Содержание

Введение

Актуальность работы

В современной системе высшего образования и научных исследований Российской Федерации проверка соответствия научных публикаций официальному перечню рецензируемых изданий Высшей аттестационной комиссии (ВАК) [9] является критически важной задачей. Эта процедура необходима как при аттестации научных кадров, так и при оценке исследовательской деятельности вузов и научных организаций. Однако текущий процесс валидации полностью основан на ручной проверке через веб-портал, где необходимо последовательно искать каждый журнал в перечне. Это создаёт существенные трудности, а именно:

- **Отсутствие API:** перечень доступен исключительно в виде PDF-документов, что исключает возможность автоматизированной интеграции;
- **Динамичность данных:** перечень обновляется каждые 1–3 месяца, и необходимо отслеживать исторические версии для проверки старых публикаций;

Нерешенные проблемы

Анализ предметной области выявил следующие критические пробелы:

- Отсутствие стандартизированной структуры данных для хранения информации о перечнях на конкретные даты;
- Невозможность надёжного автоматического парсинга PDF из-за вариативности структуры и форматирования документов;
- Отсутствие механизма версионирования с отслеживанием изменений в составе перечня;
- Недостаток информации о точных датах включения и исключения журналов;
- Наличие аномалий в данных, затрудняющих корректную валидацию;
- Необходимость интеграции с внешними сервисами (такими как CrossRef API или Arxiv API) для получения метаданных публикаций.

Научная новизна работы

Научная новизна данного исследования заключается в преодолении фундаментальных проблем неструктурированных, динамически изменяющихся и зашумленных данных в кон-

тексте научной метрологии и информационного обеспечения науки. В отличие от существующих решений, предлагаемая работа базируется на следующих научных положениях и подходах:

1. **Разработка формальной онтологии и концептуальной модели для представления динамических нормативных перечней.** Впервые предлагается многоаспектная модель данных, которая не просто фиксирует статичный список объектов, а описывает жизненный цикл научного журнала в контексте перечня ВАК. Модель учитывает временную размерность (включение, исключение), статусную атрибутику и связи между различными версиями перечня, что позволяет однозначно определять статус журнала на любую историческую дату и решает проблему «версионного хаоса».
2. **Разработка алгоритмов временного логического вывода для валидации научных публикаций.** В работе предлагаются формальные алгоритмы, которые на основе построенной онтологии и временной метки публикации выполняют проверку её соответствия не просто актуальному списку, а той версии перечня, которая действовала на момент публикации. Этот подход решает ключевую научно-практическую проблему ретроспективной валидации, которая ранее не могла быть решена в автоматизированном режиме из-за отсутствия связанных исторических данных

Содержание по главам

Работа состоит из следующих ключевых разделов:

1. **Глава 1:** анализ существующих методов парсинга, сравнение подходов к концептуализации предметной области, выбор концептуальной модели, технологического стека и их обоснование, постановка задач для реализации.

Практическая значимость работы

Разработанная система может быть использована вузами, научными учреждениями, издательствами и организациями РАН для автоматизации и оптимизации процесса проверки соответствия публикаций перечню ВАК, что существенно снижает временные затраты и риск ошибок.

1. Анализ существующих методов парсинга данных в области документооборота

В настоящей главе представлены результаты изучения существующих методов парсинга данных в области документооборота: парсинг PDF-документов, открытых и закрытых API, способов организации семантического описания предметной области, технологий разработки веб-приложений и их сравнительный анализ применительно к задаче построения системы валидации научных публикаций по официальному перечню рецензируемых научных изданий Российской Федерации.

1.1 Изучение и сравнительный анализ методов концептуализации предметной области

Аннотация. *Раздел посвящён исследованию и сравнительному анализу методов концептуализации предметной области, применимых при проектировании системы валидации научных публикаций. Рассмотрены традиционные и современные подходы к моделированию семантических связей между сущностями (ER-диаграммы, UML, OWL-онтологии, плоские таблицы). Проведено их сравнение по критериям применимости, гибкости, сложности реализации и способности отражать временные аспекты данных. Выделены преимущества комбинированного подхода, основанного на использовании ER-модели для концептуального описания структуры базы данных и UML-диаграмм для отображения семантических и поведенческих связей. Сформулирована трёхуровневая концептуальная модель данных, обеспечивающая поддержку версионности перечней и семантических связей между сущностями «Журнал», «Специальность» и «Версия перечня».*

1.1.1 Основные подходы к выбору методологии построения концептуальных моделей

Выбор адекватной модели концептуализации предметной области является критическиважным этапом проектирования информационной системы. Для задачи валидации научныхпубликаций необходимо выбрать подход, который:

- обеспечивает эффективное хранение исторических версий перечня ВАК;
- поддерживает семантические связи между журналами, специальностями и датами включения/исключения;

– позволяет отвечать на запросы типа «Был ли журнал X в перечне на дату Y?»;
– обладает достаточной гибкостью для обработки аномалий в данных;
– интегрируется с современными системами управления базами данных;
– поддается расширению при изменении требований.

1.1.2 Методы описания концептуальной модели

Для организации извлечённых данных необходимо выбрать модель, которая позволит эффективно хранить исторические версии перечня и поддерживать семантические связи между журналами, специальностями и датами (см. табл. 1.1).

Таблица 1.1 – Сравнение методов концептуализации предметной области

Метод	Описание	Преимущества	Недостатки
ER-диаграмма	Сущности и связи(классический подход)	Простая реализация, хорошо поддерживается SQL БД, хорошо поддерживается ORM	Не отражает временные аспекты
UML-диаграмма	Классы, ассоциации, композиции, агрегация	Гибкость, поддержка наследования	Избыточна для простых случаев
OWL-онтология	Граф знаний, семантика	Расширяемость, логический вывод	Сложность реализации
Плоская таблица	Денормализованная структура	Простота, быстрый поиск	Избыточность данных, аномалии

1.1.3 Выбор метода концептуализации

На основе проведенного анализа для системы валидации выбран **комбинированный подход**:

1. **ER-диаграммы** используются для построения концептуальной модели базы данных с расширением для отслеживания версий перечня и временных интервалов включения журналов;
2. **UML-диаграммы** применяются для построения семантических связей между моделями данных, включая диаграммы классов, сценариев и деятельности, описывающими бизнес-процессы и взаимодействие пользователя с системой.

Трехуровневая концептуальная модель

Трехуровневая концептуальная модель данных обеспечивает поддержку версионности перечней и семантических связей между сущностями «Журнал», «Специальность» и «Версия перечня»:

1. **Слой версий перечня:** Сущность `JournalListVersion` хранит информацию о каждой загруженной версии перечня с указанием даты «по состоянию на» и временной метки загрузки;
2. **Слой журналов и специальностей:** Сущность `Journal` и `ScientificSpecialty` содержат базовые данные о журналах (ISSN, название, издатель, страна) и специальностях (код, категория, наименование);
3. **Слой связей:** Сущность `JournalListing` связывает версию перечня, журнал и диапазон дат включения/исключения, позволяя отслеживать статус журнала на конкретную дату;

Такая архитектура позволяет ответить на ключевой вопрос: «Был ли журнал `Journal` с ISSN `Journal.issn` = «X» включён в перечень ВАК на дату «Y»?» путём запроса к таблице `JournalListing` с условием:

```
/* Абстрактный запрос на получение
статуса журнала на конкретную дату Y */
SELECT * FROM journal_listings
WHERE version_id IN (
    SELECT id FROM journal_list_versions
    WHERE valid_date <= Y
    AND journal_id = (
        SELECT id FROM journals WHERE issn = 'X'
    )
AND date_included <= Y
AND (date_excluded IS NULL OR date_excluded > Y)
)
```

Дополнительные семантические ограничения

Модель включает следующие семантические ограничения целостности:

- **Временная согласованность:** `date_included` всегда предшествует `date_excluded`;
- **Уникальность версий:** на каждую дату может быть загружена только одна версия перечня;
- **Невозможность перекрытия:** для одного журнала в одной версии не допускается наличие пересекающихся временных интервалов включения или исключения;
- **Ссылочная целостность:** все ссылки на журналы и специальности должны указывать на существующие записи в соответствующих таблицах;

1.1.4 Выводы по разделу

Проведённый анализ методов концептуализации показал, что ER-модель в сочетании с UML-диаграммами представляет оптимальное решение для задачи проектирования системы валидации. Такой комбинированный подход обеспечивает:

- **Простоту реализации** благодаря прямому соответствию ER-модели с реляционной схемой БД;
- **Гибкость в представлении** сложных семантических отношений через UML-диаграммы;
- **Поддержку версионности** через трёхуровневую архитектуру с явным отслеживанием временных интервалов;
- **Масштабируемость** за счёт применения нормализации и минимизации избыточности.

Разработанная концептуальная модель служит основой для детального проектирования архитектуры базы данных и определяет интеграционные точки между различными компонентами системы.

1.2 Изучение и сравнительный анализ возможностей семантического описания процессов в предметной области для задачи сценариев выполнения задач

Аннотация. *Раздел посвящён исследованию методологических подходов к семантическому описанию бизнес-процессов валидации научных публикаций. Рассмотрены традиционные и современные методы формализации процессов: IDEF0, UML Activity Diagrams, BPMN, а также текстовое описание. Проведено их сравнение по критериям наглядности, формализма, применимости и практической ценности для данной предметной области. Выделены преимущества комбинированного подхода, основанного на использовании UML Activity Diagrams для визуальной коммуникации и BPMN для формального описания процессов.*

1.2.1 Актуальность семантического описания процессов

В контексте разработки информационной системы валидации научных публикаций критически важной является четкая формализация и документирование бизнес-процессов. Семантическое описание процессов решает следующие задачи

- **Требования к функциональности:** служит основой для формулирования и верификации функциональных требований к системе;
- **Архитектурное проектирование:** определяет структуру модулей, интеграционные точки и потоки данных;
- **Тестирование и валидация:** предоставляет сценарии для разработки тестовых случаев и проверки корректности реализации;
- **Документирование:** создает базис для подготовки пользовательской документации и руководств.

1.2.2 Сравнительный анализ методов формализации процессов

Для описания процесса валидации научной статьи рассмотрены следующие подходы (см. табл. 1.2):

Таблица 1.2 – Сравнение методов формализации процессов

Метод	Применение	Наглядность	Формализм
IDEF0	Функциональное моделирование	Средняя	Высокий
UML Activity Diagrams	Визуальное моделирование процессов	Высокая	Средний
BPMN (Business Process Model and Notation)	Бизнес-процессы, управление потоками	Очень высокая	Высокий
Текстовое описание	Неформальное описание процессов	Низкая	Низкий

Для системы валидации научных публикаций оптимальным является комбинированный подход:

1. **UML Activity Diagrams** используются для визуализации основных процессов и сценариев использования, обеспечивая ясное понимание потоков данных и взаимодействия компонентов системы [10, 5, 7];
2. **BPMN** применяется для формального описания критических и сложных процессов [5, 7, 11], особенно при необходимости интеграции с внешними системами (CrossRef API, системы вузов);

3. **Текстовое описание** используется для дополнительного описания процессов и сценариев использования, обеспечивая гибкость и универсальность при отсутствии необходимости визуального представления.

1.2.3 Выводы по разделу

Проведенный анализ показал, что для эффективного описания процессов в системе валидации необходимо применять комбинированный подход, сочетающий достоинства различных методологий. UML Activity Diagrams обеспечивают необходимую наглядность для коммуникации [10], BPMN предоставляет формальность для реализации [5, 11], а детальные сценарии использования служат мостом между концептуальным описанием и практической реализацией. Такой комплексный подход гарантирует, что все заинтересованные стороны имеют единое понимание функциональности и требований системы, что существенно снижает риск неправильной интерпретации требований и ошибок в реализации.

1.3 Сравнительный анализ серверных технологий для веб-приложений

Аннотация. *В разделе представлен анализ современных серверных технологий и языков программирования, применимых для разработки backend-части системы. Рассмотрены критерии выбора: универсальность языка, наличие библиотек для парсинга PDF, работа с внешними API, поддержка асинхронности и зрелость экосистемы. Проведён анализ выбора языка программирования с акцентом на задачу парсинга и активной работы с БД. На основании анализа обоснован выбор языка Python благодаря широкому набору специализированных библиотек и зрелой инфраструктуре для построения REST API. Выбран фреймворк FastAPI как оптимальный вариант для асинхронной и масштабируемой реализации сервиса,обеспечивающий автоматическую валидацию данных и высокую производительность.*

Для реализации серверной части системы валидации научных статей необходимо выбрать адекватный язык программирования и веб-фреймворк. Выбор обусловлен спецификой задачи: парсинг PDF, работа с внешними API, обработка структурированных данных, асинхронная обработка запросов.

1.3.1 Выбор языка программирования

На основе анализа предметной области и архитектурных требований сформулированы следующие критерии выбора языка:

– **Универсальность и объектно-ориентированность:** возможность разработки как консольных утилит (парсеров), так и веб-сервисов;

- **Наличие библиотек для парсинга данных:** готовые решения для работы с PDF;
- **Качественные веб-фреймворки:** наличие проверенных и хорошо документированных фреймворков;
- **Экосистема и сообщество:** готовые решения, наличие обучающих материалов;
- **Производительность:** возможность асинхронной обработки I/O-bound задач (парсинг, API запросы, взаимодействие с БД);
- **Взаимодействие с ML, CV и OCR:** наличие удобных библиотек для работы с моделями машинного обучения, компьютерного зрения и оптического распознавания текста.

На основе требований предметной области и сформулированных критериев выбора языка программирования выбран язык Python.

1.3.2 Обоснование выбора языка Python

Python обладает множеством преимуществ, которые делают его идеальным выбором для разработки серверной части исследуемой предметной области [3, 12, 4]:

1. **Интеграция с CrossRef API:**
   - Наличие готовых официальных библиотек для работы с CrossRef API (habanero, crossrefapi, crossref-commons);
   - Примеры кода для работы с CrossRef API;
   - Простая работа с JSON-объектами через стандартные структуры данных Python.
2. **Наличие мощных фреймворков для разработки веб-сервисов:**
   - **FastAPI:** — современный, асинхронный фреймворк для разработки RESTful API и микросервисов;
   - **Django:** — монолитный фреймворк со встроенными ”батарейками” для построения больших проектов;
   - **Flask:** — микрофреймворк для быстрой разработки RESTful API.
3. **Развитая экосистема для парсинга данных:**
   - Наличие готовых библиотек для работы с PDF (PyPDF2, pdfminer.six, pdfplumber);
   - Хорошо документированные библиотеки для парсинга сайтов (BeautifulSoup, Scrapy, Playwright);
   - Готовые решения для парсинга данных по API (aiohttp, httpx, requests).
4. **Возможность параллельных вычислений:**
   - Наличие встроенных средств для параллельных вычислений (asyncio, threading, multiprocessing);

- Возможность использования ASGI (асинхронный серверный шлюз) серверов (Uvicorn, Hypercorn);
- Библиотеки для использования WebSocket (socketio, websockets) обеспечивающих двунаправленную и непрерывную связь между клиентом и сервером.
- Библиотеки для использования GraphQL (graphene, strawberry) обеспечивающих гибкую и мощную систему запросов к данным.

5. **Тесное взаимодействие с ML, CV и OCR:**
   - Готовые библиотеки для работы, обучения и интеграции ML-моделей (TensorFlow, PyTorch, scikit-learn);
   - Инструменты для работы с CV (OpenCV, Tesseract Pillow, scikit-image);

1.3.3 Выбор веб-фреймворка

Одним из главных факторов выбора фреймворка для построения backend-части системы является скорость разработки и масштабируемость продукта в будущем (возможность перехода на микросервисную архитектуру, асинхронная обработка I/O-bound задач, возможность подключения асинхронных брокеров сообщений для обработки фоновых задач и общения между другими микросервисами) [2, 3, 8].

Таблица 1.3 – Сравнение веб-фреймворков

Фреймворк	Описание	Преимущества	Недостатки
FastAPI	Быстрый и легкий в использовании для построения REST API	Быстрая разработка, валидация, типизация, ASGI, OpenAPI	Не имеет готовых ”батареек” по типу админ-панели, работы с пользователями.
Django	Монолитный фреймворк для построения веб-приложений	Встроенная ORM, Django-Admin, User Management, встроенная система безопасности	Нет полной поддержки асинхронности. Нет типизации. Нет валидации данных из ”коробки”.
Flask	Микрофреймворк для быстрой разработки RESTful API	Быстрая разработка, легкое масштабирование, хорошая документация	Не подходит для построения сложных API и веб-приложений

Среди имеющихся в Python фреймворков для построения backend-сервиса (см. табл. 1.3) был выбран FastAPI, преимуществами которого являются [2]:

1. Асинхронность из ”коробки”

Работа с запросами к API, взаимодействие с базой данных — I/O-bound задачи (чтение/запись, не требующие процессорных вычислений), которые могут быть выполнены асинхронно без использования дополнительных библиотек, что позволяет повысить производительность сервиса при увеличении количества запросов [2]:

```python
from fastapi import FastAPI, Depends
from typing import Annotated

app = FastAPI()

@app.post('/api/validate')
async def validate_article(
    article: Article,
    doi_service: Annotated[DoiService, Depends(get_doi_service)],
    validation_service: Annotated[
        ValidationService, Depends(get_validation_service)
    ],
    logging_service: Annotated[
        LoggingService, Depends(get_logging_service)
    ]
) -> ResponseSchema:

    # async получение данных из API
    article_data = await doi_service.fetch_article_data(article)

    # async валидация данных
    validation_result = await validation_service.validate_article(
        article_data,
    )
    # async логирование результата в БД
    await logging_service.log_validation_result(
        validation_result
    )
    return validation_result
```

2. Автоматические валидация данных и генерации документации:

FastAPI под капотом использует библиотеку Pydantic для автоматической валидации и преобразования данных (JSON ↔ Python), что позволяет избежать многих ошибок при обработке данных и обеспечивает высокую производительность за счет оптимизации работы с данными:

```
"""Пример автоматической валидации данных"""

from fastapi import FastAPI, Depends
from pydantic import BaseModel

class Article(BaseModel):
    """Модель валидации статьи."""

    # Модель автоматически валидирует
    # поля исходя из их типа данных.
    title: str
    author: str

article = Article(
    title=1,
    author="John Doe",
)
# -> ValidationError: 1 is not a valid string
```

К тому же, FastAPI автоматически генерирует документацию в формате OpenAPI, где можно детально изучить все входные и выходные данные для каждой конечной точки API.

3. Инъекция зависимостей:

FastAPI, как нынче популярно в языке Java, активно использует паттерн проектирования Dependency Injection (DI), с помощью которого можно легко управлять интерфейсами между компонентами системы, обеспечивая высокую модульность и тестируемость кода, благодаря внедрению DI-контейнеров [12, 2, 5]:

```python
from typing import Annotated, ABC, abstractmethod

from fastapi import FastAPI, Depends

class AbstractValidationService(ABC):  # Абстрактный интерфейс
    @abstractmethod
    async def validate(self, article: Article) -> ValidationResult:
        pass  # Реализация в подклассах


class FirstValidationService(AbstractValidationService):
    async def validate(self, article: Article) -> ValidationResult:
        # Логика валидации статьи

class SecondValidationService(AbstractValidationService):
    async def validate(self, article: Article) -> ValidationResult:
        # Другая логика валидации статьи

def get_validation_service(
    service_type: Literal["first", "second"]
) -> AbstractValidationService:
    if service_type == "first":
        return FirstValidationService()
    elif service_type == "second":
        return SecondValidationService()
    raise ValueError(f"Invalid service type: {service_type}")

@app.post('/api/validate')
async def validate_article(  # DI-контейнер
    article: Article, validation_service: Annotated[
        AbstractValidationService, Depends(get_validation_service)
    ]) -> ValidationResult:
    return await validation_service.validate(article)
```

4. Гибкость при работе с базами данных:

FastAPI не навязывает работу с конкретной ORM (Object-Relational Mapping - технология, позволяющая работать с реляционными базами данных как с объектами Python), что позволяет использовать любую ORM или библиотеку для работы с БД, такие как SQLAlchemy, Peewee, Tortoise ORM и т.д.

1.3.4 Выводы по разделу

Среди имеющихся в Python фреймворков рассматривались в основном FastAPI и Django. Выбор FastAPI был обусловлен тем, что он позволяет разработчику быстро создавать сервисы, которые можно будет легко масштабировать и не привязываться к конкретной ORM. В ходе анализа было замечено, что Django обязует разработчика использовать только его ORM и поддерживает ограниченное количество баз данных, с которыми можно работать. В случае же FastAPI сам разработчик может сделать выбор в пользу той или иной ORM, в зависимости от задачи. Также, стоит отметить, что в Django с версии 4.X была добавлена поддержка асинхронности, но она не является полной и не всегда может быть использована, тогда как FastAPI из «коробки» поддерживает параллельную обработку запросов. [8, 2]

1.4 Сравнительный анализ современных фронтенд-фреймворков

Аннотация. *Раздел содержит обзор и сравнительный анализ наиболее популярных фронтенд-фреймворков: Angular, Vue.js и React. На основе данных исследования State of JavaScript 2024 рассмотрены ключевые метрики (Awareness, Usage, Retention, Satisfaction), а также динамика популярности и удовлетворённости разработчиков. Проведён сравнительный анализ архитектуры, производительности, экосистемы и зрелости каждого инструмента. По результатам анализа обоснован выбор библиотеки React как наиболее рационального решения: она обеспечивает высокую производительность за счёт Virtual DOM, обладает крупнейшей экосистемой компонентов и поддерживается ведущими компаниями. React сочетает гибкость, зрелость и масштабируемость, что делает его оптимальным выбором для реализации интерактивного клиентского интерфейса.*

В процессе разработки веб-приложения особое значение имеет выбор технологического стека, определяющего архитектуру, производительность и устойчивость программного решения. Среди современных инструментов для построения интерфейсов были рассмотрены наиболее распространённые фреймворки и библиотеки: **Angular**, **Vue.js** и **React** [6, 4].

1.4.1 Критерии выбора фронтенд-фреймворка

По результатам анализа были сформулированы следующие критерии выбора фронтенд-фреймворка:

- **Архитектурная гибкость:** возможность адаптации структуры приложения под конкретные требования проекта;
- **Производительность:** высокая производительность и масштабируемость приложения;
- **Экосистема:** наличие большого количества библиотек и инструментов для разработки приложения;
- **Сложность освоения:** простота и скорость освоения фреймворка;

1.4.2 Сравнительный анализ современных фронтенд-фреймворков

Для объективного анализа и выбора фронтенд-фреймворка использованы данные ежегодного исследования **State of JavaScript** [6], которое проводится сообществом и является одним из наиболее авторитетных источников статистики по экосистеме JavaScript. Исследование включает опросы более 10,000 разработчиков ежегодно и анализирует тренды в использовании, удовлетворенности и предпочтениях.

Ключевые метрики State of JavaScript:

- **Awareness:** уровень осведомленности разработчиков о фреймворке;
- **Usage:** уровень использования фреймворка;
- **Satisfaction:** уровень удовлетворенности разработчиков;
- **Interest:** процент тех, кто хочет его изучить;
- **Positivity:** процент тех, кто считает использование фреймворка положительным опытом;

Анализ данных State of JavaScript позволяет получить объективную картину о популярности и уровне удовлетворенности разработчиков различными фреймворками.

По результатам анализа данных State of JavaScript за 2024 год были получены следующие результаты:

React занимает лидирующую позицию на рынке фреймворков для фронтенд-разработки по большинству метрик:

- **Awareness:** 99% — практически все разработчики знают о React;
- **Usage:** 82% — используется в 82% проектов, что почти в 2 раза больше, чем у Vue.js и Angular (51% и 50% соответственно);

– **Satisfaction:** 74% — удовлетворенность разработчиков React находится на уровне Vue.js и Angular (79% и 50% соответственно);
– **Interest:** 37% — большинство разработчиков хотели бы изучить React или Vue.js (48%), но не Angular (17%);
– **Positivity:** 69% — большинство разработчиков считают использование React положительным опытом;

Сравнение Angular, Vue.js и React

Таблица 1.4 – Сравнение Angular, Vue.js и React

Критерий	Angular	Vue.js	React
Архитектурная гибкость	Строгая структура, ограниченная свобода выбора инструментов	Поддерживает постепенную интеграцию в существующие приложения и адаптивную модульность	Многократное использование компонентов, возможность использования различных библиотек
Производительность (рендеринг)	Более низкая из-за комплексной архитектуры и большого объёма встроенных модулей.	Эффективный виртуальный DOM, низкие затраты на рендеринг, быстрый отклик интерфейса.	Высокая производительность за счёт использования виртуального DOM
Экосистема	Включает в себя полный стек встроенных решений: маршрутизацию, формы, управление состоянием.	Уступает React по масштабу, но имеет большое количество официальных и сторонних модулей.	Наиболее развитая экосистема с множеством библиотек и инструментов
Сложность освоения	Наибольшая сложность: использование TypeScript, RxJS и концепции модульности	Наиболее проста для изучения: декларативный синтаксис шаблонов и логически разделённая структура файлов	Средняя сложность освоения: использование JSX и необходимость понимания принципов управления состоянием

На основе анализа таблицы 1.4 можно выделить следующие явные преимущества React над Angular и Vue.js применительно к построению фронтенд-части приложения:

– **Архитектурная гибкость:** React позволяет использовать различные библиотеки и инструменты для построения приложения, что позволяет разработчику выбрать наиболее подходящее решение для конкретного проекта.
– **Производительность:** React использует виртуальный DOM, что позволяет достичь

высокой производительности приложения и быструю ре-рендеризацию при частом обновлении данных.

- **Экосистема:** React обладает самой развитой экосистемой среди рассматриваемых фреймворков, что позволяет разработчику найти готовое решение для большинства задач.
- **Сложность освоения:** React имеет среднюю сложность освоения, что в данном случае является преимуществом, так как позволяет разработчику быстро освоить фреймворк.

1.4.3 Выводы по разделу

React является наиболее рациональным и подходящим решением для построения фронтенд-части системы автоматической валидации научных публикаций. Это обусловлено гибкостью, высокой производительностью и широкой поддержкой со стороны сообщества, что говорит о непроблематичности решения возможных вопросов при разработке.

1.5 Выводы по главе 1

В результате проведённого исследования и сравнительного анализа существующих методов и технологий, направленных на автоматизацию валидации научных публикаций по официальному перечню рецензируемых изданий, были получены следующие результаты.

Анализ современного состояния системы документооборота показал, что существующая практика проверки публикаций вручную не отвечает требованиям эффективности и достоверности. Отсутствие открытого API, вариативность форматов PDF-документов и регулярные изменения в перечне ВАК создают предпосылки для ошибок и затрудняют интеграцию с информационными системами научных учреждений. Обоснована необходимость создания автоматизированной системы, обеспечивающей централизованное хранение, обновление и проверку данных с возможностью учёта исторических версий перечня.

В ходе анализа методов концептуализации предметной области установлено, что оптимальным подходом для моделирования данных является комбинированное использование ER-диаграмм для построения концептуальной модели базы данных и UML-диаграмм для описания семантических и поведенческих связей. Разработана трёхуровневая концептуальная модель, включающая слои версий перечня, сущностей и связей, что обеспечивает возможность отслеживания статуса журнала на любую дату и отражение временной динамики данных.

Сравнительный анализ серверных технологий показал, что язык программирования Python в сочетании с фреймворком FastAPI является наиболее целесообразным выбором для реализации серверной части системы. Данный технологический стек обеспечивает поддержку

асинхронной обработки запросов, встроенную валидацию данных, широкие возможности интеграции с внешними API и высокую производительность при работе с I/O-bound задачами, включая парсинг PDF-документов и взаимодействие с CrossRef API.

В рамках анализа современных фронтенд-фреймворков установлено, что библиотека React обладает оптимальным балансом между производительностью, гибкостью и зрелостью эко системы. React обеспечивает модульный компонентный подход, высокую скорость ренде ринга за счёт использования механизма Virtual DOM и широкую поддержку со стороны сообщества разработчиков и индустрии. Выбор данной технологии гарантирует устойчивость и масштабируемость клиентской части веб-приложения.

Таким образом, в первой главе сформирована концептуальная основа проектирования системы: обоснована её актуальность, определены ключевые проблемы и требования, выбраны методы моделирования предметной области и определён технологический стек разработки. Полученные результаты создают методологическую базу для реализации архитектуры и прототипирования автоматизированной системы валидации научных публикаций, что станет предметом рассмотрения в последующих главах.

1.6 Постановка задачи

Цель работы

Целью данной работы является разработка и реализация прототипа информационной системы автоматической валидации научных статей по официальному перечню рецензируемых научных изданий Высшей аттестационной комиссии (ВАК) Российской Федерации, обеспечивающей проверку соответствия публикации перечню с учётом даты публикации статьи и специальности автора. Основной задачей является устранение критической проблемы отсутствия единого программного интерфейса для проверки соответствия научных публикаций перечню ВАК, что в настоящее время требует ручной проверки на сайте портала и затрудняет интеграцию в информационные системы вузов и научных учреждений.

Задачи работы

Для достижения поставленной цели необходимо решить следующие задачи:

1. Построение концептуальной модели базы данных для качественной оценки предметной области;
2. Постановка сценариев выполнения задач в предметной области (проведение семантического анализа и формализации процессов);

3. Описание архитектуры серверной части системы, а также взаимодействия с внешними системами;
4. Проектирование и реализация базы данных с учётом концептуальной модели и сценариев выполнения задач;
5. Разработка RESTful API для взаимодействия с серверной частью системы;
6. Разработка фронтенд-части системы с использованием React;
7. Тестирование и отладка системы;
8. Документирование системы с полным описанием всех API-эндпоинтов и функциональности;

Ожидаемые результаты

Разработанный прототип информационной системы автоматической валидации научных статей по официальному перечню рецензируемых научных изданий Высшей аттестационной комиссии (ВАК) Российской Федерации должен обеспечивать:

- **Работающий прототип системы:** Полнофункциональная информационная система, готовая к использованию и ко вводу в опытную эксплуатацию, состоящая из парсера PDF, серверной части (REST API) и веб-интерфейса.
- **Реляционная СУБД**: Спроектированная и реализованная база данных с полной схемой, включающей все необходимые таблицы, индексы и ограничения целостности.
- **Парсер перечней:** Инструмент автоматического парсинга, обеспечивающий извлечение информации о журналах с точностью 99%, с учётом исторических версий перечня.
- **Веб-интерфейс:** Интерактивный и удобный веб-интерфейс, позволяющий пользователям легко взаимодействовать с системой, просматривать результаты валидации и управлять перечнями.
- **Интерфейс взаимодействия с серверной частью системы:** бэкенд-часть системы, позволяющая интегрировать и использовать её в других сервисах.
- **Тестирование и отладка:** Система должна содержать набор модульных (unit) и интеграционных (integration) тестов, а также бенчмарки для оценки производительности и масштабируемости системы.
- **Техническая документация:** Полная документация проекта, включающая описание архитектуры, документацию по использованию API, инструкции по развёртке, руководство разработчика.

2. Разработка моделей и алгоритмов автоматической валидации научных публикаций

В настоящей главе представлены результаты разработки ключевых моделей и алгоритмов, составляющих методологическую основу системы автоматической валидации. Описана формальная модель предметной области, включающая сущности «Журнал», «Специальность», «Версия перечня» и их семантические связи. Разработан алгоритм парсинга PDF-документов перечней ВАК, учитывающий вариативность форматов и обеспечивающий точность извлечения данных не менее 99%. Представлен метод валидации публикаций, основанный на сопоставлении даты публикации статьи с временными интервалами включения журнала в перечень на соответствующую дату. Описан алгоритм нормализации и дедупликации данных о журналах, поступающих из различных источников (PDF-перечни, CrossRef API). Разработана обобщенная архитектура системы, определяющая взаимодействие основных компонентов: парсера, базы данных, REST API и веб-интерфейса. Представлены интерфейсы модулей и протоколы обмена данными между компонентами системы.

3. Проектирование системы автоматической валидации научных публикаций

В настоящей главе представлены детальные результаты проектирования системы автоматической валидации научных публикаций. Сформулированы функциональные требования, включающие возможности загрузки и парсинга перечней ВАК, валидации публикаций по ISSN и дате, поиска журналов по специальностям, просмотра истории изменений перечня. Разработаны диаграммы вариантов использования (use case), описывающие сценарии работы различных категорий пользователей. Представлена трёхуровневая архитектура системы (presentation layer, business logic layer, data layer) с детализацией компонентов каждого уровня и протоколов их взаимодействия. Спроектирована физическая модель базы данных, включающая схему таблиц с индексами, внешними ключами и триггерами для поддержки версионности и обеспечения целостности данных. Обоснован выбор технологического стека: Python 3.11+ с фреймворком FastAPI для реализации REST API, React для frontend-части сервиса, а также выбор СУБД PostgreSQL. Детально описаны проектные решения по организации модульной архитектуры backend с применением паттернов Dependency Injection, Repository и Service Layer, обеспечивающих тестируемость и поддерживаемость кода.

4. Программная реализация и экспериментальная проверка системы автоматической валидации научных публикаций

В настоящей главе представлены результаты практической реализации и экспериментальной проверки разработанной системы. Описан состав и структура реализованного программного обеспечения: backend-сервис на FastAPI, включающий модули парсинга PDF-документов, работы с базой данных через SQLAlchemy ORM, интеграции с CrossRef API и REST API эндпоинты; frontend-приложение на React с компонентами валидации публикаций, поиска журналов, просмотра истории перечней и административной панели; база данных PostgreSQL с реализованной схемой и системой миграций. Представлены основные сценарии работы пользователей с описанием интерфейсов: для исследователей — форма валидации публикации с получением мгновенного результата проверки; для администраторов — загрузка и обработка PDF-перечней, мониторинг системы; для внешних систем — интеграция через REST API с примерами запросов и ответов. Разработаны тестовые примеры, включающие модульные тесты для алгоритмов парсинга и валидации, интеграционные тесты для проверки взаимодействия компонентов.

Список литературы

[1] *CrossRef API*. Англ. URL: https://www.crossref.org/documentation/retrieve-metadata/rest-api/ (дата обр. 19 окт. 2025).

[2] Bill Lubanovic. *FastAPI. Modern Python Web Development*. Англ. O'Reilly Media, Inc., 2024. URL: https://www.oreilly.com/library/view/fastapi-modern/9781492090233/ (дата обр. 13 окт. 2025).

[3] Bill Lubanovic. *Introducing Python*. Англ. O'Reilly Media, Inc., 2024. URL: https://www.oreilly.com/library/view/introducing-python-2nd/9781492051374/ (дата обр. 15 окт. 2025).

[4] Debani P. M., Surender R. S. и Kshirod K. R. «Modern tools and current trends in web-development». Англ. В: *Indonesian Journal of Electrical Engineering and Computer Science* (). DOI: 10.11591/ijeecs.v24.i2.pp978-985. URL: https://www.researchgate.net/publication/355948452_Modern_tools_and_current_trends_in_web-development (дата обр. 18 окт. 2025).

[5] Laguna M. и other. *Business process modelling, simulation and design*. Англ. CRC Press, 2013. DOI: 10.1201/b14763. URL: https://www.taylorfrancis.com/books/mono/10.1201/b14763/business-process-modeling-simulation-design-johan-marklund-manuel-laguna (дата обр. 13 окт. 2025).

[6] *State of JavaScript*. Англ. 2024. URL: https://stateofjs.com/en-US (дата обр. 18 окт. 2025).

[7] Sergey Kosikov Viacheslav Wolfengagen Larisa Ismailova. «Semantic neighborhood in cognitive modeling». Англ. В: *Cognitive Systems Research* (2025). DOI: 10.1016/j.cogsys.2025.101398. URL: https://www.sciencedirect.com/science/article/pii/S1389041725000786?via%3Dihub (дата обр. 29 сент. 2025).

[8] Zhanymkanov. *FastAPI Best Practices*. Англ. 2025. URL: https://github.com/zhanymkanov/fastapi-best-practices (дата обр. 13 окт. 2025).

[9] *Высшая аттестационная комиссия (ВАК) Российской Федерации*. URL: https://vak.gisnauka.ru/documents/editions (дата обр. 16 окт. 2025).

[10] Чисников П. И. «Unified Modeling Language (UML)». В: *Экономика, статистика и информатика. Вестник УМО* (2010). URL: https://cyberleninka.ru/article/n/unified-modeling-language-uml/viewer (дата обр. 15 окт. 2025).

[11] Новикова Г. М. и Малютина Т. В. «Развитие инструментов моделирования предприятия для корпоративных систем управления». enlish. В: (2011). URL: https://cyberleninka.ru/article/n/the-development-of-enterprise-modeling-instrument-for-corporate-management-systems/viewer (дата обр. 15 окт. 2025).

[12] Лавров Д. Н. и Лаврова Д. Д. «SOLID против GRASP». В: *Математические структуры и моделирование* (2025). DOI: 10.24147/2222-8772.2025.1.125-135. URL: https://msm.omsu.ru/jrns/jrn73/Lavrovs_73.pdf (дата обр. 17 окт. 2025).