить
существующую;
- менять профиль подготовки;
- придумывать педагогические дисциплины;
- возвращать Markdown.

ВАЖНОЕ ПРАВИЛО:
Если пользователь пишет "добавь экзамен по дисциплине X",
это значит нужно ОБНОВИТЬ поле "Форма контроля" у существующей
дисциплины X,
а НЕ добавлять новую дисциплину.

Если пользователь пишет "увеличь часы по дисциплине X",
нужно обновить поле "Часы", но не уменьшать значение случайно.
Если точное значение не указано, верни error.

ФОРМАТЫ JSON:

UPDATE:
{
  "action": "update",
  "discipline": "Название дисциплины",
  "field": "Часы",
  "value": 144
}

UPDATE ФОРМЫ КОНТРОЛЯ:
{
  "action": "update",
  "discipline": "Операционные системы",
  "field": "Форма контроля",
  "value": "экзамен"
}

MOVE:
{
  "action": "update",
  "discipline": "Архитектура вычислительных систем",
  "field": "Семестр",
  "value": 4
}

DELETE:
{
  "action": "delete",
  "discipline": "Название дисциплины"
}

ADD:
{
  "action": "add",
  "field": "row",
  "value": {
    "Блок": "Блок 1. Вариативная часть",
```

```
    "Семестр": 1,
    "Дисциплина": "Название дисциплины",
    "Часы": 72,
    "Форма контроля": "зачёт",
    "Компетенции ФГОС": [],
    "Трудовые функции": "",
    "Обоснование": "Добавлено через ИИ-консультанта"
  }
}

ERROR:
{
  "action": "error",
  "value": "Причина ошибки"
}

Верни только JSON.
"""

    messages = [
        {"role": "system", "text": system_prompt},
        {"role": "user", "text": prompt}
    ]

    try:
        result = post_to_yandex(
            messages=messages,
            model_name=YANDEX_MODEL_CHAT,
            temperature=0.02,
            max_tokens=500
        )

        text = result["result"]["alternatives"][0]["message"]["text"]
        parsed = extract_json(text)

        return json.dumps(parsed, ensure_ascii=False)

    except Exception as e:
        return json.dumps({
            "action": "error",
            "value": f"Ошибка обращения к модели: {e}"
        }, ensure_ascii=False)
```

Листинг В.5 — JSON-валидация и редактирование через ИИ
app.py

```
def apply_edit_command(df: pd.DataFrame, command: dict) ->
tuple[pd.DataFrame, str]:
    action = command.get("action")

    if action == "update":
        disc = command.get("discipline")
        field = command.get("field")
        value = command.get("value")

        if disc is None or field is None:
```

```
            return df, "Команда некорректна: нет discipline или
field."

        if field not in df.columns:
            return df, f"Поле '{field}' не найдено в таблице."

        mask = df["Дисциплина"] == disc
        if not mask.any():
            return df, f"Дисциплина '{disc}' не найдена."

        df.loc[mask, field] = value
        return df, f"Обновлено поле '{field}' у дисциплины '{disc}' →
{value}."

    elif action == "delete":
        disc = command.get("discipline")
        if disc is None:
            return df, "Команда некорректна: нет discipline."

        before = len(df)
        df = df[df["Дисциплина"] != disc].reset_index(drop=True)
        after = len(df)

        if before == after:
            return df, f"Дисциплина '{disc}' не найдена."
        else:
            return df, f"Дисциплина '{disc}' удалена."

    elif action == "add":
        field = command.get("field")
        value = command.get("value")

        if field != "row" or not isinstance(value, dict):
            return df, "Команда add некорректна: ожидается field='row'
и объект value."

        new_row = {}
        for col in df.columns:
            new_row[col] = value.get(col, None)

        df = pd.concat([df, pd.DataFrame([new_row])],
ignore_index=True)
        return df, f"Добавлена новая дисциплина
'{value.get('Дисциплина', 'Без названия')}'."

    elif action == "error":
        return df, f"Ошибка на стороне ИИ: {command.get('value')}"

    else:
        return df, f"Неизвестное действие: {action}"


tab_plan, tab_chat, tab_rpd = st.tabs([
    "□ Учебный план",
    "□ Чат с ИИ",
    "□ Рабочие программы"
])
```

```
with tab_plan:
    st.header("Генерация учебного плана")

    uploaded_fgos = st.file_uploader("Загрузите ФГОС", type=["pdf",
"txt"], key="fgos_uploader")
    uploaded_tf = st.file_uploader(
        "Загрузите профстандарт (необязательно)",
        type=["pdf", "txt"],
        key="prof_uploader"
    )

    # Инициализация session_state
    if "df_fgos" not in st.session_state:
        st.session_state.df_fgos = pd.DataFrame()

    if "fgos_text" not in st.session_state:
        st.session_state.fgos_text = ""

    if "tf_struct" not in st.session_state:
        st.session_state.tf_struct = {"TF": []}

    if "prof_text" not in st.session_state:
        st.session_state.prof_text = ""

    if "detected_profiles" not in st.session_state:
        st.session_state.detected_profiles = []

    if uploaded_fgos:
        with st.spinner("Обработка ФГОС..."):
            try:
                if uploaded_fgos.name.endswith(".pdf"):
                    fgos_text =
extract_text_from_pdf_file(uploaded_fgos)
                    if fgos_text.startswith("OCR error") or
len(fgos_text.strip()) < 50:
                        st.warning("□ Возможны проблемы с извлечением
текста из PDF.")
                else:
                    fgos_text = uploaded_fgos.read().decode("utf-8",
errors="ignore")
                    uploaded_fgos.seek(0)

                if not fgos_text or len(fgos_text.strip()) < 50:
st.error("✕ Не удалось извлечь текст из файла ФГОС.")
                    st.session_state.df_fgos = pd.DataFrame()
                    st.session_state.fgos_text = ""
                else:
                    st.session_state.fgos_text = fgos_text
                    competencies =
extract_competencies_full(fgos_text)

                    if competencies:
                        df_fgos = pd.DataFrame(competencies)
                        st.session_state.df_fgos = df_fgos
```

st.success(f"✓ ФГОС обработан. Извлечено компетенций:
{len(competencies)}")
```
                        st.dataframe(df_fgos,
use_container_width=True, height=300)

                        try:
                            profiles =
detect_profile_from_fgos(fgos_text)
                            if profiles:
                                st.session_state.detected_profiles =
profiles
                                st.info(f" Определенный профиль: {',
'.join(profiles)}")
                        except Exception as e:
                            st.warning(f" Не удалось определить
профиль: {e}")
                    else:
                        st.warning(" Компетенции не найдены.")
                        st.session_state.df_fgos = pd.DataFrame()

            except Exception as e:
                st.session_state.df_fgos = pd.DataFrame()
                st.session_state.fgos_text = ""
                st.error(f"Ошибка при обработке ФГОС: {e}")

    if uploaded_tf:
        with st.spinner("Обработка профстандарта..."):
            try:
                if uploaded_tf.name.endswith(".pdf"):
                    prof_text =
extract_text_from_pdf_file(uploaded_tf)
                else:
                    prof_text = uploaded_tf.read().decode("utf-8",
errors="ignore")
                    uploaded_tf.seek(0)

                if not prof_text or len(prof_text.strip()) < 50:
                    st.session_state.tf_struct = {"TF": []}
                    st.session_state.prof_text = ""
                    st.warning(" Не удалось извлечь текст из
профстандарта.")
                else:
                    st.session_state.prof_text = prof_text
                    tf_struct, error =
analyze_prof_standard(prof_text)

                    if error:
                        st.session_state.tf_struct = {"TF": []}
                        st.warning(f" Ошибка анализа профстандарта:
{error}")

                    elif tf_struct:
                        st.session_state.tf_struct = tf_struct
                        tf_list = tf_struct.get("TF", [])

                        if tf_list:
```

```
                                tf_display = []
                                for tf in tf_list:
                                    tf_display.append({
                                        "Код": tf.get("code", ""),
                                        "Название": (tf.get("name") or
"")[:100],
                                        "Действия": len(tf.get("actions")
or []),
                                        "Знания": len(tf.get("knowledge")
or []),
                                        "Умения": len(tf.get("skills") or
[])
                                    })

                                df_tf = pd.DataFrame(tf_display)
                    st.success(f"✓ Профстандарт обработан. Найдено ТФ: {len(tf_list)}")
                                st.dataframe(df_tf,
use_container_width=True, height=300)
                            else:
                                st.session_state.tf_struct = {"TF": []}
                                st.warning("□ Трудовые функции не найдены
в профстандарте.")
                        else:
                            st.session_state.tf_struct = {"TF": []}
                            st.warning("□ Не удалось обработать
профстандарт.")

                except Exception as e:
                    st.session_state.tf_struct = {"TF": []}
                    st.session_state.prof_text = ""
                    st.warning(f"Профстандарт не обработан: {e}")

    # Генерация плана теперь возможна и без профстандарта
    fgos_ready = not st.session_state.get("df_fgos",
pd.DataFrame()).empty
    tf_struct = st.session_state.get("tf_struct", {"TF": []})

    if fgos_ready:
        if len(tf_struct.get("TF", [])) > 0:
            st.info(f"Профстандарт загружен: найдено ТФ
{len(tf_struct.get('TF', []))}")
        else:
            st.info("Профстандарт не загружен или не распознан. План
будет сгенерирован без него.")

        if st.button("□ Сгенерировать учебный план", type="primary",
use_container_width=True):
            try:
                df = generate_plan_pipeline(
                    st.session_state.df_fgos,
                    tf_struct,
                    {},
                    st.session_state.get("fgos_text", "")
                )
                st.session_state.df = df
        st.success("✓ Учебный план успешно сгенерирован!")
```

```
                st.balloons()
            except Exception as e:
                st.error(f"Ошибка генерации учебного плана: {e}")

    if "df" in st.session_state:
        st.subheader("📋 Сформированный учебный план")
        st.dataframe(st.session_state.df, use_container_width=True)

        buffer = io.BytesIO()
        st.session_state.df.to_excel(buffer, index=False,
engine="openpyxl")
        buffer.seek(0)

        st.download_button(
            "📥 Скачать учебный план (Excel)",
            buffer,
            "учебный_план.xlsx",
            "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet",
            use_container_width=True
        )


with tab_chat:
    st.header("💬 Чат с ИИ-методистом")

    if "df" not in st.session_state:
        st.info("ℹ️ Сначала сгенерируйте учебный план.")
    else:
        df = st.session_state.df

        if "chat_mode" not in st.session_state:
            st.session_state.chat_mode = "consultation"

        col1, col2 = st.columns(2)

        if col1.button("💬 Консультация", use_container_width=True):
            st.session_state.chat_mode = "consultation"
            st.session_state.consultation_messages = []
            st.rerun()

        if col2.button("✏️ Редактирование", use_container_width=True):
            st.session_state.chat_mode = "editing"
            st.session_state.edit_messages = []
            st.rerun()

        if st.session_state.chat_mode == "consultation":
            from ai import consult_with_methodologist

            if "consultation_messages" not in st.session_state:
                st.session_state.consultation_messages = []

            plan_context = {
                "disciplines_count": len(df),
                "profile": st.session_state.get("detected_profiles",
["не определен"])[0]
```

```
                if st.session_state.get("detected_profiles") else "не
определен",
                "competencies_count":
len(st.session_state.get("df_fgos", pd.DataFrame())),
                "tf_count": len(st.session_state.get("tf_struct",
{}).get("TF", [])),
                "disciplines_list": df["Дисциплина"].tolist() if
"Дисциплина" in df.columns else [],
                "chat_history": st.session_state.consultation_messages
            }

            for msg in st.session_state.consultation_messages:
                st.chat_message(msg["role"]).write(msg["content"])

            prompt = st.chat_input("Вопрос методисту")

            if prompt:
                st.session_state.consultation_messages.append({"role":
"user", "content": prompt})
                try:
                    response = consult_with_methodologist(prompt,
plan_context)
                except Exception as e:
                    response = f"Ошибка при обращении к методисту:
{str(e)}"
                st.session_state.consultation_messages.append({"role":
"assistant", "content": response})
                st.rerun()

        else:
            if "edit_messages" not in st.session_state:
                st.session_state.edit_messages = []

            for msg in st.session_state.edit_messages:
                st.chat_message(msg["role"]).write(msg["content"])

            prompt = st.chat_input("Опишите изменение")

            if prompt:
                st.session_state.edit_messages.append({"role": "user",
"content": prompt})
                raw = completion_with_ai(prompt)

                try:
                    command = json.loads(raw)
                    df, text = apply_edit_command(df, command)
                    st.session_state.df = df
                except Exception:
                    text = raw

                st.session_state.edit_messages.append({"role":
"assistant", "content": text})
                st.rerun()

        st.subheader("□ Обновлённый учебный план")
        st.dataframe(st.session_state.df, use_container_width=True)
with tab_rpd:
    st.header("□ Рабочие программы дисциплин")
```

```
    if "df" not in st.session_state or st.session_state.df.empty:
        st.info("Сначала сгенерируйте учебный план.")
    else:
        from work_program import generate_work_program_content,
create_work_program_docx

        df = st.session_state.df.copy()

        if "Дисциплина" not in df.columns:
            st.error("В учебном плане нет колонки 'Дисциплина'.")
        else:
            disciplines =
df["Дисциплина"].dropna().astype(str).tolist()

            selected_discipline = st.selectbox(
                "Выберите дисциплину",
                disciplines,
                key="selected_rpd_discipline"
            )

            default_direction_code = "09.03.01"

            detected_profiles =
st.session_state.get("detected_profiles", [])
            detected_profile = detected_profiles[0] if
detected_profiles else ""

            direction_code = select_or_custom(
                "Код направления",
                list(DIRECTION_MAP.keys()),
                default=default_direction_code,
                key="direction_code"
            )

            direction_info = DIRECTION_MAP.get(direction_code, {
                "name": "",
                "qualification": "",
                "profiles": [],
                "faculties": [],
            })

            auto_direction_name = direction_info.get("name", "")
            auto_qualification = direction_info.get("qualification",
"")
            auto_profiles = direction_info.get("profiles", [])
            auto_faculties = direction_info.get("faculties", [])

            st.text_input(
                "Направление подготовки",
                value=auto_direction_name,
                disabled=True,
                key="direction_name_locked"
            )

            st.text_input(
                "Квалификация",
                value=auto_qualification,
```

```
                disabled=True,
                key="qualification_locked"
            )

            profile_default = detected_profile if detected_profile
else (auto_profiles[0] if auto_profiles else "")

            profile = select_or_custom(
                "Профиль подготовки",
                auto_profiles,
                default=profile_default,
                key="profile"
            )

            education_form = select_or_custom(
                "Форма обучения",
                ["очная", "очно-заочная", "заочная"],
                default="очная",
                key="education_form"
            )

            university_name = select_or_custom(
                "Университет",
                UNIVERSITY_OPTIONS,
                default="Пензенский государственный университет",
                key="university_name"
            )

            faculty_default = auto_faculties[0] if auto_faculties else
"Ваш факультет"

            faculty_name = select_or_custom(
                "Факультет",
                auto_faculties + [
                    "Факультет вычислительной техники",
                    "Факультет информационных технологий",
                    "Юридический факультет",
                    "Педагогический факультет",
                    "Экономический факультет",
                    "Ваш факультет",
                ],
                default=faculty_default,
                key="faculty_name"
            )

            row = df[df["Дисциплина"].astype(str) ==
selected_discipline].iloc[0].to_dict()

            with st.expander("Параметры для генерации",
expanded=False):
                st.write({
                    "discipline": selected_discipline,
                    "direction_code": direction_code,
                    "direction_name": auto_direction_name,
                    "qualification": auto_qualification,
                    "profile": profile,
                    "education_form": education_form,
                    "university_name": university_name,
```

```
                    "faculty_name": faculty_name,
                })

            if st.button("Сгенерировать рабочую программу DOCX",
type="primary", use_container_width=True):
                try:
                    with st.spinner("Генерация рабочей программы..."):
                        content = generate_work_program_content(
                            discipline_row=row,
                            profile=profile,
                            direction_code=direction_code,
                            direction_name=auto_direction_name,
                            qualification=auto_qualification,
                            education_form=education_form,
                            university_name=university_name,
                            faculty_name=faculty_name,
                        )

                        docx_bytes = create_work_program_docx(content)

                    st.success("Рабочая программа сформирована.")

                    filename =
f"Рабочая_программа_{selected_discipline.replace(' ', '_')}.docx"

                    st.download_button(
                        label="  Скачать DOCX",
                        data=docx_bytes,
                        file_name=filename,
                        mime="application/vnd.openxmlformats-
officedocument.wordprocessingml.document",
                        use_container_width=True,
                    )

                    with st.expander("Предпросмотр структуры",
expanded=False):
                        st.json(content)

                except Exception as e:
                    st.error(f"Ошибка генерации рабочей программы:
{e}")
```

Листинг В.3 — Генерация дисциплин
disciplines.py

```
def generate_disciplines(profile, df_fgos=None, tf_struct=None,
fgos_text="", min_fund=12, min_var=15):
    """
    Генерация фундаментальных и вариативных дисциплин под профиль.

    Главное исправление:
    модель получает жёсткое указание НЕ использовать педагогические
дисциплины
    для технических, физико-математических и художественных
направлений.
    """
    profile = profile or "Не указан"
```

```
    profile_type = detect_profile_type(profile)
    rules = PROFILE_RULES.get(profile_type, PROFILE_RULES["generic"])

    competencies = _safe_get_competencies(df_fgos)
    tf_info = _safe_get_tf(tf_struct)

    allowed_list = "\n".join([f"- {x}" for x in rules["allowed"]]) or
"- подбирать строго по профилю"
    forbidden_list = "\n".join([f"- {x}" for x in rules["forbidden"]])
or "- нет"

    fgos_short = ""
    if fgos_text:
        fgos_short = str(fgos_text)[:2500]

    prompt = f"""
Ты — методист российского вуза.

Сгенерируй учебные дисциплины строго для указанного направления.

НАПРАВЛЕНИЕ / ПРОФИЛЬ:
{profile}

ТИП НАПРАВЛЕНИЯ:
{rules["description"]}

КОМПЕТЕНЦИИ ФГОС:
{competencies}

ТРУДОВЫЕ ФУНКЦИИ ПРОФСТАНДАРТА:
{tf_info}

ФРАГМЕНТ ТЕКСТА ФГОС:
{fgos_short}

РАЗРЕШЁННЫЕ ТЕМАТИЧЕСКИЕ ОБЛАСТИ:
{allowed_list}

ЗАПРЕЩЁННЫЕ ТЕМЫ И СЛОВА:
{forbidden_list}

СТРОГИЕ ПРАВИЛА:
1. Генерируй дисциплины только для профиля: {profile}.
2. Если направление НЕ педагогическое, запрещено добавлять
педагогические дисциплины.
3. Не используй слова из списка запрещённых тем.
4. Не переноси дисциплины из предыдущих генераций.
5. Не используй общий шаблон для всех направлений.
6. Дисциплины должны быть похожи на реальные вузовские дисциплины.
7. Фундаментальные дисциплины — базовые для направления.
8. Вариативные дисциплины — профильные и прикладные для направления.
9. Верни только JSON без пояснений, Markdown и текста вокруг.

Сгенерируй:
- минимум {min_fund} фундаментальных дисциплин;
- минимум {min_var} вариативных дисциплин.

Формат ответа:
```

```
{{
  "fundamental": [
    {{"name": "Название дисциплины"}}
  ],
  "variative": [
    {{"name": "Название дисциплины"}}
  ]
}}
"""

    raw = call_yandex_lite(
        [{"role": "user", "text": prompt}],
        temperature=0.03,
        max_tokens=3000
    )

    try:
        start = raw.index("{")
        end = raw.rindex("}") + 1
        data = json.loads(raw[start:end])
    except Exception:
        data = _fallback_disciplines(profile_type)

    fund = data.get("fundamental", [])
    var = data.get("variative", [])

    filtered_fund = _filter_disciplines(fund, rules["forbidden"])
    filtered_var = _filter_disciplines(var, rules["forbidden"])

    # Если после фильтрации модель вернула слишком мало дисциплин,
    # используем безопасный профильный fallback.
    if len(filtered_fund) < 5 or len(filtered_var) < 5:
        safe = _fallback_disciplines(profile_type)
        filtered_fund = _filter_disciplines(safe["fundamental"],
rules["forbidden"])
        filtered_var = _filter_disciplines(safe["variative"],
rules["forbidden"])

    for d in filtered_fund:
        d["block_hint"] = "обязательная"

    for d in filtered_var:
        d["block_hint"] = "вариативная"

    return filtered_fund + filtered_var
```

Листинг В.2 — Извлечение компетенций из ФГОС

fgos.py

```
def extract_competencies_full(text):
    """
    Извлекает УК, ОПК, ПК из текста ФГОС.
    """
    pattern = (
        r"(УК-\d+|ОПК-\d+|ПК-\d+)\.?\s*(.*?)\s*"
        r"(?=(УК-\d+|ОПК-\d+|ПК-\d+|IV\. Требования к условиям
реализации программы бакалавриата|$))"
```

```
    )
    matches = re.findall(pattern, text, re.DOTALL)

    competencies = []
    for code, desc, _ in matches:
        desc = " ".join(desc.split())
        competencies.append({"code": code, "description": desc})
    return competencies
```

Листинг В.4 — Формирование учебного плана

plan.py

```
def generate_plan_pipeline(df_fgos, tf_struct, match_json, fgos_text):
    if df_fgos is None or (isinstance(df_fgos, pd.DataFrame) and
df_fgos.empty):
        df_fgos = pd.DataFrame(columns=["code", "description"])

    if tf_struct is None:
        tf_struct = {"TF": []}

    profile = detect_profile_by_code(fgos_text)

    if profile is None:
        profile = detect_profile_advanced(fgos_text, [])

    if profile is None:
        profile = "ИВТ"

    level = detect_level(fgos_text, profile)

    discs = generate_disciplines(profile, df_fgos, tf_struct,
fgos_text)

    if not discs:
        raise ValueError("Не удалось сгенерировать дисциплины")

    discs = remove_duplicates(discs)

    enriched = {}

    for d in discs:
        enriched[d["name"]] = enrich_discipline_metadata(
            d,
            df_fgos,
            tf_struct,
            profile,
            fgos_text
        )

    obligatory = [
        d for d in discs
        if d.get("block_hint") == "обязательная"
    ]

    variative = [
        d for d in discs
        if d.get("block_hint") == "вариативная"
```

```
    ]

    semester_map = balanced_distribution(obligatory, variative, level)

    rows = []

    for sem, disc_list in semester_map.items():
        for disc in disc_list:
            name = disc["name"]
            meta = enriched.get(name, {})

            competencies = detect_competencies(profile, name)

            if not competencies and meta.get("competencies"):
                competencies = meta.get("competencies", [])

            if not competencies and not df_fgos.empty:
                competencies = find_competencies_by_discipline(name,
df_fgos)

            if not isinstance(competencies, list):
                competencies = []

            tf_codes = meta.get("TF", [])

            if not isinstance(tf_codes, list):
                tf_codes = []

            block_name = (
                "Блок 1. Обязательная часть"
                if disc.get("block_hint") == "обязательная"
                else "Блок 1. Вариативная часть"
            )

            hours = 144 if disc.get("block_hint") == "обязательная"
else 108

            rows.append({
                "Блок": block_name,
                "Семестр": sem,
                "Дисциплина": name,
                "Часы": hours,
                "Форма контроля": assign_assessment(name),
                "Компетенции ФГОС": competencies,
                "Трудовые функции": ", ".join(tf_codes) if tf_codes
else "",
                "Обоснование": meta.get("reason", "") or
generate_reason(name, competencies, tf_codes)
            })

    rows = _add_practice_and_gia(rows, profile, level)

    return pd.DataFrame(rows)
```

Листинг В.6 — Генерация РПД в DOCX
work_program.py

```
def generate_work_program_content(
    discipline_row: Dict[str, Any],
    profile: str,
    direction_code: str,
    direction_name: str,
    qualification: str,
    education_form: str,
    university_name: str,
    faculty_name: str,
) -> Dict[str, Any]:
    discipline_name = _safe_str(discipline_row.get("Дисциплина"))
    semester = _safe_int(discipline_row.get("Семестр"), 1)
    hours = _safe_int(discipline_row.get("Часы"), 108)
    control_form = _safe_str(discipline_row.get("Форма контроля")) or
"зачёт"

    competencies = _extract_competencies_only(discipline_row)
    tf_list = _extract_tf_only(discipline_row)
    reason = _safe_str(discipline_row.get("Обоснование"))

    credits = max(1, round(hours / 36)) if hours else 3

    competencies_text = ", ".join(competencies) if competencies else
"не указаны"
    tf_text = ", ".join(tf_list) if tf_list else "не указаны"

    prompt = f"""
Ты — опытный методист российского вуза.

Сгенерируй рабочую программу дисциплины в формате СТРОГОГО JSON.
Без markdown.
Без пояснений.
Без текста до JSON и после JSON.

Входные данные:
- Дисциплина: {discipline_name}
- Профиль: {profile}
- Код направления: {direction_code}
- Направление подготовки: {direction_name}
- Квалификация: {qualification}
- Форма обучения: {education_form}
- Семестр: {semester}
- Объем часов: {hours}
- Зачетные единицы: {credits}
- Форма контроля: {control_form}
- Компетенции ФГОС: {competencies_text}
- Трудовые функции профстандарта: {tf_text}
- Обоснование дисциплины: {reason if reason else "не указано"}

СТРОГИЕ ПРАВИЛА:
1. Используй только компетенции ФГОС (УК-..., ОПК-..., ПК-...) в
таблице результатов.
2. Не включай трудовые функции профстандарта в таблицу результатов.
3. Трудовые функции можно упоминать только в описательном тексте, если
это уместно.
4. Содержание должно соответствовать дисциплине "{discipline_name}".
5. Сделай документ реалистичным для вуза РФ.
6. Минимум:
```

   - 8 тем лекций,
   - 4 лабораторных или практических работ,
   - 4 строки самостоятельной работы,
   - 5 источников литературы.
7. Таблица структуры должна содержать не менее 6 строк.
8. Самостоятельная работа должна быть разнообразной:
   - подготовка к занятиям,
   - анализ практики,
   - реферат,
   - решение кейсов.
9. Не используй "..." и пустые заглушки.
10. Программное обеспечение и оборудование должны быть конкретными.
11. Используй минимум 1 УК, 1 ОПК и 1 ПК.
12. Если дисциплина гуманитарная, не вставляй ИТ-содержание.
13. Если дисциплина техническая, не вставляй юридическое содержание.

Верни JSON такой структуры:
```
{{
  "title": "Рабочая программа дисциплины",
  "discipline_code": "Б1.О.01",
  "discipline_name": "{discipline_name}",
  "goals": "1-2 абзаца",
  "place_in_program": "1-2 абзаца",
  "results": [
    {{
      "code": "УК-1",
      "competence": "Формулировка компетенции",
      "indicator": "Индикатор достижения",
      "know": "Что знает",
      "able": "Что умеет",
      "master": "Чем владеет"
    }}
  ],
  "total_credits": {credits},
  "total_hours": {hours},
  "structure_rows": [
    {{
      "section": "Раздел 1. ...",
      "topic": "Тема 1.1. ...",
      "semester": {semester},
      "weeks": "1-2",
      "lectures": 2,
      "labs": 2,
      "other_contact": 0,
      "self_study": 6,
      "current_control": "Опрос",
      "intermediate_control": "{control_form}"
    }}
  ],
  "lecture_topics": [
    "Тема 1.1. ..."
  ],
  "lab_topics": [
    {{
      "name": "Практическое занятие 1. ...",
      "hours": 2
    }}
  ],
```

```
  "education_technologies": [
    "Проблемное обучение"
  ],
  "self_study_rows": [
    {{
      "weeks": "1-2",
      "topic": "Тема 1.1. ...",
      "kind": "Подготовка к занятиям",
      "task": "Конкретное задание",
      "literature": "1-3",
      "hours": 6
    }}
  ],
  "assessment_tools": [
    "Опрос",
    "Проверка практических работ",
    "{control_form}"
  ],
  "literature": [
    "1. ..."
  ],
  "software": [
    "КонсультантПлюс"
  ],
  "equipment": [
    "Компьютерный класс"
  ]
}}
"""

    raw = call_yandex_lite(
        [{"role": "user", "text": prompt}],
        temperature=0.2,
        max_tokens=3500,
    )

    data = _safe_json_from_text(raw)

    data["meta"] = {
        "university_name": university_name,
        "faculty_name": faculty_name,
        "direction_code": direction_code,
        "direction_name": direction_name,
        "profile": profile,
        "qualification": qualification,
        "education_form": education_form,
        "input_competencies": competencies,
        "input_tf": tf_list,
    }

    return data


def create_work_program_docx(data: Dict[str, Any]) -> bytes:
    doc = Document()
    _set_base_style(doc)

    meta = data.get("meta", {})
```

```
    university_name = _safe_str(meta.get("university_name"))
    faculty_name = _safe_str(meta.get("faculty_name"))
    direction_code = _safe_str(meta.get("direction_code"))
    direction_name = _safe_str(meta.get("direction_name"))
    profile = _safe_str(meta.get("profile"))
    qualification = _safe_str(meta.get("qualification"))
    education_form = _safe_str(meta.get("education_form"))

    discipline_code = _safe_str(data.get("discipline_code")) or
"Б1.О.01"
    discipline_name = _safe_str(data.get("discipline_name"))
    total_credits = _safe_int(data.get("total_credits"), 3)
    total_hours = _safe_int(data.get("total_hours"), 108)

    current_year = datetime.now().year
    city_name = "Пенза"

    # Блок "Утверждаю"
    p = doc.add_paragraph()
    p.alignment = WD_ALIGN_PARAGRAPH.RIGHT
    run = p.add_run("УТВЕРЖДАЮ")
    run.bold = True
    run.font.name = "Times New Roman"
    run.font.size = Pt(12)

    p = doc.add_paragraph()
    p.alignment = WD_ALIGN_PARAGRAPH.RIGHT
    run = p.add_run("Декан факультета")
    run.font.name = "Times New Roman"
    run.font.size = Pt(12)

    p = doc.add_paragraph()
    p.alignment = WD_ALIGN_PARAGRAPH.RIGHT
    run = p.add_run("_____ /_____/")
    run.font.name = "Times New Roman"
    run.font.size = Pt(12)

    p = doc.add_paragraph()
    p.alignment = WD_ALIGN_PARAGRAPH.RIGHT
    run = p.add_run(f'"____" _____ {current_year} г.')
    run.font.name = "Times New Roman"
    run.font.size = Pt(12)

    _add_paragraph(doc, "", align=WD_ALIGN_PARAGRAPH.CENTER)

    # Титульный лист
    _add_paragraph(doc, "МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ",
bold=True, align=WD_ALIGN_PARAGRAPH.CENTER)
    _add_paragraph(doc, "РОССИЙСКОЙ ФЕДЕРАЦИИ", bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)
    _add_paragraph(doc, "", align=WD_ALIGN_PARAGRAPH.CENTER)
    _add_paragraph(doc, university_name, bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)
    _add_paragraph(doc, faculty_name, bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)
    _add_paragraph(doc, "", align=WD_ALIGN_PARAGRAPH.CENTER)
    _add_paragraph(doc, "РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ", bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER, size=14)
```

```
    _add_paragraph(doc, "", align=WD_ALIGN_PARAGRAPH.CENTER)
    _add_paragraph(doc, f"{discipline_code} {discipline_name}",
bold=True, align=WD_ALIGN_PARAGRAPH.CENTER, size=13)
    _add_paragraph(doc, "", align=WD_ALIGN_PARAGRAPH.CENTER)

    _add_paragraph(doc, f"Направление подготовки: {direction_code}
«{direction_name}»", align=WD_ALIGN_PARAGRAPH.LEFT)
    _add_paragraph(doc, f"Направленность подготовки: «{profile}»",
align=WD_ALIGN_PARAGRAPH.LEFT)
    _add_paragraph(doc, f"Квалификация выпускника: {qualification}",
align=WD_ALIGN_PARAGRAPH.LEFT)
    _add_paragraph(doc, f"Форма обучения: {education_form}",
align=WD_ALIGN_PARAGRAPH.LEFT)

    _add_paragraph(doc, "", align=WD_ALIGN_PARAGRAPH.CENTER)
    _add_paragraph(doc, f"{city_name}, {current_year}",
align=WD_ALIGN_PARAGRAPH.CENTER)

    doc.add_page_break()

    # 1
    _add_heading(doc, "1. Цели освоения дисциплины")
    _add_paragraph(doc, _safe_str(data.get("goals")))

    # 2
    _add_heading(doc, "2. Место дисциплины в структуре ОПОП")
    _add_paragraph(doc, _safe_str(data.get("place_in_program")))

    # 3
    _add_heading(doc, f"3. Результаты освоения дисциплины
«{discipline_name}»")

    results = data.get("results", [])
    if isinstance(results, list) and results:
        table = doc.add_table(rows=1, cols=6)
        table.style = "Table Grid"
        table.alignment = WD_TABLE_ALIGNMENT.CENTER

        hdr = table.rows[0].cells
        _set_cell_text(hdr[0], "Код", bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)
        _set_cell_text(hdr[1], "Компетенция", bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)
        _set_cell_text(hdr[2], "Индикатор", bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)
        _set_cell_text(hdr[3], "Знать", bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)
        _set_cell_text(hdr[4], "Уметь", bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)
        _set_cell_text(hdr[5], "Владеть", bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)

        for item in results:
            code = _safe_str(item.get("code")).upper().replace(" ",
"")
            if not (code.startswith("УК-") or code.startswith("ОПК-")
or code.startswith("ПК-")):
                continue
```

```
            row = table.add_row().cells
            _set_cell_text(row[0], item.get("code", ""))
            _set_cell_text(row[1], item.get("competence", ""))
            _set_cell_text(row[2], item.get("indicator", ""))
            _set_cell_text(row[3], item.get("know", ""))
            _set_cell_text(row[4], item.get("able", ""))
            _set_cell_text(row[5], item.get("master", ""))

    # 4
    _add_heading(doc, f"4. Структура и содержание дисциплины
«{discipline_name}»")
    _add_paragraph(
        doc,
        f"Общая трудоемкость дисциплины составляет {total_credits}
зачетных единиц, {total_hours} часов."
    )

    structure_rows = data.get("structure_rows", [])
    if isinstance(structure_rows, list) and structure_rows:
        table = doc.add_table(rows=1, cols=10)
        table.style = "Table Grid"
        table.alignment = WD_TABLE_ALIGNMENT.CENTER

        hdr = table.rows[0].cells
        headers = [
            "Раздел",
            "Тема",
            "Семестр",
            "Недели",
            "Лекции",
            "Лаб./практ.",
            "Иное",
            "СРС",
            "Текущий контроль",
            "Промежуточный контроль",
        ]
        for i, h in enumerate(headers):
            _set_cell_text(hdr[i], h, bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)

        for item in structure_rows:
            row = table.add_row().cells
            _set_cell_text(row[0], item.get("section", ""))
            _set_cell_text(row[1], item.get("topic", ""))
            _set_cell_text(row[2], item.get("semester", ""))
            _set_cell_text(row[3], item.get("weeks", ""))
            _set_cell_text(row[4], item.get("lectures", ""))
            _set_cell_text(row[5], item.get("labs", ""))
            _set_cell_text(row[6], item.get("other_contact", ""))
            _set_cell_text(row[7], item.get("self_study", ""))
            _set_cell_text(row[8], item.get("current_control", ""))
            _set_cell_text(row[9], item.get("intermediate_control",
""))

    _add_heading(doc, "4.2.1. Содержание лекционных занятий")
    for topic in data.get("lecture_topics", []):
        _add_paragraph(doc, f"• {_safe_str(topic)}")
```

```
    _add_heading(doc, "4.2.2. Темы лабораторных работ")
    lab_topics = data.get("lab_topics", [])
    if isinstance(lab_topics, list) and lab_topics:
        table = doc.add_table(rows=1, cols=2)
        table.style = "Table Grid"
        table.alignment = WD_TABLE_ALIGNMENT.CENTER

        hdr = table.rows[0].cells
        _set_cell_text(
            hdr[0],
            "Наименование лабораторной / практической работы",
            bold=True,
            align=WD_ALIGN_PARAGRAPH.CENTER
        )
        _set_cell_text(hdr[1], "Часы", bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)

        for item in lab_topics:
            row = table.add_row().cells
            _set_cell_text(row[0], item.get("name", ""))
            _set_cell_text(row[1], item.get("hours", ""))

    # 5
    _add_heading(doc, "5. Образовательные технологии")
    for item in data.get("education_technologies", []):
        _add_paragraph(doc, f"• {_safe_str(item)}")

    # 6
    _add_heading(
        doc,
        "6. Учебно-методическое обеспечение самостоятельной работы
студентов. "
        "Оценочные средства для текущего контроля успеваемости и
промежуточной аттестации."
    )

    ss_rows = data.get("self_study_rows", [])
    if isinstance(ss_rows, list) and ss_rows:
        table = doc.add_table(rows=1, cols=6)
        table.style = "Table Grid"
        table.alignment = WD_TABLE_ALIGNMENT.CENTER

        hdr = table.rows[0].cells
        headers = ["Недели", "Тема", "Вид работы", "Задание",
"Литература", "Часы"]
        for i, h in enumerate(headers):
            _set_cell_text(hdr[i], h, bold=True,
align=WD_ALIGN_PARAGRAPH.CENTER)

        for item in ss_rows:
            row = table.add_row().cells
            _set_cell_text(row[0], item.get("weeks", ""))
            _set_cell_text(row[1], item.get("topic", ""))
            _set_cell_text(row[2], item.get("kind", ""))
            _set_cell_text(row[3], item.get("task", ""))
            _set_cell_text(row[4], item.get("literature", ""))
            _set_cell_text(row[5], item.get("hours", ""))
```

```
    _add_heading(doc, "6.3. Оценочные средства")
    for item in data.get("assessment_tools", []):
        _add_paragraph(doc, f"• {_safe_str(item)}")

    # 7
    _add_heading(doc, "7. Учебно-методическое и материально-
техническое обеспечение дисциплины")

    _add_paragraph(doc, "а) Литература", bold=True,
align=WD_ALIGN_PARAGRAPH.LEFT)
    for item in data.get("literature", []):
        _add_paragraph(doc, _safe_str(item))

    _add_paragraph(doc, "б) Программное обеспечение", bold=True,
align=WD_ALIGN_PARAGRAPH.LEFT)
    for item in data.get("software", []):
        _add_paragraph(doc, f"• {_safe_str(item)}")

    _add_paragraph(doc, "в) Материально-техническое обеспечение",
bold=True, align=WD_ALIGN_PARAGRAPH.LEFT)
    for item in data.get("equipment", []):
        _add_paragraph(doc, f"• {_safe_str(item)}")

    # Подписи в конце
    _add_paragraph(doc, "", align=WD_ALIGN_PARAGRAPH.LEFT)
    _add_heading(doc, "8. Лист согласования")
    _add_signature_line(doc, "Разработчик рабочей программы")
    _add_signature_line(doc, "Заведующий кафедрой")
    _add_signature_line(doc, "Председатель методической комиссии")

    buffer = io.BytesIO()
    doc.save(buffer)
    buffer.seek(0)
    return buffer.getvalue())
```

ПРИЛОЖЕНИЕ Г

Сгенерированный УП

Таблица Г.1 – Учебный план Бакалавриат 09.03.01 «Информатика и вычислительная техника»

Блок	Семестр	Дисциплина	Часы	Форма контроля	Компетенции ФГОС	Трудовые функции	Обоснование
Блок 1. Обязательная часть	1	Математический анализ	144	экзамен	['УК-1', 'ОПК-1', 'ОПК-2', 'ПК-3']	A/01.3	Дисциплина 'Математический анализ' развивает аналитические навыки и умение применять математические методы для решения задач, что важно для формализации и алгоритмизации задач при разработке программного кода.
Блок 1. Обязательная часть	1	Архитектура вычислительных систем	144	экзамен	['УК-1', 'УК-2', 'ОПК-2', 'ПК-5']	A/01.3	Дисциплина 'Архитектура вычислительных систем' формирует понимание принципов работы вычислительных систем, что необходимо для применения естественнонаучных и общеинженерных знаний (ОПК-1). Также она обучает использованию современных информационных технологий и программных средств (ОПК-2), что важно для алгоритмизации задач (A/01.3).
Блок 1. Обязательная часть	2	Дискретная математика	144	экзамен	['УК-1', 'ОПК-1', 'ПК-3', 'ПК-4']	A/01.3	Дисциплина 'Дискретная математика' развивает навыки анализа и синтеза информации, необходимые для формализации задач, а также умение использовать математические методы в профессиональной деятельности.
Блок 1. Обязательная часть	2	Операционные системы	144	зачёт	['УК-1', 'УК-2', 'ОПК-2', 'ОПК-3', 'ПК-6', 'ПК-5']	A/01.3	Дисциплина 'Операционные системы' формирует умение анализировать информацию и использовать современные технологии для решения задач, что соответствует УК-1 и ОПК-2. Трудовая функция A/01.3 поддерживает эти компетенции, так как требует формализации задач для разработки программного кода.

Блок	Семестр	Дисциплина	Часы	Форма контроля	Компетенции ФГОС	Трудовые функции	Обоснование
Блок 1. Обязательная часть	3	Теория вероятностей и математическая статистика	144	экзамен	['УК-1', 'ОПК-1', 'ПК-10', 'ПК-13']	A/01.3	Дисциплина 'Теория вероятностей и математическая статистика' развивает аналитические навыки и умение работать с данными, что важно для формализации задач и разработки алгоритмов.
Блок 1. Обязательная часть	3	Базы данных	144	экзамен	['УК-1', 'УК-2', 'ОПК-2', 'ОПК-3', 'ПК-8', 'ПК-9']	A/01.3	Дисциплина 'Базы данных' формирует ОПК-2 через изучение современных информационных технологий и программных средств, а также ОПК-1 через применение методов математического моделирования и анализа в контексте работы с базами данных. Трудовая функция A/01.3 поддерживает эти компетенции, так как требует формализации и алгоритмизации задач для разработки программного кода, связанного с базами данных.
Блок 1. Вариативная часть	3	Системное программирование	108	экзамен	['УК-1', 'УК-2', 'ОПК-2', 'ПК-6', 'ПК-5']	A/01.3	Дисциплина 'Системное программирование' формирует ОПК-2 через использование современных инструментов и технологий, а ОПК-1 — через применение математических методов и моделирования в программировании. Трудовая функция A/01.3 поддерживает эти компетенции, обучая алгоритмизации задач.
Блок 1. Вариативная часть	3	Технологии искусственного интеллекта	108	зачёт	['УК-1', 'УК-2', 'ОПК-2', 'ПК-10', 'ПК-11']	A/01.3	Дисциплина 'Технологии искусственного интеллекта' формирует УК-1 через анализ и синтез информации о технологиях ИИ, а ОПК-2 — через использование современных технологий и программных средств в ИИ. Трудовая функция A/01.3 поддерживается, так как студенты учатся формализации задач, что является ключевым для разработки ИИ-систем.
Блок 1. Вариативная часть	3	Технологии интернета вещей	108	зачёт	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Технологии интернета вещей' формирует УК-1 через анализ и синтез информации о технологиях IoT, а ОПК-2 — через использование информационных технологий для разработки IoT-

Блок	Семестр	Дисциплина	Часы	Форма контроля	Компетенции ФГОС	Трудовые функции	Обоснование
							систем. Трудовая функция A/01.3 поддерживается, так как студенты учатся формализации задач для разработки IoT-приложений.
Блок 1. Вариативная часть	3	Технологии программирования для интернета вещей	108	экзамен	['ОПК-2', 'ОПК-1']	A/01.3	Дисциплина формирует навыки использования современных информационных технологий и программных средств, а также применения естественнонаучных и общеинженерных знаний для разработки программного кода.
Блок 1. Обязательная часть	4	Физика	144	зачёт	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Физика' формирует умение применять естественнонаучные знания (ОПК-2) и развивает навыки критического анализа информации (УК-1), необходимые для алгоритмизации задач в ИТ.
Блок 1. Обязательная часть	4	Алгоритмы и структуры данных	144	экзамен	['УК-1', 'УК-2', 'ОПК-1', 'ПК-3']	A/01.3	Дисциплина 'Алгоритмы и структуры данных' развивает навыки анализа информации и применения математических методов, что соответствует УК-1. Также она обучает использованию информационных технологий и программных средств, что соответствует ОПК-2. Трудовая функция A/01.3 (формализация и алгоритмизация задач) напрямую поддерживается этой дисциплиной.
Блок 1. Вариативная часть	4	Разработка мобильных приложений	108	диф. зачёт	['УК-2', 'УК-4', 'ОПК-2', 'ПК-16']	A/01.3	Дисциплина 'Разработка мобильных приложений' формирует УК-1 через анализ и синтез информации при решении задач, а ОПК-2 — через использование современных технологий и программных средств. Трудовая функция A/01.3 поддерживает эти компетенции, обучая алгоритмизации задач для разработки программного кода.

Блок	Семестр	Дисциплина	Часы	Форма контроля	Компетенции ФГОС	Трудовые функции	Обоснование
Блок 1. Вариативная часть	4	Системы управления базами данных	108	зачёт	['УК-1', 'УК-2', 'ОПК-2', 'ОПК-3', 'ПК-8', 'ПК-9']	A/01.3	Дисциплина 'Системы управления базами данных' формирует ОПК-2 через работу с современными информационными технологиями и программными средствами, а ОПК-1 — через применение методов математического анализа и моделирования в контексте управления базами данных. Трудовая функция A/01.3 поддерживает эти компетенции, так как включает формализацию и алгоритмизацию задач, что необходимо для работы с СУБД.
Блок 1. Вариативная часть	4	Технологии машинного обучения	108	экзамен	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Технологии машинного обучения' формирует УК-1 через анализ и синтез данных, а ОПК-2 — через использование информационных технологий. Трудовая функция A/01.3 поддерживается, так как включает алгоритмизацию задач, что является ключевым в машинном обучении.
Блок 1. Вариативная часть	4	Технологии разработки мобильных игр	108	зачёт	['ОПК-2', 'ОПК-1']	A/01.3	Дисциплина 'Технологии разработки мобильных игр' формирует ОПК-2 через использование современных инструментов и технологий, а ОПК-1 — через применение математических и алгоритмических знаний для разработки игр. Трудовая функция A/01.3 поддерживается, так как студенты учатся формализации задач для написания кода игровых приложений.
Блок 1. Обязательная часть	5	Информатика	144	зачёт	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Информатика' формирует умение анализировать информацию и использовать современные информационные технологии, что необходимо для формализации и алгоритмизации задач.
Блок 1. Обязательная часть	5	Теория автоматов и формальных языков	144	экзамен	['УК-1', 'ОПК-1', 'ПК-4']	A/01.3	Дисциплина 'Теория автоматов и формальных языков' развивает навыки анализа и синтеза информации, необходимые для формализации задач (УК-1), а также умение использовать информационные технологии для решения задач (ОПК-2). Трудовая

Блок	Семестр	Дисциплина	Часы	Форма контроля	Компетенции ФГОС	Трудовые функции	Обоснование
							функция A/01.3 поддерживает эти компетенции, обучая формализации задач для разработки программного кода.
Блок 1. Вариативная часть	5	Web-программирование	108	экзамен	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Web-программирование' формирует УК-1 через анализ и синтез информации при разработке веб-приложений, а ОПК-2 — через использование современных информационных технологий и программных средств. Трудовая функция A/01.3 поддерживается, так как студенты учатся формализации и алгоритмизации задач для разработки программного кода.
Блок 1. Вариативная часть	5	Технологии облачных вычислений	108	зачёт	['УК-1', 'УК-2', 'ОПК-2', 'ПК-13']	A/01.3	Дисциплина 'Технологии облачных вычислений' формирует компетенции в области использования современных информационных технологий и программных средств, а также применения естественнонаучных и общеинженерных знаний для решения задач в профессиональной деятельности.
Блок 1. Вариативная часть	5	Технологии блокчейн	108	зачёт	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Технологии блокчейн' формирует УК-1 через анализ и синтез информации о блокчейне, а ОПК-2 — через использование информационных технологий для изучения и разработки блокчейн-систем. Трудовая функция A/01.3 поддерживается, так как дисциплина включает формализацию задач, связанных с анализом и разработкой блокчейн-технологий.
Блок 1. Вариативная часть	5	Технологии разработки кроссплатформенных приложений	108	зачёт	['ОПК-2', 'ОПК-1']	A/01.3	Дисциплина формирует умение использовать современные информационные технологии и программные средства, а также применять естественнонаучные и общеинженерные знания для разработки кроссплатформенных приложений.

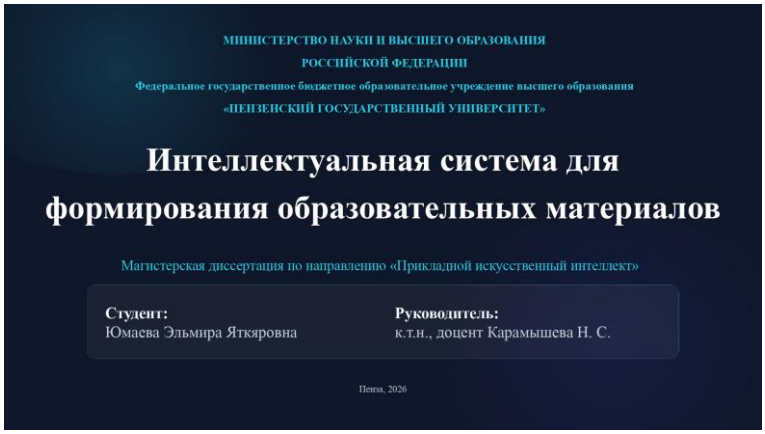
Блок	Семестр	Дисциплина	Часы	Форма контроля	Компетенции ФГОС	Трудовые функции	Обоснование
Блок 1. Обязательная часть	6	Программирование на языке высокого уровня	144	экзамен	['ОПК-2', 'ОПК-1']	A/01.3	Дисциплина формирует умение использовать современные информационные технологии и программные средства, а также применять естественнонаучные и общеинженерные знания для решения задач программирования.
Блок 1. Обязательная часть	6	Сети и телекоммуникации	144	экзамен	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Сети и телекоммуникации' формирует УК-1 через анализ и синтез информации о сетях, а ОПК-2 — через использование информационных технологий для настройки и управления сетями. Трудовая функция A/01.3 поддерживается, так как требует алгоритмизации задач, связанных с настройкой и анализом сетевых параметров.
Блок 1. Вариативная часть	6	Технологии распределённых вычислений	108	зачёт	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Технологии распределённых вычислений' формирует умение анализировать и синтезировать информацию для решения задач (УК-1), а также использовать современные информационные технологии (ОПК-2). Трудовая функция A/01.3 поддерживает эти компетенции, обучая формализации и алгоритмизации задач.
Блок 1. Вариативная часть	6	Технологии больших данных	108	зачёт	['ОПК-2', 'ОПК-1']	A/01.3	Дисциплина 'Технологии больших данных' формирует ОПК-2 через использование современных информационных технологий и программных средств для анализа больших данных, а также ОПК-1 через применение методов математического анализа и моделирования в контексте обработки больших объёмов информации.
Блок 1. Вариативная часть	6	Технологии кибербезопасности	108	экзамен	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Технологии кибербезопасности' формирует УК-1 через анализ угроз и ОПК-2 через использование современных технологий для защиты информации. Трудовая функция A/01.3 поддерживается, так как студенты учатся формализации задач в области кибербезопасности.

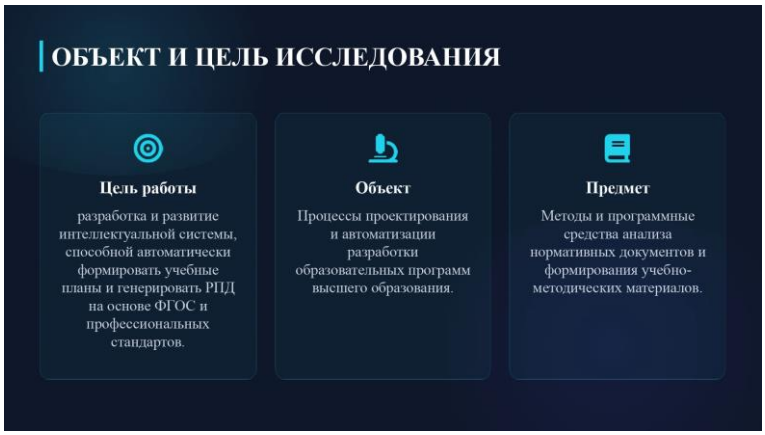
Блок	Семестр	Дисциплина	Часы	Форма контроля	Компетенции ФГОС	Трудовые функции	Обоснование
Блок 1. Вариативная часть	6	Технологии разработки встраиваемых систем	108	зачёт	['ОПК-2', 'ОПК-1']	A/01.3	Дисциплина формирует навыки использования современных информационных технологий и программных средств, а также применения естественнонаучных и общеинженерных знаний для разработки встраиваемых систем.
Блок 1. Вариативная часть	7	Параллельное программирование	108	экзамен	['УК-1', 'УК-2', 'ОПК-2', 'ПК-13', 'ПК-3']	A/01.3	Дисциплина 'Параллельное программирование' формирует ОПК-2 через использование современных технологий и инструментов, а ОПК-1 — через применение математических методов и моделирования в программировании. Трудовая функция A/01.3 поддерживает эти компетенции, обучая алгоритмизации задач для разработки параллельного кода.
Блок 1. Вариативная часть	7	Технологии виртуальной и дополненной реальности в ИТ	108	зачёт	['ОПК-2', 'ОПК-1']	A/01.3	Дисциплина формирует умение использовать современные информационные технологии и программные средства, а также применять естественнонаучные и общеинженерные знания для решения задач в области виртуальной и дополненной реальности.
Блок 1. Вариативная часть	7	Технологии разработки игр	108	зачёт	['УК-1', 'ОПК-2']	A/01.3	Дисциплина 'Технологии разработки игр' формирует УК-1 через анализ и синтез информации при разработке игр, а ОПК-2 — через использование современных технологий и инструментов для создания игровых приложений. Трудовая функция A/01.3 поддерживается через алгоритмизацию задач при разработке игровых механик.
Блок 1. Вариативная часть	7	Технологии разработки робототехнических систем	108	зачёт	['УК-1', 'ОПК-2']	A/01.3	Дисциплина формирует умение анализировать информацию и использовать современные технологии для разработки робототехнических систем, что соответствует УК-1 и ОПК-2. Трудовая функция A/01.3 поддерживает развитие навыков формализации задач.

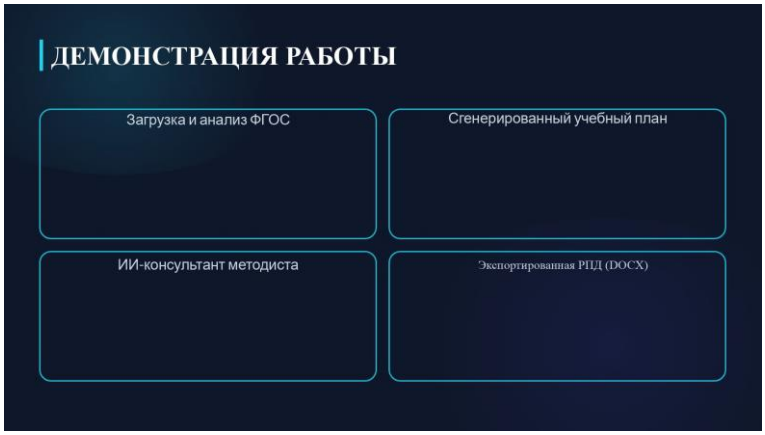
Блок	Семестр	Дисциплина	Часы	Форма контроля	Компетенции ФГОС	Трудовые функции	Обоснование
Блок 2. Практика	7	Учебная практика	108	зачёт	['УК-2', 'УК-3', 'УК-4']		Практика по ФГОС
Блок 2. Практика	8	Преддипломная практика	108	зачёт	['УК-2', 'УК-3', 'УК-4']		Преддипломная практика по ФГОС
Блок 3. ГИА	8	ВКР	216	защита	['УК-2', 'УК-3', 'УК-4']		Государственная итоговая аттестация

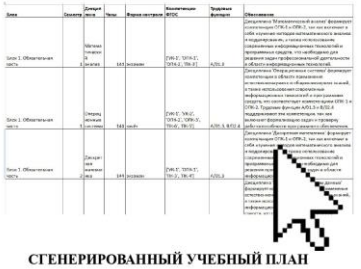
ПРИЛОЖЕНИЕ Д

Презентация









Оценка эффективности системы

Критерий	Ручная подготовка	Разработанная система
Анализ ФГОС	полностью вручную	автоматически
Извлечение компетенций	вручную	автоматически
Формирование дисциплин	вручную	автоматически
Распределение по семестрам	вручную	автоматически
Формирование РПД	вручную	автоматически
Генерация DOCX	вручную	автоматически
Время подготовки учебного плана	от 2 дней	3–7 минут
Время подготовки РПД	от 4 часов	3–7 минут
Необходимость ручного редактирования	высокая	частичная

Гипотеза исследования: использование больших языковых моделей и методов обработки естественного языка при анализе ФГОС и профессиональных стандартов позволит сократить время формирования учебных планов и рабочих программ дисциплин не менее чем в 10 раз по сравнению с полностью ручной подготовкой документов при сохранении полноты структуры документов и корректности извлеченных компетенций.

НОВИЗНА И ПРАКТИЧЕСКАЯ ЗНАЧИМОСТЬ

Научная новизна

Двухступенчатая методика генерации: извлечение данных (lite-модель) → формирование контента (full-модель) с использованием JSON-схем.

Практическая ценность

Система внедрена для сокращения трудозатрат методистов. Обеспечивает единство структуры РПД и соответствие новым ФГОС.

ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Функция	MMIS «Планы»	1С:Университет	LMS (Moodle)	Разраб. система
Хранение данных	✓	✓	✓	×
Анализ текста ФГОС	×	×	×	✓
Извлечение компетенций	×	×	×	✓
Генерация РПД	×	×	×	✓
Использование LLM	×	×	×	✓

ЗАДАЧИ ИССЛЕДОВАНИЯ

- Проанализировать существующие подходы и программные решения для автоматизации формирования учебных планов.
- Исследовать методы NLP и возможности облачных языковых моделей для анализа нормативных документов.
- Разработать архитектуру интеллектуальной системы, включающей модули анализа ФГОС, анализа профстандартов, генерации дисциплин и формирования учебного плана.
- Реализовать модуль автоматической генерации рабочих программ дисциплин (РПД) с использованием YandexGPT и python-docx.
- Разработать веб-интерфейс для взаимодействия пользователя с системой.
- Провести тестирование и экспериментальную оценку работы системы.

АРХИТЕКТУРА СИСТЕМЫ

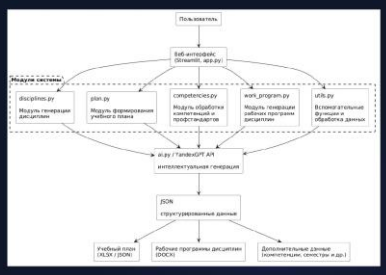
Интерфейс: Streamlit (app.py)

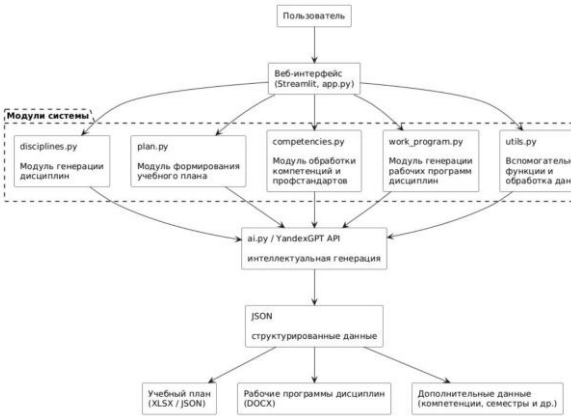
Модули:
- disciplines.py (генерация)
- plan.py (формирование УП)
- competencies.py (обработка ФГОС)

Ядро ИИ: YandexGPT API (ai.py)

Данные: JSON структуры

Экспорт: DOCX / XLSX





ВЫБОР МОДЕЛИ И ИНСТРУМЕНТОВ

YandexGPT API

Двухступенчатая генерация: **lite** для структуры и **full** для содержательной части.

Python & Библиотеки

Pandas (данные), python-docx (РПД), openpyxl (планы), Pytesseract (OCR).

JSON Schema

Driven Prompting: управление структурой ответа модели для стабильной валидации.

МЕТОД: JSON-DRIVEN PROMPTING

Для минимизации галлюцинаций LLM и обеспечения корректной интеграции используется жесткая валидация:

- Системный промпт: задание роли эксперта-методиста.
- JSON-структура: требование ответа в строго заданном формате.
- Верификация: программная проверка наличия всех обязательных полей (УК, ОПК, ПК).

```
prompt_match = f"""
Ты — эксперт по образовательным стандартам РФ.

Тебе даны:
- Компетенции ФГОС (коды и описания).
- Трудовые функции профессионального стандарта (код, название, действия, знания, умения).
Твоя задача — сопоставить компетенции и трудовые функции.
Верни строго JSON:
{{
  "matches": [
    {{
      "competency": "УК-1",
      "related_TF": ["A/01.3"],
      "comment": "Краткое пояснение"
    }}
  ],
  "gaps": [
    {{
      "TF": "B/03.4",
      "reason": "Почему не покрыта"
    }}
  ],
  "recommendations": ["Краткая рекомендация"]
}}
```

System Prompt роль эксперта-методиста → JSON Schema требуемая структура → YandexGPT генерация ответа → JSON Response структурированные данные → Validation проверка обязательных полей → DOCX / XLSX учебный план и РПД

АКТУАЛЬНОСТЬ ИССЛЕДОВАНИЯ
Трудоемкость процесса
Подготовка учебных планов и РПД вручную требует анализа сотен страниц ФГОС и профстандартов.
Регулярные обновления
Необходимость быстрой актуализации программ при изменении нормативной базы (переход на ФГОС 3++).
Человеческий фактор
Риск ошибок при сопоставлении компетенций, индикаторов и дисциплин в документации.
Отсутствие инструментов
Существующие системы (1С, MMIS) решают задачи учета, но не интеллектуального проектирования.

Благодарю за внимание!
Готова ответить на ваши вопросы