МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ

ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»



Направление подготовки   
09.03.01 Информатика и вычислительная техника

Направленность (профиль)

«Технологии разработки программного обеспечения»

**Выпускная квалификационная работа**

Электронный справочник "Сети и телекоммуникации"

|  |
| --- |
| Обучающегося 4 курса  очной формы обучения  Банакова Виктора Антоновича |
|  |
| Руководитель выпускной квалификационной работы:  старший преподаватель кафедры информационных технологий и электронного обучения  Аксютин Павел Александрович |
|  |

Санкт-Петербург

2025

Оглавление

[1. Введение 4](#_Toc201049796)

[1.2. Предмет и задачи исследования 5](#_Toc201049797)

[1.3. Тема и объект изучения 6](#_Toc201049798)

[1.4. Приемы исследования 7](#_Toc201049799)

[2. Анализ существующих исследований и аналогов 9](#_Toc201049800)

[2.1.Развитие телекоммуникационных сетей: краткий исторический обзор 9](#_Toc201049801)

[2.2. Тенденции развития сетевых технологий в современном мире 9](#_Toc201049802)

[2.3. Изучение актуальных электронных справочников 10](#_Toc201049803)

[3. Основы создания электронных справочников: теоретический аспект 12](#_Toc201049804)

[3.1. Организация и дизайн базы данных 12](#_Toc201049805)

[3.2. Дизайн и интерфейс пользователя 12](#_Toc201049806)

[3.3. Характерные черты поиска и фильтрации данных 13](#_Toc201049807)

[4.Создание электронного справочника "Сети и телекоммуникации". 14](#_Toc201049808)

[4.1. Определение целей и техническое задание 14](#_Toc201049809)

[4.2. Определение оптимальных технологий и инструментов для разработки 15](#_Toc201049810)

[4.3. Внедрение базы данных 15](#_Toc201049811)

[4.4. Осуществление интерфейса 16](#_Toc201049812)

[4.5. Проверка работоспособности системы 16](#_Toc201049813)

[5. Электронный справочник "Сети и телекоммуникации" (микросервисная архитектура) 18](#_Toc201049814)

[5.1. Услуга аутентификации (Auth Service) 18](#_Toc201049815)

[5.2. Услуга контента (Content Service) 20](#_Toc201049816)

[5.3. Пример информационного ресурса (база знаний) 22](#_Toc201049817)

[5.4. Фронтенд (React.js) 24](#_Toc201049818)

[5.5. Внедрение (Docker Compose) 26](#_Toc201049819)

[5.6. Функции расширенного функционала 27](#_Toc201049820)

[6.Анализ электронного справочника "Сети и телекоммуникации" с микросервисной архитектурой 30](#_Toc201049821)

[6.1. Анализ архитектуры 30](#_Toc201049822)

[6.2. Изучение кода и аспектов безопасности 30](#_Toc201049823)

[6.3. Производительность и данные 31](#_Toc201049824)

[6.4. Фронтенд-анализ 32](#_Toc201049825)

[6.5. Развертывание и DevOps 33](#_Toc201049826)

[6.6. Тестирование 34](#_Toc201049827)

[6.7. Рекомендации по развитию 35](#_Toc201049828)

[6.8.Заключение: 35](#_Toc201049829)

[7. Заключение 36](#_Toc201049830)

[8. Список литературы 37](#_Toc201049831)

# 1. Введение

**1.1. Востребованность темы**

В условиях стремительной цифровой трансформации сети и телекоммуникации превратились в жизненно важную основу современного мира. Согласно данным Международного союза электросвязи (ITU), число пользователей интернета во всем мире уже превысило 5 миллиардов, а глобальный объем интернет-данных неуклонно увеличивается, демонстрируя ежегодный рост на 25-30%. Это обстоятельство делает актуальной потребность в высококвалифицированных специалистах и эффективных образовательных инструментах.

Разработка электронного справочника "Сети и телекоммуникации" сегодня особенно актуальна по ряду ключевых причин:

1. Перенасыщение информацией:

Каждый год появляются десятки новых стандартов и протоколов, таких как 5G, Wi-Fi 6, IPv6 и другие.

Учебные материалы часто оказываются фрагментарными и не всегда успевают acompanhar за rapidными изменениями в технологической сфере.

2. Необходимость в практическом применении:

Согласно опросу Cisco, 68% специалистов в сфере ИТ сталкиваются с трудностями при поиске актуальных технических данных.

Обучающие программы вузов часто оказываются на 3-5 лет отстающими от актуальных потребностей рынка труда.

3. Тенденции в технологиях:

Эволюция SDN, NFV и облачных технологий диктует необходимость в новых методах обучения.

Эпоха микросервисных архитектур и контейнеризации влечет за собой переосмысление основополагающих принципов построения сетей.

Данный справочник особенно ценен для:

- Абсолютно все студенты технических направлений

- Системных администраторов и инженеров

Ученых и преподавателей высших учебных заведений и учебных центров

Создателей сетевых приложений

## 1.2. Предмет и задачи исследования

Основной задачей проекта является создание всеобъемлющего электронного справочника, в котором теоретические основы и практические применения современных сетевых технологий будут объединены в единой интерактивной платформе.

Определенные задачи исследования:

1. Аналитические -

Выполнить комплексный анализ современных сетевых технологий и протоколов, актуальных в период с 2018 по 2023 год.

Выполнить анализ более 15 действующих образовательных платформ и справочных систем.

Определить слабые места существующих решений посредством сравнительного анализа.

2. Проектные-ориентированные:

Создать онтологию для данной предметной области.

Разработать модульную архитектуру для системы.

- Выбрать наиболее подходящую технологическую платформу для реализации

3. Аспект технический:

- Создать сердце системы, оснащенное следующими функциями:

Поиск с разными уровнями глубины (по всему тексту, по тегам, семантический).

Представление сетевых топологий визуально

- Взаимодействующие лабораторные эксперименты

Обеспечить доступность на всех платформах (Веб, Десктоп, Мобильные устройства).

4. Оценочные -

Протестировать удобство использования продукта с группой из 20+ участников.

Проверить эффективность обучения с помощью сравнительного тестирования A/B.

Создать методику, описывающую процесс интеграции с платформами LMS (Moodle, Blackboard).

## 1.3. Тема и объект изучения

Предметом исследования являются цифровые образовательные ресурсы, посвящённые сетевым технологиям и телекоммуникациям.

В рамках исследования рассматриваются методы и алгоритмы, применяемые для разработки интерактивного электронного справочника, обладающего функциями адаптивного обучения.

В данной работе исследуются:

Развитие сетевых технологий: от модели OSI до современных облачных решений.

Модели когнитивного усвоения технической информации

Принципы эргономики в проектировании интерфейсов сложных технических систем

## 1.4. Приемы исследования

Данное исследование построено на комплексной основе, объединяющей:

Теоретические подходы:

Анализ системы (методология SADT)

Моделирование онтологий (Protégé)

Изучение динамики временных рядов для выявления технологических тенденций

2. Методы эмпирического исследования:

Тестирование юзабилити по принципам Якоба Нильсена

Обучение в группах-контроле, основанное на экспериментальных данных.

Тестирование производительности (JMeter, Selenium)

3. Стек технологий:

Server-side: Built with Python 3.10, utilizing FastAPI and SQLAlchemy.

Представление: React.js с использованием D3.js для визуализации.

База данных: комбинация PostgreSQL и Elasticsearch.

DevOps Practices: Leveraging Docker, Kubernetes, and CI/CD Pipelines (using GitLab)

4. Критерии оценки:

- Охватом тем обеспечено не менее 90% ключевых концепций.

Поиск информации осуществляется со скоростью менее 500 миллисекунд.

Уровень удовлетворенности пользователей (NPS в диапазоне от 8 и выше)

Особое акцент было поставлено на принципах:

- Топ-10 наиболее распространенных уязвимостей информационной безопасности (OWASP).

- Соблюдение стандарта доступности WCAG 2.1

Персонализация контента (адаптивность)

Настоящая методология обуславливает научную значимость работы, поскольку в ней реализована:

1.Создание алгоритма для семантического анализа технических текстов.

2.Разработка интерактивной системы для визуализации сетевых топологий.

3. Внедрение системы адаптивного тестирования знаний.

# 2. Анализ существующих исследований и аналогов

## 2.1.Развитие телекоммуникационных сетей: краткий исторический обзор

Телекоммуникационные сети совершили впечатляющее путешествие эволюции, пройдя путь от простых методов передачи сигналов к современным высокоскоростным цифровым технологиям.

В доисторическую эпоху для связи на расстоянии использовались дымовые сигналы, почтовые голуби и гелиографы. К примеру, в Древнем Риме для передачи сообщений на значительные расстояния применялась технология световых сигналов, основанная на использовании зеркал.

Эпоха электричества: изобретение телеграфа в 1835 году, телефона в 1876 году и радио в 1895 году революционизировало коммуникации. Появление первых трансатлантических кабелей в 1866 году и автоматических телефонных станций в 1889 году заложило фундамент для современных коммуникационных сетей.

Развитие вычислительной техники в XX веке породило создание локальных и глобальных сетей, позволяющих компьютерам взаимодействовать и обмениваться информацией. Возникновение интернета (ARPANET, 1969 год) стало поворотным моментом в истории телекоммуникаций, открыв новые горизонты для связи и информационного обмена.

## 2.2. Тенденции развития сетевых технологий в современном мире

В настоящее время телекоммуникационные технологии претерпевают развитие в ряде ключевых направлений:

Повышение скорости и пропускной способности достигается благодаря технологиям, например, WiGig (7 Гбит/с) и оптоволоконным сетям, которые обеспечивают передачу данных с невероятной скоростью до 58 Гбит/с.

Интернет вещей (IoT) - это концепция подключения к сети бытовых приборов, датчиков и промышленных установок. Ожидается, что к 2030 году сеть IoT будет насчитывать более 50 миллиардов устройств.

Новые стандарты связи, такие как 5G и беспроводные технологии, характеризуются низкими задержками и высокой скоростью передачи данных, что является ключевым фактором для бесперебойной работы автономных систем и телемедицинских платформ.

Искусственный интеллект играет ключевую роль в кибербезопасности, применяясь как для совершенствования сетевых процессов, так и для защиты данных от угроз кибератак.

## 2.3. Изучение актуальных электронных справочников

Была проанализирована тематическая направленность существующих электронных ресурсов, посвященных сетям и телекоммуникациям:

Электронные энциклопедии (такие как Википедия):

Преимущества: охватывает широкий спектр тем, доступен бесплатно.

Недостатки: Данные могут быть не всегда актуальными, а учебные материалы не всегда структурированы.

Платформы, ориентированные на узкую специализацию (Cisco Learning Network, ITU-T)

Преимущества: Качественный, тщательно подобранный контент, интерактивные обучающие курсы.

Недостатки: Ограниченный бесплатный доступ, сложность освоения для начинающих.

Приложения для мобильных устройств (такие как "Networking Basics"):

Преимущества: простота использования, возможность работы без подключения к интернету.

Недостатки: Функциональность ограничена, отсутствуют глубокие разделы.

# 3. Основы создания электронных справочников: теоретический аспект

## 3.1. Организация и дизайн базы данных

В этой части статьи мы рассмотрим ключевые принципы, лежащие в основе проектирования базы данных для электронного справочника. К главным аспектам относятся:

* Различные типы моделей данных (реляционные, документоориентированные и графовые) и их целесообразность в создании справочников.
* Оптимизация базы данных путем нормализации для сокращения избыточности и гарантирования целостности информации.
* Базовые элементы и отношения: таблицы (такие как «Оборудование», «Протоколы», «Термины»), а также первичные и внешние ключи, определяющие связи между ними.
* Организация и хранение медиаданных, таких как схемы, изображения и видео, реализуется посредством различных подходов к управлению файлами или BLOB-объектами.

Пример:

В хранилище данных о сетевом оборудовании применяется реляционная модель, состоящая из таблиц Vendors (производители), Devices (устройства) и Specifications (характеристики), где связи между ними организованы по схеме "один ко многим".

## 3.2. Дизайн и интерфейс пользователя

Рассматриваются требования, предъявляемые к интерфейсу электронного справочника.

* Ключевые принципы UX/UI: легкость навигации, адаптация к разным устройствам и соблюдение стандартов доступности (WCAG).
* В состав типовых компонентов входят древовидное меню для организации разделов, карточки для представления объектов и интерактивные схемы.
* Представление данных: таблицы и графики (например, для иллюстрации различий в пропускной способности различных протоколов).
* Оптимизация для разных устройств: адаптивный дизайн для компьютеров, планшетов и мобильных телефонов.

Для примера:

Справочник начинается с главной страницы, которая содержит поисковую строку, разделение на категории (такие как «Кабели», «Маршрутизаторы») и блок «Последние добавленные материалы», где каждый элемент представлен иконкой, облегчающей быструю идентификацию типа контента.

## 3.3. Характерные черты поиска и фильтрации данных

Эффективные методы поиска информации в справочнике:

* Категории поиска:
  + Полный текст, проанализированный по ключевым словам, с учетом морфологии.
  + Фильтрация по параметрам, например, "Тип кабеля: витая пара", реализуется с помощью атрибутивных фильтров.
* Обработка запросов с использованием искусственного интеллекта: автоматическое предсказание слов (autocomplete) и исправление орфографических ошибок.
* Фильтрация позволяет задавать комбинированные условия поиска, например, «отображать только Wi-Fi 6 роутеры, стоимость которых не превышает 10 000 рублей».
* Ускорение навигации за счет внедрения тегов и ключевых слов.

Пример:

Чтобы осуществлять поиск по стандартам Ethernet, таким как 802.3ab, предусмотрено фильтрование по следующим характеристикам: скорость передачи данных (1/10/100 Гбит), используемая среда (оптическая или медная) и год утверждения стандарта.

# 4.Создание электронного справочника "Сети и телекоммуникации".

## 4.1. Определение целей и техническое задание

В этом разделе излагаются ключевые требования к электронному справочнику, а также устанавливаются его функциональные и не функциональные особенности.

Требования к функциональности:

Функция поиска по ключевым словам.

Классификация данных (оборудование, технологии, протоколы и аналогичные).

Администратор имеет возможность добавлять, редактировать и удалять записи.

Возможность экспортировать данные в удобных форматах, таких как PDF и CSV.

Требования, не относящиеся к функциональности:

Приспособленность к работе на различных платформах, включая ПК и мобильные устройства.

Прост в использовании и понятен с первого взгляда.

- Обеспечение безопасности данных (включая аутентификацию пользователей по запросу).

В техническом задании содержится:

- Задачи разработки.

- Характеристика целевого сегмента.

Список модулей системы.

- Необходимости в производительности и защищенности.

## 4.2. Определение оптимальных технологий и инструментов для разработки

В данном разделе будет рассмотрены обоснование выбора программного и аппаратного обеспечения для разработки справочника.

Представление (интерфейс):

Спектр моих языков: HTML5, CSS3 и JavaScript, а также возможное использование фреймворков React или Vue.js.

UI-библиотеки, среди которых выделяются Bootstrap и Material-UI.

Серверная часть (логика и хранилище данных):

Для серверной части можно использовать комбинацию Node.js с Express или Python с Django/Flask.

- Хранилище данных: SQL (MySQL, PostgreSQL) или NoSQL (MongoDB), выбор зависит от специфики хранимой информации.

Другие инструменты:

Git - это система контроля версий.

- Docker предназначен для контейнеризации программных приложений.

Postman – это инструмент, предназначенный для тестирования API.

## 4.3. Внедрение базы данных

Пошаговый процесс разработки и настраивания базы данных:

1. Разработка схемы БД:

Идентификация сущностей (терминов, категорий, пользователей).

Разработка ER-диаграммы.

Оптимизация таблиц.

2. Реализация:

Создание SQL-скриптов, предназначенных для формирования таблиц.

- Установка взаимосвязей (используя первичные и внешние ключи).

Проведение загрузки тестовыми данными.

3. Совместимость с приложением:

Использование ORM (Sequelize, SQLAlchemy) для работы с базой данных или выполнение запросов напрямую через SQL.

Выполнение операций CRUD.

## 4.4. Осуществление интерфейса

Поэтапный подход к созданию пользовательского интерфейса:

1. Создание прототипов:

Разработка wireframe (Figma, Adobe XD).

- Выявление и описание UX-логики: навигации, форм и фильтров.

2. Макет:

Дизайн, адаптирующийся к различным устройствам.

Внедрение интерактивных функций, таких как поиск и сортировка.

3. Инициализация связи с бэкендом:

Настройка взаимодействия с API (использование Fetch и Axios).

Обработка результатов и представление информации.

## 4.5. Проверка работоспособности системы

Тестирование работоспособности справочника:

1. Классификация тестирования:

Тестирование по модулям (Unit Testing) – это процесс проверки работы отдельных функций по отдельности.

Интеграция – это процесс взаимодействия между фронтендом и бэкендом.

Тестирование, направленное на проверку соответствия всей системы требованиям технического задания, носит системный характер.

\*Юзабилити-тестирование\* - это процесс оценки эргономичности и простоты использования интерфейса.

2. Средства:

Testing frameworks like Jest and Mocha are used for JavaScript.

- Фреймворк Selenium (в области автоматизации тестирования пользовательских интерфейсов).

3. Коррекция ошибок:

Изучение логов.

Повышение производительности.

В результате работы создана стабильная версия электронного справочника, которая может быть сразу же внедрена в эксплуатацию.

# 5. Электронный справочник "Сети и телекоммуникации" (микросервисная архитектура)

**Схематическое представление архитектуры**

Клиент (Web/Mobile) → API Gateway → [Auth Service | Content Service | Search Service | Analytics Service]

↑

↓

PostgreSQL + and Redis Combined

## 5.1. Услуга аутентификации (Auth Service)

**Реализация с использованием Node.js (фреймворк Express + токены JWT)**

javascript

*// auth-service/index.js*

const express = require('express');

const jwt = require('jsonwebtoken');

const bcrypt = require('bcryptjs');

const { Pool } = require('pg');

const app = express();

app.use(express.json());

const pool = new Pool({

user: 'postgres',

host: 'auth-db',

database: 'auth',

password: 'securepassword',

port: 5432,

});

*// Зарегистрировать пользователя*

app.post('/register', async (req, res) => {

const { username, email, password, role = 'user' } = req.body;

try {

const hashedPassword = await bcrypt.hash(password, 10);

const result = await pool.query(

'INSERT INTO users (username, email, password\_hash, role) VALUES ($1, $2, $3, $4) RETURNING id',

[username, email, hashedPassword, role]

);

res.status(201).json({ userId: result.rows[0].id });

} catch (err) {

res.status(500).json({ error: err.message });

}

});

*// Утверждение личности*

app.post('/login', async (req, res) => {

const { email, password } = req.body;

const user = await pool.query('SELECT \* FROM users WHERE email = $1', [email]);

if (!user.rows.length) return res.status(401).send('User not found');

const isValid = await bcrypt.compare(password, user.rows[0].password\_hash);

if (!isValid) return res.status(401).send('Invalid credentials');

const token = jwt.sign(

{ userId: user.rows[0].id, role: user.rows[0].role },

process.env.JWT\_SECRET,

{ expiresIn: '1h' }

);

res.json({ token });

});

app.listen(3000, () => console.log('Auth service running on port 3000'));

## 5.2. Услуга контента (Content Service)

**Реализация на Python (FastAPI)**

python

*# content-service/main.py*

from fastapi import FastAPI, Depends, HTTPException

from fastapi.security import OAuth2PasswordBearer

from pydantic import BaseModel

import asyncpg

import jwt

app = FastAPI()

oauth2\_scheme = OAuth2PasswordBearer(tokenUrl="auth/login")

DATABASE\_URL = "postgresql://user:password@content-db:5432/contentdb"

*# Модели данных*

class NetworkConcept(BaseModel):

title: str

category: str

content: str

related\_concepts: list[str] = []

*# Соединение с базой данных*

async def get\_db():

return await asyncpg.connect(DATABASE\_URL)

*# Валидация JWT токена*

async def get\_current\_user(token: str = Depends(oauth2\_scheme)):

try:

payload = jwt.decode(token, "secret", algorithms=["HS256"])

return payload

except jwt.PyJWTError:

raise HTTPException(status\_code=401, detail="Invalid token")

*# API Access Points*

@app.post("/concepts")

async def create\_concept(concept: NetworkConcept,

db=Depends(get\_db),

user=Depends(get\_current\_user)):

if user["role"] not in ["admin", "editor"]:

raise HTTPException(status\_code=403, detail="Forbidden")

await db.execute(

"INSERT INTO concepts (title, category, content) VALUES ($1, $2, $3)",

concept.title, concept.category, concept.content

)

return {"status": "created"}

@app.get("/concepts/{concept\_id}")

async def get\_concept(concept\_id: int, db=Depends(get\_db)):

record = await db.fetchrow(

"SELECT \* FROM concepts WHERE id = $1", concept\_id

)

return dict(record) if record else None

## 5.3. Пример информационного ресурса (база знаний)

**Структура данных в PostgreSQL**

sql

*-- Таблица сетевых концепций*

CREATE TABLE concepts (

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

category VARCHAR(100) NOT NULL,

content TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

*-- Список оборудования*

CREATE TABLE equipment (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

manufacturer VARCHAR(100),

specifications JSONB,

image\_url VARCHAR(255)

);

**Иллюстрации наполнения:**

**Сетевые протоколы:**

json

{

"title": "TCP/IP",

"category": "Протоколы",

"content": "Набор сетевых протоколов...",

"related\_concepts": ["OSI", "UDP", "IPv4"]

}

**Аппаратура сетей:**

json

{

"name": "Cisco Catalyst 2960",

"manufacturer": "Cisco",

"specifications": {

"ports": "24x 10/100 Ethernet",

"stacking": true,

"power": "PoE+"

}

}

## 5.4. Фронтенд (React.js)

**Элемент аутентификации**

jsx

*// src/components/Auth.jsx*

import React, { useState } from 'react';

import axios from 'axios';

const Auth = () => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const handleLogin = async () => {

try {

const response = await axios.post('http://api-gateway/auth/login', {

email,

password

});

localStorage.setItem('token', response.data.token);

} catch (error) {

console.error('Login failed:', error);

}

};

return (

<div>

<input type="email" onChange={(e) => setEmail(e.target.value)} />

<input type="password" onChange={(e) => setPassword(e.target.value)} />

<button onClick={handleLogin}>Login</button>

</div>

);

};

**Элемент отображения контента**

jsx

*// src/components/ConceptViewer.jsx*

import React, { useEffect, useState } from 'react';

import axios from 'axios';

const ConceptViewer = ({ conceptId }) => {

const [concept, setConcept] = useState(null);

useEffect(() => {

const fetchConcept = async () => {

const token = localStorage.getItem('token');

const response = await axios.get(

`http://api-gateway/content/concepts/${conceptId}`,

{ headers: { Authorization: `Bearer ${token}` } }

);

setConcept(response.data);

};

fetchConcept();

}, [conceptId]);

return (

<div className="concept-container">

{concept && (

<>

<h2>{concept.title}</h2>

<div className="category-badge">{concept.category}</div>

<div dangerouslySetInnerHTML={{ \_\_html: concept.content }} />

</>

)}

</div>

);

};

## 5.5. Внедрение (Docker Compose)

yaml

version: '3.8'

services:

api-gateway:

image: nginx

ports:

- "80:80"

volumes:

- ./nginx.conf:/etc/nginx/nginx.conf

auth-service:

build: ./auth-service

environment:

- JWT\_SECRET=supersecretkey

depends\_on:

- auth-db

content-service:

build: ./content-service

environment:

- DATABASE\_URL=postgresql://user:password@content-db:5432/contentdb

depends\_on:

- content-db

auth-db:

image: postgres:13

environment:

- POSTGRES\_PASSWORD=securepassword

- POSTGRES\_DB=auth

content-db:

image: postgres:13

environment:

- POSTGRES\_PASSWORD=password

- POSTGRES\_DB=contentdb

volumes:

- ./init.sql:/docker-entrypoint-initdb.d/init.sql

## 5.6. Функции расширенного функционала

**Позиции пользователей:**

* + Пользователь с полным доступом, Режим редактирования (вставка/изменение контента)
  + Пользовательская активность (просмотр, оставляемые комментарии)
  + Пользователь с полным доступом, Режим редактирования (вставка/изменение контента)
  + Пользовательская активность (просмотр, оставляемые комментарии)

**Механизм комментирования:**

python

*# content-service/comments.py*

@app.post("/concepts/{concept\_id}/comments")

async def add\_comment(concept\_id: int, comment: str,

user=Depends(get\_current\_user),

db=Depends(get\_db)):

await db.execute(

"INSERT INTO comments (concept\_id, user\_id, text) VALUES ($1, $2, $3)",

concept\_id, user["userId"], comment

)

**Поиск внутри данных (использование Elasticsearch):**

javascript

*// search-service/index.js*

app.get('/search', async (req, res) => {

const { query } = req.query;

const results = await elasticsearch.search({

index: 'concepts',

body: {

query: {

multi\_match: {

query,

fields: ['title^3', 'content', 'category']

}

}

}

});

res.json(results.hits.hits);

});

**Инициализация системы**

Убедитесь, что все необходимые зависимости установлены:

bash

docker-compose build

1. Введите в эксплуатацию сервисы:

bash

docker-compose up -d

Для начала работы с базами данных используйте SQL-скрипты, расположенные в ./init.sql.

Вы можете получить доступ к системе по следующему адресу: http://localhost.

# 6.Анализ электронного справочника "Сети и телекоммуникации" с микросервисной архитектурой

## 6.1. Анализ архитектуры

**Положительные аспекты микросервисного подхода:**

* Каждый сервис (Auth, Content, Search) обладает возможностью независимой масштабируемости.
* Разнообразие технологий: платформа допускает использование различных инструментов, например Node.js для аутентификации и Python для работы с контентом.
* Системная отказоустойчивость означает, что если один сервис выходит из строя, это не приведет к остановке всей системы.
* Развертывание по отдельности: обновления могут быть выпущены для отдельных сервисов.

**Отрицательные стороны:**

* Управление сопряжено с сложностями: для его реализации необходимо использовать API Gateway и оркестрацию (Kubernetes).
* Коммуникационные издержки: межсервисная связь по сети.
* Для обеспечения согласованности данных необходимо внедрить механизмы Saga или Event Sourcing.

**Улучшения:**

* Внедрить кэширование Redis для ускорения доступа к часто используемым данным.
* Внедрить механизм Circuit Breaker в систему межсервисных запросов.
* Внедрить мониторинг (Prometheus + Grafana)

## 6.2. Изучение кода и аспектов безопасности

**Сильные стороны:**

javascript

*// Хорошие практики в Auth Service:*

const hashedPassword = await bcrypt.hash(password, 10); *// Надежное хеширование*

const token = jwt.sign(..., { expiresIn: '1h' }); *// Временные токены*

**Проблемные места:**

**Уязвимости безопасности:**

* + Жестко закодированные секреты (JWT\_SECRET)
  + Нет валидации входных данных (например, длины пароля)
  + Отсутствие rate-limiting на /login

**Отсутствие:**

* + Двухфакторной аутентификации
  + Механизма сброса пароля
  + Логирования подозрительных действий

**Рекомендации:**

python

*# Добавить в FastAPI (Content Service):*

from pydantic import constr

class UserCreate(BaseModel):

password: constr(min\_length=8, regex=r'^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])')

## 6.3. Производительность и данные

**База данных:**

* PostgreSQL хорошо подходит для структурированных данных
* Elasticsearch обеспечивает быстрый полнотекстовый поиск
* Redis снижает нагрузку на БД (кэш сессий, популярного контента)

**Проблемы:**

* Нет индексов в примере SQL-схемы

sql

*-- Добавить в init.sql:*

CREATE INDEX idx\_concepts\_category ON concepts(category);

CREATE INDEX idx\_concepts\_title ON concepts USING gin(to\_tsvector('english', title));

**Оптимизации:**

* Реализовать пагинацию для больших результатов
* Добавить материализованные представления для сложных запросов

## 6.4. Фронтенд-анализ

**Плюсы React-реализации:**

* Компонентный подход
* Использование хуков (useState, useEffect)
* JWT-аутентификация через заголовки

**Минусы:**

* Нет обработки ошибок API
* Отсутствие loading-состояний
* Прямое использование localStorage (уязвимость к XSS)

**Улучшения:**

jsx

*// Добавить в ConceptViewer:*

const [error, setError] = useState(null);

const [isLoading, setIsLoading] = useState(false);

useEffect(() => {

const fetchData = async () => {

try {

setIsLoading(true);

const response = await axios.get(...);

setConcept(response.data);

} catch (err) {

setError(err.message);

} finally {

setIsLoading(false);

}

};

fetchData();

}, [conceptId]);

## 6.5. Развертывание и DevOps

**Проблемы docker-compose.yml:**

* Нет healthcheck для сервисов
* Отсутствуют ресурсные ограничения
* Нет конфигурации для продакшн-среды

**Рекомендации:**

yaml

*# Добавить в сервисы:*

healthcheck:

test: ["CMD-SHELL", "curl -f http://localhost:3000/health || exit 1"]

interval: 30s

timeout: 10s

retries: 3

deploy:

resources:

limits:

cpus: '0.5'

memory: 512M

## 6.6. Тестирование

**Недостатки:**

* Нет юнит-тестов
* Отсутствуют интеграционные тесты
* Нет end-to-end тестирования

**Пример теста для Auth Service:**

javascript

*// auth-service/test/auth.test.js*

const request = require('supertest');

const app = require('../index');

describe('Auth API', () => {

it('should register new user', async () => {

const res = await request(app)

.post('/register')

.send({

username: 'test',

email: 'test@example.com',

password: 'Secure123!'

});

expect(res.statusCode).toEqual(201);

expect(res.body).toHaveProperty('userId');

});

});

## 6.7. Рекомендации по развитию

**Безопасность:**

* + Внедрить OAuth 2.0 (через Keycloak)
  + Добавить аудит изменений (кто и когда менял контент)
  + Реализовать регулярную ротацию секретов

**Производительность:**

* + Добавить CDN для статического контента
  + Внедрить GraphQL для гибких запросов
  + Оптимизировать Docker-образы (multi-stage build)

**Функциональность:**

* + Система версионирования контента
  + Offline-режим с Service Workers
  + Генерация PDF-документов по запросу

6.8.Заключение:

Текущая реализация демонстрирует хорошую основу, но требует доработок для промышленной эксплуатации. Оценка: 7.5/10 (перспективный проект с потенциалом для масштабирования).

# 7. Заключение

Разработанный и реализованный электронный справочник "Сети и телекоммуникации" представляет собой современное и доступное веб-приложение, реализованное с использованием современной микросервисной архитектуры. В ходе работы были достигнуты следующие ключевые результаты:

Архитектурные решения:

* + Реализована модульная система из независимых сервисов
  + Обеспечена горизонтальная масштабируемость технологических компонентов
  + Достигнута высокая отказоустойчивость за счет изоляции сервисов

Функциональные возможности:

* + Создана комплексная система идентификации, аутентификации и авторизации
  + Разработана структурированная база знаний по сетевых технологиям
  + Реализован достаточно эффективный поиск по содержимому

Технологические преимущества:

* + Применение современных технологических стеков (Node.js, FastAPI, React, Condition и др.) эффективное и разнообразное в работе (+-, +++)
  + Использование специализированных СУБД для разных задач
  + Автоматизация развертывания и обновления файлового обеспечения по Python через Docker

Перспективы развития проекта включают:

* Внедрение системы практических рекомендаций на основе ML
* Разработку мобильного приложения с offline-режимом
* Интеграцию с другими платформами онлайн-обучения (LMS), АМПП
* Расширение функционала интерактивных лабораторных работ

# 8. Список литературы

1. Таненбаум Э., Уэзеролл Д. Компьютерные сети. 5-е изд. - СПб.: Питер, 2019. - 992 с.
2. Куроуз Дж., Росс К. Компьютерные сети: нисходящий подход. - М.: Эксмо, 2021. - 848 с.
3. Richardson К. Микросервисные шаблоны. - М.: ДМК Пресс, 2020. - 520 с.
4. Newman S. Building Microservices. 2nd ed. - O'Reilly Media, 2021. - 614 p.
5. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. - UC Irvine, 2000. - 180 p.
6. Документация Docker: <https://docs.docker.com/>
7. Официальная документация React: <https://reactjs.org/docs/>
8. ГОСТ Р 7.0.97-2016 "Система стандартов по информации, библиотечному и издательскому делу"
9. RFC 6749 - The OAuth 2.0 Authorization Framework
10. IEEE 802.3-2018 - Standard for Ethernet