

федеральное государственное бюджетное образовательное учреждение
высшего образования
«ЮЖНО-РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ (НПИ) имени М.И.Платова»

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.02 Информационные системы и технологии

УТВЕРЖДАЮ:
Заведующий кафедрой ИИСТ
Н.И. Горбатенко

(Подпись) « » Г.

Новочеркасск, 202__ г.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**
федеральное государственное бюджетное образовательное учреждение высшего образования
«ЮЖНО-РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
(НПИ) имени М.И. Платова»

ФАКУЛЬТЕТ Информационных технологий и управления
КАФЕДРА «Информационные и измерительные системы и технологии»
НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.02 – Информационные системы и технологии

УТВЕРЖДАЮ:
Заведующий кафедрой ИИСТ

(Подпись) Н.И. Горбатенко
« ____ » _____ Г.

ЗАДАНИЕ
на бакалаврскую работу

Студенту **Коновалову Никите Александровичу**
(Фамилия, имя, отчество)

1. Тема бакалаврской работы:
Подсистема хранения и обработки данных цифрового двойника ЮРГПУ
(НПИ) им. М.И. Платова

Тема бакалаврской работы утверждена приказом ректора _____

2. Консультанты бакалаврской работы:
Наименование раздела, должность, ученая степень, ученое звание, ФИО

3. Исходные данные к бакалаврской работе
На основании предполагаемого функционала информационной системы цифрового
двойника, создать Web-сайт со следующими редактируемыми таблицами: личности,
местоположения, виды активности, пользователи, нарушения устава, СКУД

4. Содержание пояснительной записки к бакалаврской работе

4.1 Предпроектный анализ цифрового кампуса

4.2 Системное проектирование подсистемы хранения и обработки данных

4.3 Информационное обеспечение сайта

4.4 Программная реализация сайта подсистемы хранения и обработки данных

5. Перечень графического материала: схема организационной структуры; схемы потоков
данных, документооборота; схема потоков работ (бизнес процессов, функций); схема
деления системы; схемы функциональной структуры; схема структурная комплекса
технических средств; схема структуры информационной базы; схема автоматизации; схема
алгоритма вычислительного, информационного процессов.

6. Срок сдачи студентом законченной бакалаврской работы _____ Г.
Дата выдачи задания на бакалаврскую работу _____ Г.

Руководитель _____
Хорошко М.Б.
(Фамилия, имя, отчество)

Задание принял к исполнению « ____ » _____ Г.
(Подпись)

ABSTRACT

Gradually, the Internet is transforming from a huge toy for developed intellectual personalities into a full-fledged source of diverse and at the same time useful information for any segment of the population of the planet Earth, so it has become the most effective means of promoting and exchanging information and is one of the significant elements of modern civilization. I think that's why one of the most popular topics today is the creation and maintenance of websites.

They, that is, sites, allow you to process, transmit, transform, store, sell various types of information. And for all this, you do not need to move away from the computer screen. The most important thing is that the sites allow you to present this information in a colorful and distinct form, provide animation, videos, graphics, sound, links. I am sure that with proper planning of the work performed, it is possible to demonstrate this correctly submitted information to thousands of other Internet users.

The web-site of the digital double of the YURSPU (NPI) through the Internet provides users with access to operational information that occurs at the university. Access to information requires user registration, as this information is not publicly available.

This thesis presents the Web site of the digital twin storage subsystem of the M.I. Platov YURSPU (NPI), which interacts with the Yandex Database connected to it.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	7
1 ПРЕДПРОЕКТНЫЙ АНАЛИЗ ОБЪЕКТА АВТОМАТИЗАЦИИ	8
1.1 Описание предметной области подсистемы хранения и обработки данных ЦК ЮРГПУ(НПИ).....	8
1.2 Функции организационная структура подсистемы хранения и обработки данных ЮРГПУ(НПИ)	9
1.3 Описание потоков данных и бизнес-процессов подсистемы хранения и обработки данных ЦК ЮРГПУ(НПИ).....	12
1.4 Обзор и анализ существующих проектных решений, выявление их достоинств и недостатков	17
1.5 Обоснование необходимости разработки подсистемы хранения и обработки данных системы ЦК ЮРГПУ(НПИ)	18
2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПОДСИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ ЦИФРОВОГО ДВОЙНИКА ЮРГПУ(НПИ)	22
2.1 Разработка концепции и архитектуры построения и платформы реализации информационной системы.....	22
2.2 Структура информационной системы, включающей подсистему хранения и обработки данных ЦД ЮРГПУ(НПИ), состав функциональных и обеспечивающих подсистем.	23
2.3 Техническое обеспечение информационной системы.....	26
3 ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ПОДСИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ ЦИФРОВОГО ДВОЙНИКА ЮРГПУ(НПИ)	30
3.1 Концептуальное проектирование БД.....	30
3.2 Конструирование логической модели	32
3.3 Описание физической реализации БД	34
4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПОДСИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ ЦИФРОВОГО ДВОЙНИКА ЮРГПУ(НПИ)	39
4.1 Описание структуры программного обеспечения.....	39
4.2 Алгоритмизация типовых информационных запросов.....	44
4.3 Описание пользовательского интерфейса.....	48
ЗАКЛЮЧЕНИЕ	52

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	53
ПРИЛОЖЕНИЕ А – Фрагменты листинга программы.....	54
ПРИЛОЖЕНИЕ Б – Презентация.....	91

В тексте пояснительной записки дипломной работы введены следующие специальные сокращения:

Обозначение	Описание
ИС	Информационная система
ПО	Программное обеспечение
БД	База данных
СУБД	Система управление базами данных
Подсистема	Подсистема хранения и обработки данных в цифровом двойнике ЮРГПУ(НПИ) им. М.И. Платова
Система	Информационная система цифрового двойника ЮРГПУ(НПИ) им. М.И. Платова
ЦД	Цифровой двойник
SQL	Язык структурированных запросов
API	Application programming interface
IDEF0	Стандарт описания деятельности предприятия
UML	Унифицированный язык моделирования

ВВЕДЕНИЕ

Постепенно, интернет трансформируется из громадной игрушки для развитых интеллектуальных личностей в полноценный источник разнообразной и одновременно полезной информации для любого слоя населения планеты Земля, поэтому он стал наиболее эффективным средством продвижения и обмена информацией и является одним из значительных элементов современной цивилизации. Я считаю, что поэтому одной из популярных тем на сегодняшний день является создание и поддержание сайтов.

Они, то есть сайты, позволяют обрабатывать, передавать, преобразовывать, хранить, продавать различные типы информации. И для всего этого не нужно отходить от экрана компьютера. Наиболее важным является то, что сайты позволяют подать эту информацию в красочном и отчетливом виде, снабдить анимацией, видеороликами, графикой, звуком, ссылками. Я уверен, что при правильном планировании выполняемой работы, возможно продемонстрировать эту правильно поданную информацию тысячам других пользователей интернета.

Web-сайт цифрового двойника ЮРГПУ (НПИ) будет создан для того, чтобы через сеть Интернет обеспечивает доступ пользователям к оперативной информации, которая происходит в университете. Доступ к информации требует регистрации пользователя, так как данная информация является не общедоступной.

В настоящей дипломной работе представлен Web-сайт подсистемы хранения цифрового двойника ЮРГПУ (НПИ) им. М.И. Платова, который взаимодействует с подключенной к нему базой данных Yandex Database.

1 ПРЕДПРОЕКТНЫЙ АНАЛИЗ ОБЪЕКТА АВТОМАТИЗАЦИИ

1.1 Описание предметной области подсистемы хранения и обработки данных ЦК ЮРГПУ(НПИ)

В данной дипломной работе будет представлен проект подсистемы хранения цифрового двойника ЮРГПУ(НПИ) им. М.И. Платова, основное назначение которой – хранение данных о всех существующих информационных ресурсах, относящихся к ЮРГПУ(НПИ) в едином информационном пространстве. Подсистема хранения информации для цифрового двойника ЮРГПУ(НПИ) разработана как для сотрудников, так и для студентов и прочего персонала, который непосредственно связан с ЮРГПУ(НПИ), но основную функциональную часть подсистемы может получить только персонал ВУЗа, которому будет необходимо работать с данными. Деятельность системы осуществляется следующим образом: каждый участник проходит регистрацию на сайте системы цифрового двойника, после чего ему предоставляется доступ к подсистеме хранения информации через функциональную админ-панель.

В настоящее время к системе предъявляются следующие требования:

- Защита информации. Информация должна быть защищена на разных уровнях с использованием существующих стандартов и протоколов включая систему разграничения доступа и сквозным шифрованием данных;
- Масштабируемость. Система должна иметь возможность горизонтального и вертикального масштабирования при увеличении данных и нагрузки системы;
- Работа с большими данными в реальном времени.

Целью создания данной дипломной работы является проектирование информационной системы для информационного сообщества с современными потребностями: для студентов, преподавателей и административных работников вуза.

Для решения поставленной задачи необходимо выполнить следующие шаги:

- Разработать структуру информационной системы;
- Разработать модели информационной системы;
- Разработать программное обеспечение системы.

1.2 Функции организационная структура подсистемы хранения и обработки данных ЮРГПУ(НПИ)

Данный отдел включает в себя ректора ЮРГПУ(НПИ), начальника отдела управления ЦД, главного бухгалтера, администратора сервера, а также отдел информационных систем, включающий: разработчика, аналитика, тестировщика.

Схема организационной структуры цифрового двойника ЮРГПУ(НПИ) приведена на рисунке 1.1



Рисунок 1.1 - Схема организационной структуры

Основными задачами главного бухгалтера являются ведение расчетов с сотрудниками, составление отчетности и осуществление расчетов с финансовыми органами

Основными задачами отдела информационных систем являются разработка программного обеспечения, проектированием технического

решения, модернизация информационной системы, обслуживание системы и тестирование системы.

Основными задачами администратора сервера являются обслуживание технического оборудования, работа с базой данных и поддержка работоспособности базы данных.

Схема функциональной структуры информагентства изображена на рисунке 1.2

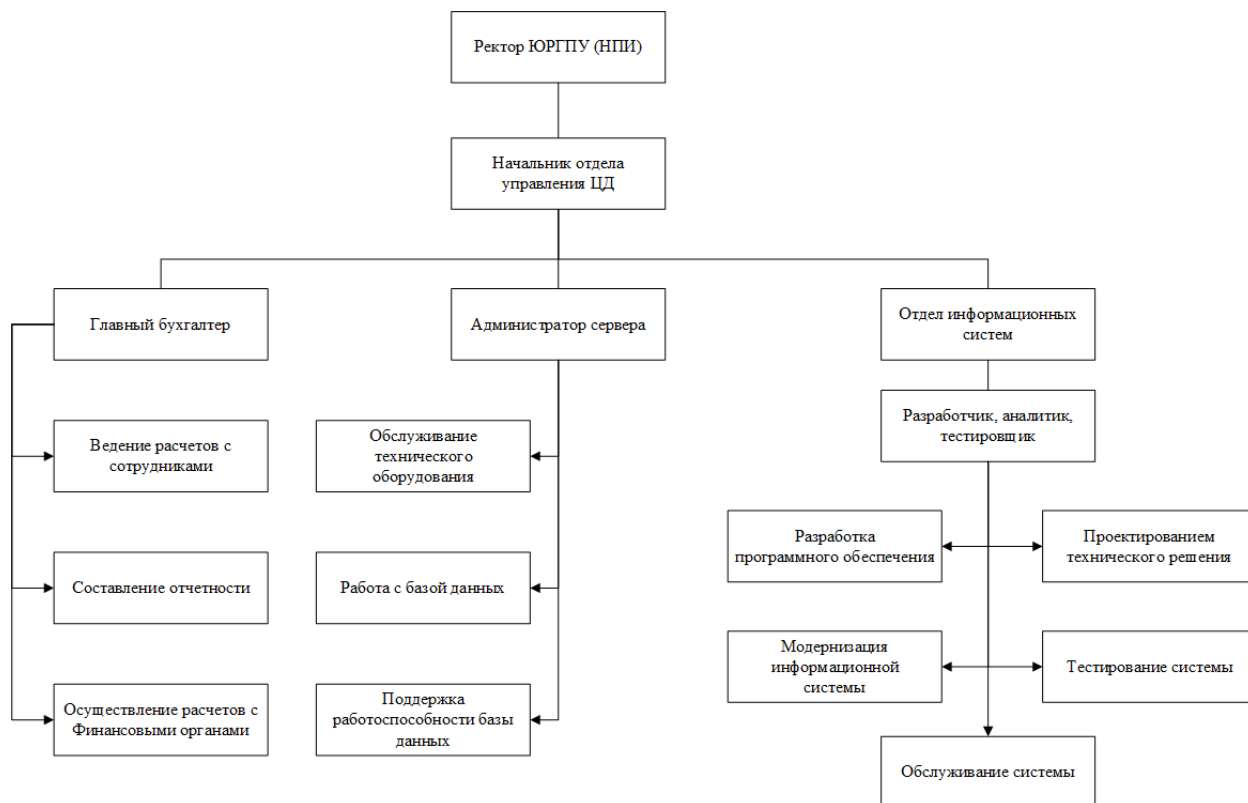


Рисунок 1.2 – Схема функциональной структуры компании

Матричная модель функциональной ответственности производит закрепление ответственности для структурных звеньев (а также отдельных специалистов) за выполнение отдельных определенных функций при реализации деятельности организации ЮРГПУ(НПИ) в целом. Матричная модель функциональной ответственности приведена таблице 1.1.

Таблица 1.1 – Матричная модель функциональной ответственности

Функции	Главный бухгалтер	Администратор сервера	Отдел информационных систем
Ведение расчетов с сотрудниками	+	—	—
Составление отчетности	+	—	—
Осуществление расчетов с Финансовыми органами	+	—	—
Обслуживание технического оборудования	—	+	—
Работа с базой данных	—	+	—
Поддержка работоспособности базы данных	—	+	—
Разработка программного обеспечения	—	—	+
Модернизация информационной системы	—	—	+
Обслуживание системы	—	—	+

Проектированием технического решения	—	—	+
Тестирование системы	—	—	+

1.3 Описание потоков данных и бизнес-процессов подсистемы хранения и обработки данных ЦК ЮГРПУ(НПИ)

Для описания бизнес-процессов цифрового двойника ЮРГПУ (НПИ) были построены функциональные диаграммы, основанные на технологии моделирования IDEF0. Контекстная диаграмма главной функции цифрового двойника ЮРГПУ (НПИ) изображена на рисунке 1.3

Диаграмма декомпозиции главной функции работы цифрового двойника ЮРГПУ (НПИ) изображена на рисунке 1.4.

Диаграмма декомпозиции работы СКУД изображена на рисунке 1.5.

Диаграмма потоков данных (в нотации DFD) функции работы СКУД изображена на рисунке 1.6.



Рисунок 1.3 – Экранная форма контекстной диаграммы главной функции цифрового двойника ЮРГПУ (НПИ)

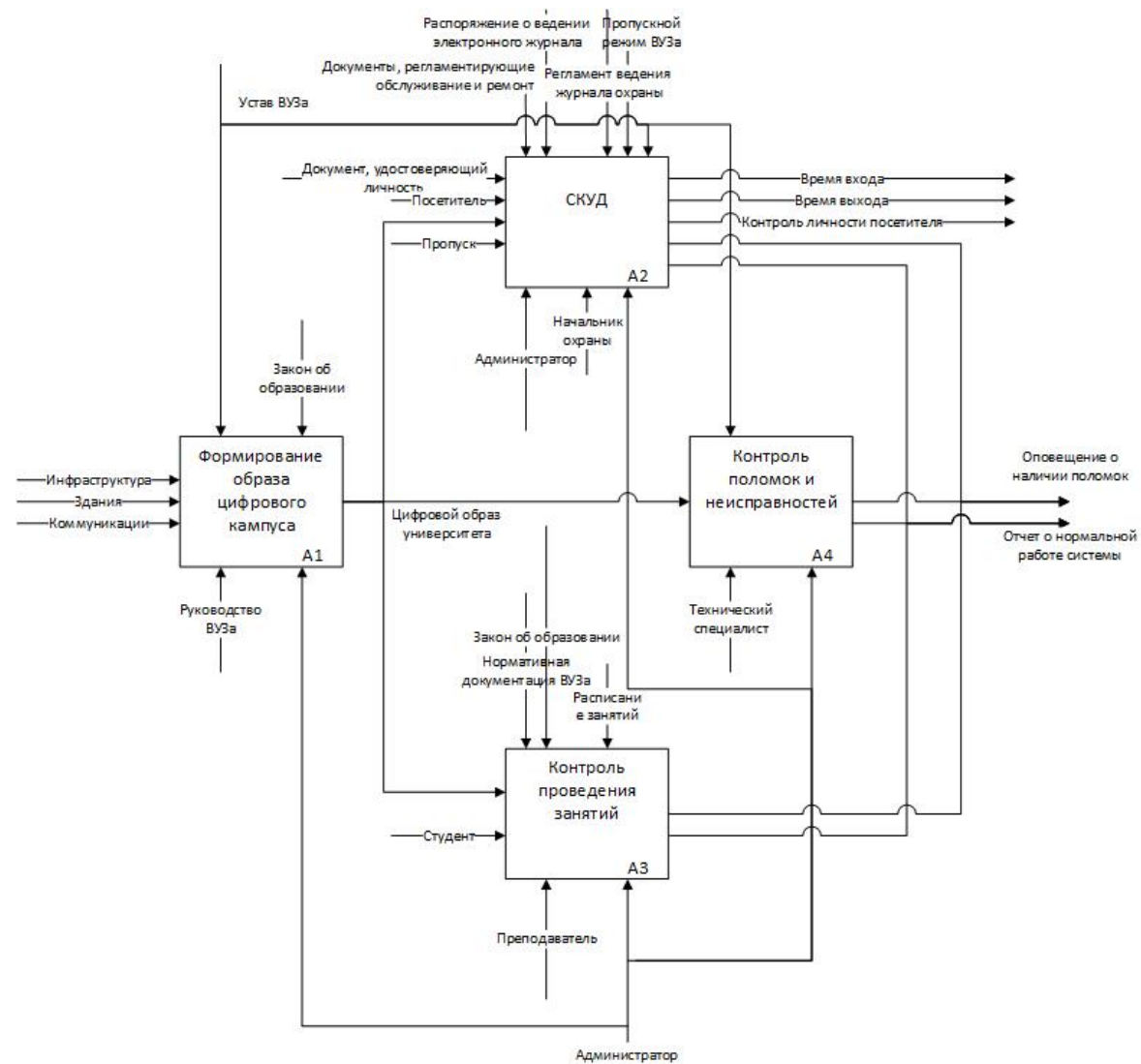


Рисунок 1.4 – Экранная форма диаграммы декомпозиции главной функции цифрового двойника ЮРГПУ (НПИ)

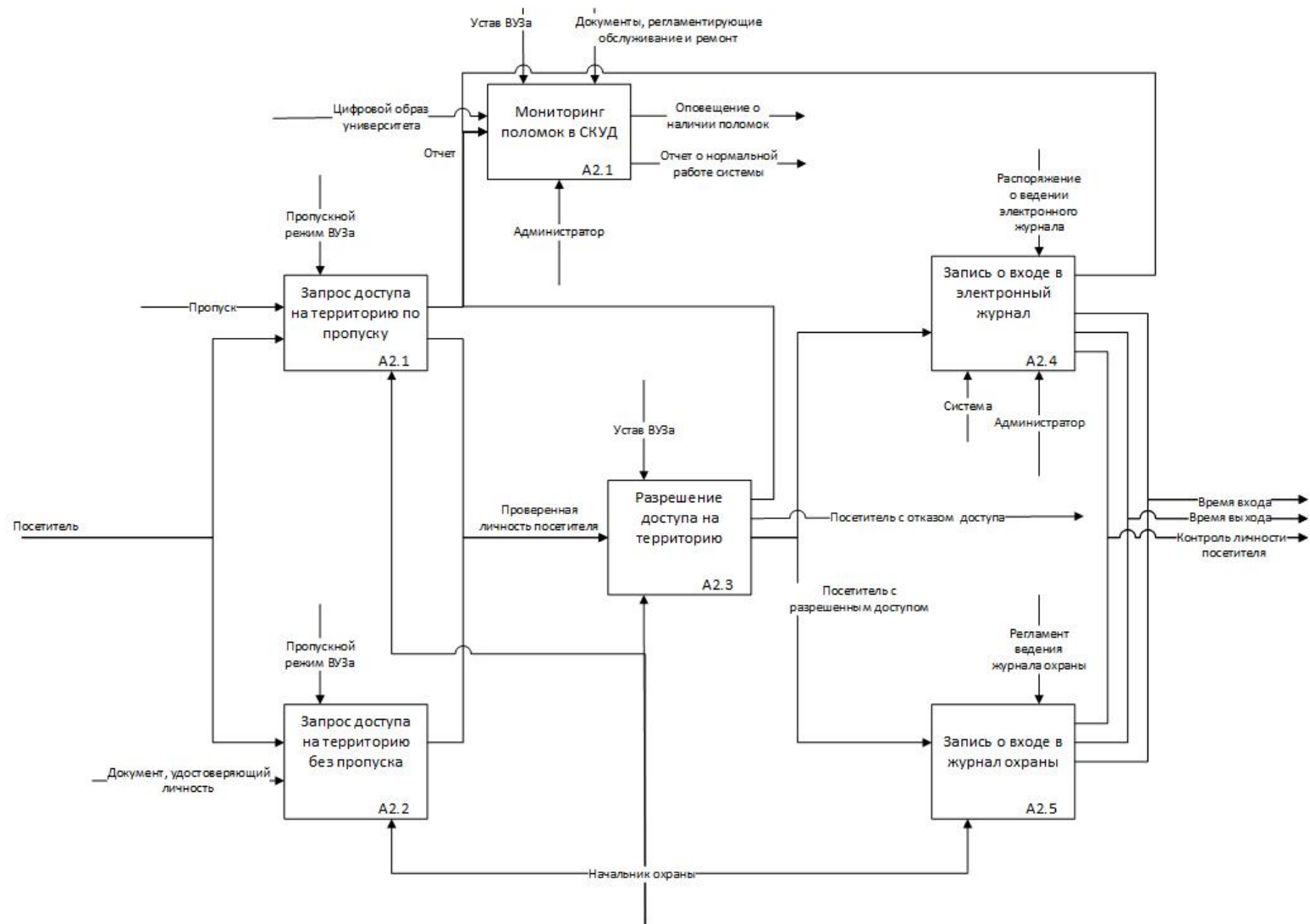


Рисунок 1.5 - Экранная форма диаграммы декомпозиции работы СКУД

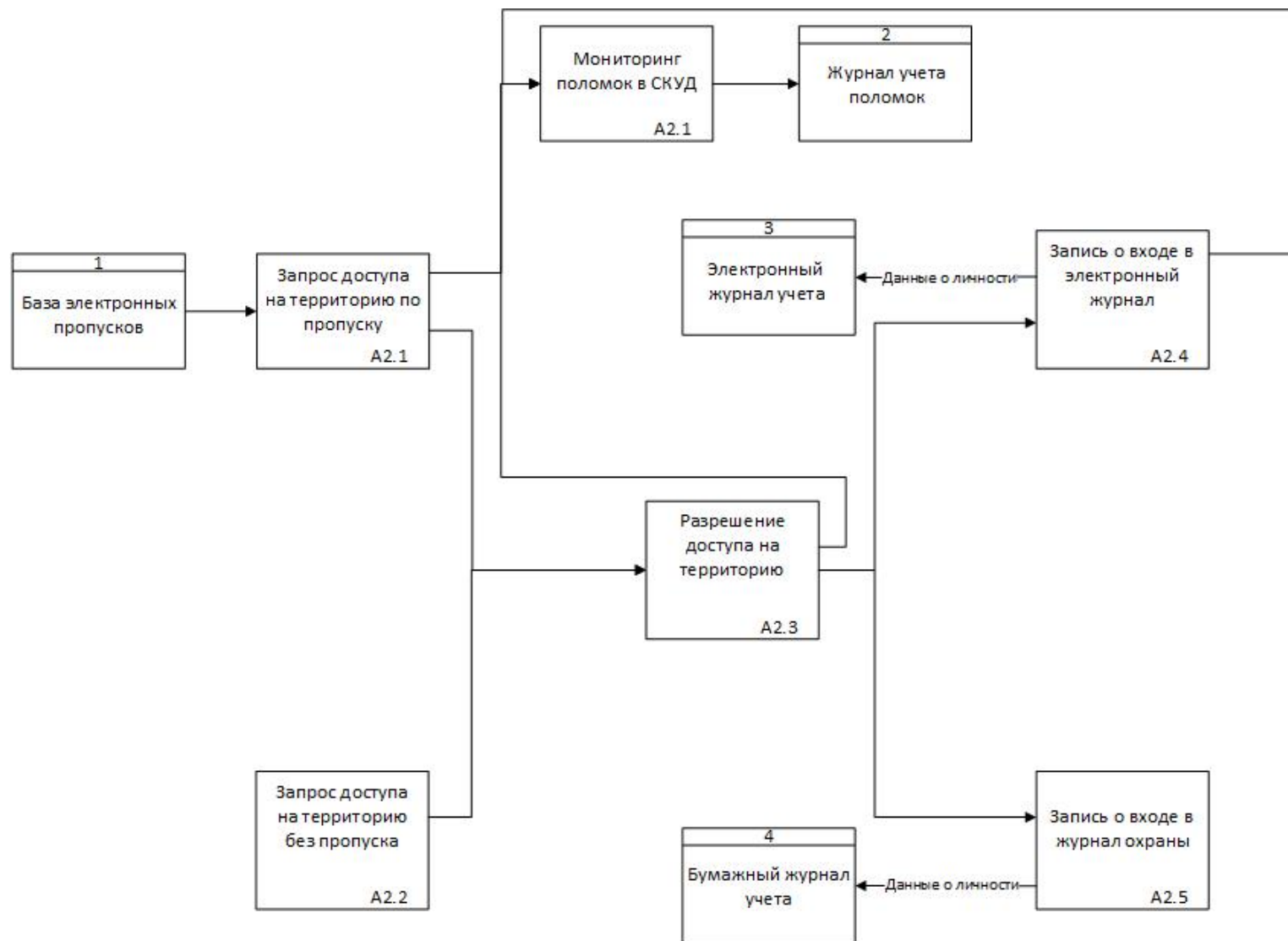


Рисунок 1.6 - Экранная форма диаграмма потоков данных (в нотации DFD) функции работы СКУД

1.4 Обзор и анализ существующих проектных решений, выявление их достоинств и недостатков

В данном пункте описаны следующие аналоги информационной системы цифрового двойника ЮРГПУ (НПИ) им. М.И. Платова:

1.4.1 Цифровой двойник НГГТИ – обеспечивает информационную открытость НГГТИ в соответствии с требованиями действующего законодательства Российской Федерации в сфере образования.

Достоинства:

- Удобный интерфейс;
- Функция поиска по сайту;
- Реализован «Ночной режим»;
- Кроссплатформенная адаптация
- Нет рекламы
- Доступ в электронную библиотеку

Недостатки:

- Не реализована «версия для слабовидящих»
- Нет формы обратной связи

1.4.2 ЦК ТТИ ЮФУ – является подобием социальной сети с образовательной направленностью, необходимость которой важна для организации эффективного образовательного процесса.

Достоинства:

- Кроссплатформенная адаптация
- Удобный интерфейс
- Реализована «версия для слабовидящих»
- Удобная форма обратной связи

- Доступ в электронную библиотеку

Недостатки:

- Реклама;
- Не реализован «Ночной режим»;
- Не реализована функция поиска по сайту

В таблице 1.2 представлены сравнения представленных решений.

Таблица 1.2 – Сравнения представленных решений

Параметр сравнения	ЦК НГГТИ	ЦК ТТИ ЮФУ
Удобство интерфейса	+	+
Функция поиска по сайту	+	–
Наличие «версии для слабовидящих»	–	+
Наличие «ночного режима»	+	–
Кроссплатформенная адаптация	+	+
Форма обратной связи	–	+
Доступ в библиотеку	+	+
Отсутствие рекламы	+	–

1.5 Обоснование необходимости разработки подсистемы хранения и обработки данных системы ЦК ЮРГПУ(НПИ)

Современные университеты [1] – это маленькие города. В них есть библиотеки, концертные залы, спортивные залы, бассейны, магазины, больницы, гостиницы, общежития, офисы, рестораны, столовые, парковки, аудитории, расчетные центры, платежные терминалы. В них есть жители – студенты, преподаватели и сотрудники, есть гости – абитуриенты, родители, работодатели, партнеры. Чтобы все это функционировало, чтобы для каждого

жителя и гостя университета был доступ к ресурсам, службам и сервисам в соответствии с их ролью, в университете необходимы:

- техническая инфраструктура - вычислительная сеть, включая оборудование беспроводного доступа, компьютерное оборудование, устройства телекоммуникации и связи, презентационное и видео оборудование, мобильные устройства для доступа к цифровым ресурсам, системы контроля и управления доступом к ресурсам, системы сигнализации и видеонаблюдения
- информационная инфраструктура, реализованная в виде цифровых ресурсов, приложений и сервисов корпоративной информационной среды
- единый атрибут для доступа к ресурсам университета – персональные идентификационные карты (типа proximity или smart)

Концепция электронного кампуса позволяет полнее раскрыть потенциал университета и оптимизировать имеющиеся в университете ресурсы. Реализованная в настоящее время в университете модель цифрового кампуса обеспечивает студентам:

- доступ на территорию и в общежития по идентификационной пластиковой карте;
- доступ в Интернет и к цифровым ресурсам университета из любой точки кампуса через проводную или беспроводную сеть;
- доступ в библиотеку и к множественному образовательному контенту в форме текста, графики, видео и аудиоматериалов, презентаций к занятиям, видеолекций, тестов и т.п.
- доступ к сервису видеоматериалов с использованием технологии потокового вещания;
- доступ к занятиям и консультациям из удаленных точек – через видеоконференцсвязь и вебинары, что повышает мобильность студентов, обеспечивает общение с преподавателями и студентами, участниками партнерских программ университета;
- доступ к спортивным, медицинским услугам;
- доступ к сервисам портала университета – индивидуальный план обучения студента, расписание занятий, успеваемость, выполнение курсовых и дипломных работ, ведение студенческих проектов, контроль платежей и т.д.

Для преподавателей и сотрудников цифровой кампус обеспечивает:

- доступ на территорию и в помещения по идентификационной пластиковой карте;

- доступ в Интернет и к цифровым ресурсам университета из любой точки кампуса через проводную или беспроводную сеть
- доступ в библиотеку
- возможность публиковать образовательный контент (тексты, презентации, видео);
- возможность проводить занятия для удаленных студентов через видеоконференцсвязь, вебинары и возможность вести занятия, находясь удаленно от кампуса
- доступ на автомобильную парковку университета
- доступ к спортивным, медицинским услугам, а также к услугам службы питания
- возможность реализовывать свои бизнес-процессы по разным направлениям деятельности с помощью ИТ, добиваясь более качественных результатов, с меньшими затратами и с большей производительностью за счет
- автоматизации учета – студентов, сотрудников, аспирантов, выпускников, материальных ценностей, недвижимости, библиотечного фонда, образовательных ресурсов, научных проектов, посещаемости, ремонтов зданий и помещений, расходов, доходов, публикаций и много другого
- автоматизации расчетов – заработной платы, табелей, амортизации, нагрузки, рейтингов студентов, преподавателей, кафедры, стипендии, трафика, закупок, оплаты за обучение, Интернет, проживание, услуги спорта, медицины, службы питания и т.д.
- автоматизация процессов – формирование образовательных программ и учебных планов, графиков учебного процесса, индивидуальных траекторий обучения, расписания занятий, приказов, договоров, сайтов, проведение сессий, практик, экзаменов, подачи и обработки заявок, планирование и отчетность деятельности подразделений, управления доступом к ресурсам, планирования ремонтов, расходов, доходов и многое другое
- предоставления доступа к необходимой актуальной информации по всем направлениям деятельности университета

В задачах управления университетом электронный кампус через сервисы КИС обеспечивает:

- Управление оргструктурой и персоналом
- Применение методов принятия решений на основе данных корпоративной информационной среды

- Использование агрегированных хранилищ данных и методов анализа данных
- Использование методов и технологий анализа бизнес-процессов (Business Intelligence)
- Применение системы управления электронного документооборотом для всех процессов, ориентированных на документы
- Контроль исполнительской дисциплины
- Планирование и отчетность по направлениям деятельности

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПОДСИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ ЦИФРОВОГО ДВОЙНИКА ЮРГПУ(НПИ)

2.1 Разработка концепции и архитектуры построения и платформы реализации информационной системы

Разрабатываемая автоматизированная информационная система обработки данных имеет распределенную, клиент-серверную архитектуру, применяется в сфере предоставления информации. При разработке будет использованы каноническое проектирование и итерационная модель проектирования информационных систем. По типу вычислительных устройств и средств вычислительной техники, разрабатываемая система относится к информационным системам для автоматизации процессов, а по типу предложенного решения – типовое решение. На рисунке 2.1 изображена схема архитектуры построения и платформы реализации подсистемы хранения и обработки данных цифрового двойника ЮРГПУ (НПИ).

Серверная часть представляет собой облачное хранилище с YandexDB базой данных. А клиентская часть представляет собой веб-приложение – совокупность HTML разметки и Node.js кода. Для работы приложения и соединения с базой данных необходимы устройства, поддерживаемые подключение к сети передачи данных.

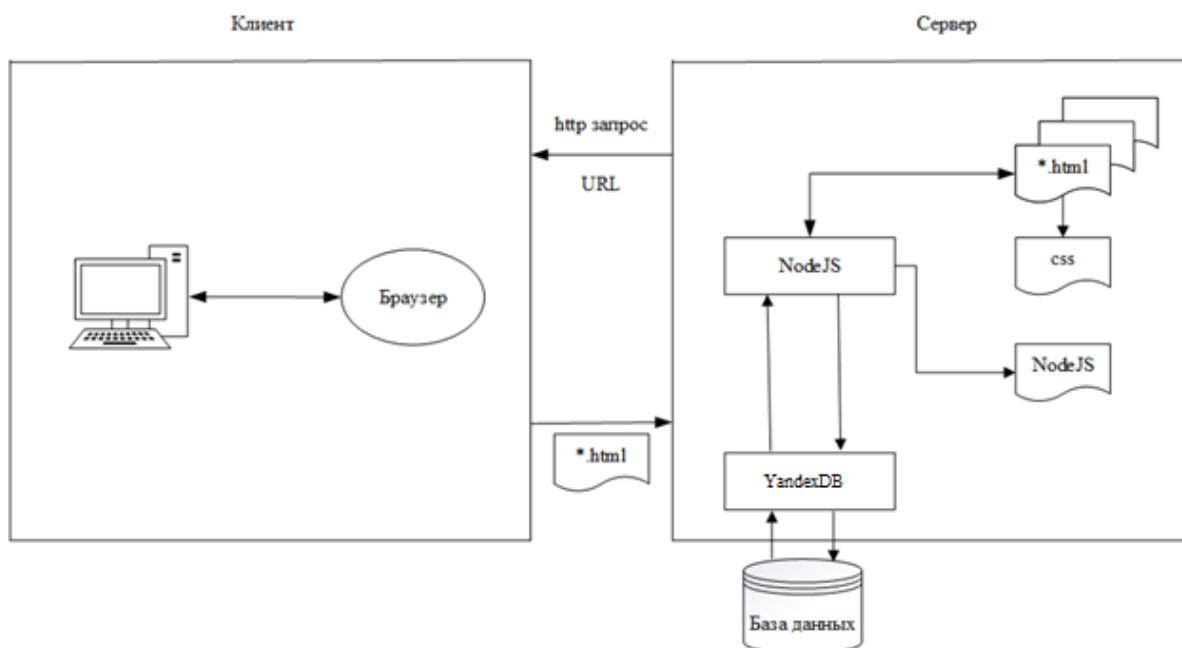


Рисунок 2.1 – схема архитектуры построения и платформы реализации подсистемы хранения и обработки данных цифрового двойника ЮРГПУ (НПИ)

2.2 Структура информационной системы, включающей подсистему хранения и обработки данных ЦД ЮРГПУ(НПИ), состав функциональных и обеспечивающих подсистем.

ИС включает следующие подсистемы:

- подсистема личного кабинета;
- подсистема поиска информации на сайте;
- подсистема отображения СКУД;
- подсистема регистрации и авторизации.

Подсистема личного кабинета предназначена для предоставления данных пользователя, а также его действий. В нее входят:

- админ панель
- ВЫХОД

Подсистема поиска информации на сайте предназначена для поиска искомого результата по выбранному способу на сайте. В нее входят:

- поиск на сайте
- просмотр информации

Подсистема отображения СКУД предназначена для добавления новостей и вывода их из базы данных. В нее входят:

- отображение данных о входе/выходе
- просмотр статистики

Подсистема регистрации и авторизации предназначена для регистрации и авторизации пользователя. В нее входят:

- регистрация
- авторизация

Структурная схема информационной системы показана на рисунке 2.2



Рисунок 2.2 – Структурная схема информационной системы цифрового двойника ЮРГПУ (НПИ)

2.3 Техническое обеспечение информационной системы

Минимальными системными требованиями для эксплуатации информационной системы цифрового двойника ЮРГПУ (НПИ) являются:

- Операционная система Windows 10
- Оперативная память: 2 Гб;
- Веб-браузер;
- Доступ в интернет.

Помимо этого, для обеспечения функционирования информационной системы в отделе информационных систем необходимо иметь следующее оборудование:

- Сервер базы данных: D-Link DVS-310-1/B1A
- Клиент: DEXP Mars E335
- Маршрутизатор: D-Link DIR-815/R1
- Витая пара: DEXP

На рисунке 2.3 показан вариант плана размещения вычислительной техники в помещениях компании.

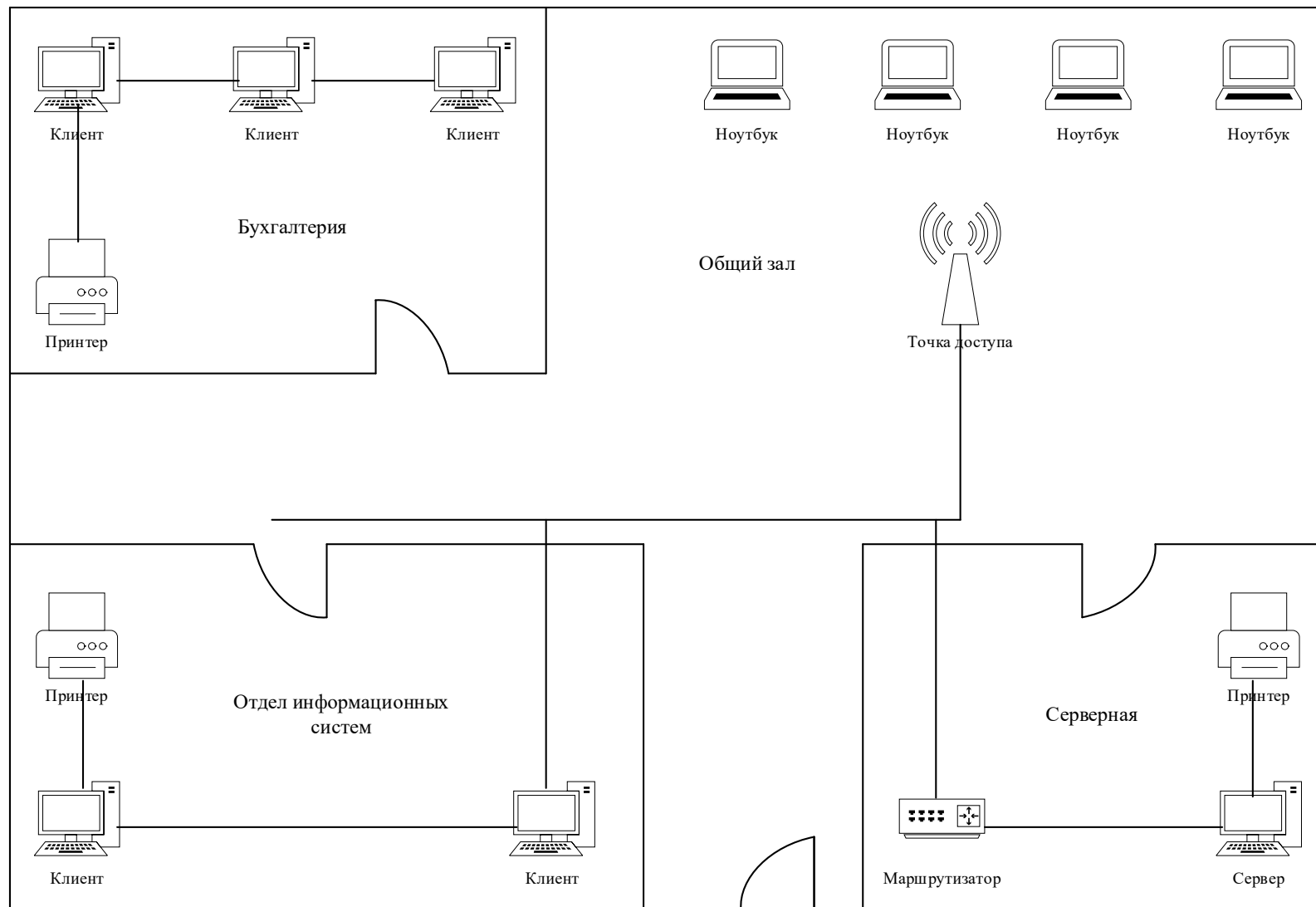


Рисунок 2.3 – Схема офиса отдела управления ЦД

В офисе расположены 3 кабинета и общий зал. Кабинет бухгалтерии, кабинет отдела информационных систем, сервер. В каждом кабинете стоят компьютеры, подключенные к сети интернет, а также принтер.

В серверной установлен маршрутизатор, связанный с интернетом с помощью витой пары. К маршрутизатору подключена беспроводная точка доступа Wi-Fi, к которой подключаются офисные компьютеры.

Сервер базы данных подключен к коммутатору, который, в свою очередь, подключен к маршрутизатору. Маршрутизатор подключен к интернету с помощью оптического волоконного кабеля. Пользователи подключаются либо к мобильному интернету, либо к точке доступа Wi-Fi, которая, посредством маршрутизатора, подключена к интернету.

На рисунке 2.4 изображена структурная схема сети передачи данных.

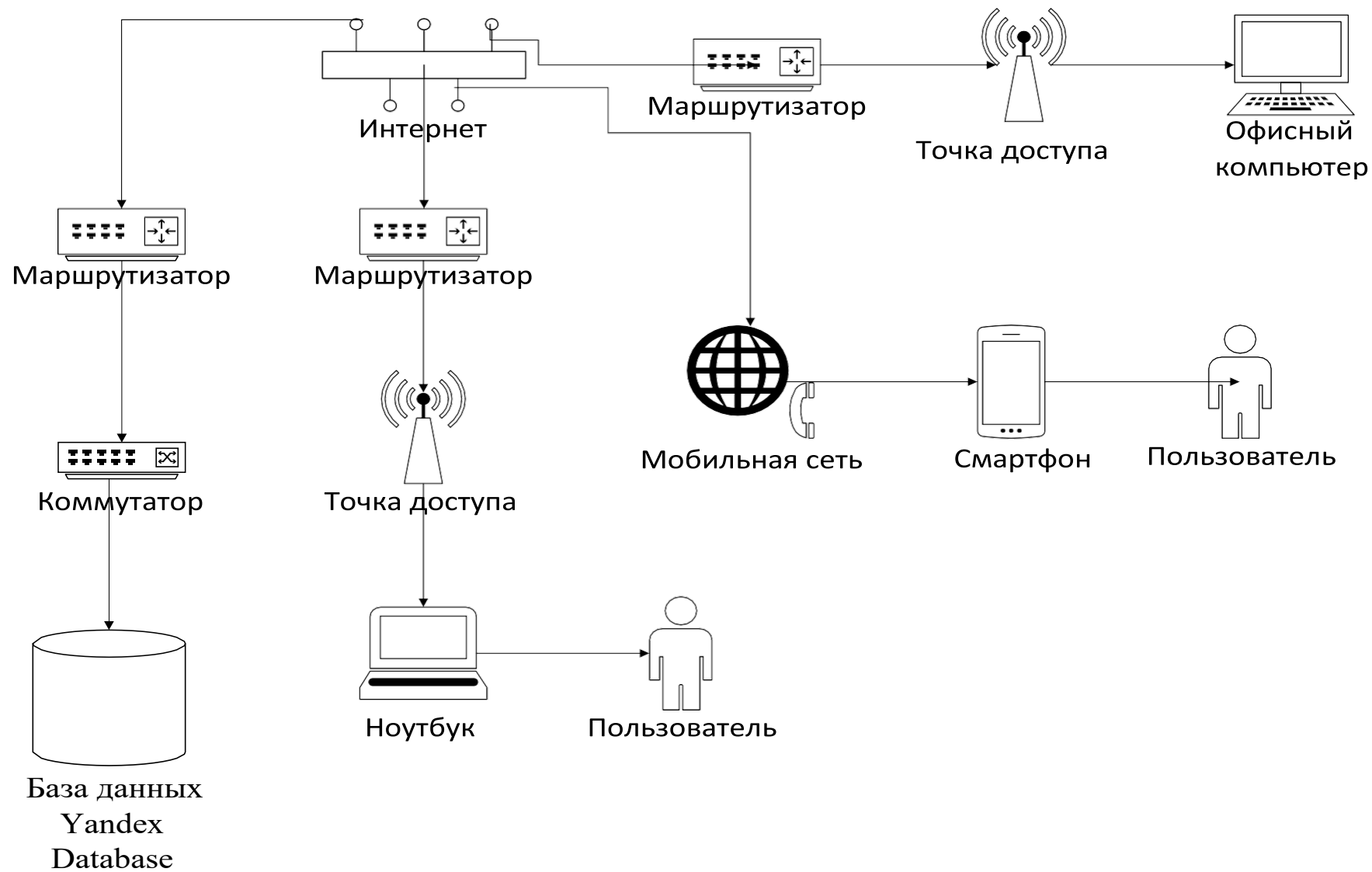


Рисунок 2.4 – Схема сети передачи данных

3 ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ПОДСИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ ЦИФРОВОГО ДВОЙНИКА ЮРГПУ(НПИ)

3.1 Концептуальное проектирование БД

Концептуальное проектирование технических систем – это основная и первичная стадия проектирования, необходимая для самого начала, на которой принимаются решения, определяющие последующий облик, и проводится исследование и согласование параметров созданных технических решений с возможной их организацией. Термин «концепция» применяется для описания принципа действия не только в технических системах, но и в научных, художественных и прочих видах деятельности. «Концепт» – содержание понятия, смысл. Таким образом, проектирование на концептуальном уровне – на уровне смысла или содержания понятия систем [2].

На раннюю стадию разработки падает основной и первостепенный объем задач концептуального проектирования технических систем (ТС): при выработке массива вариантов технических и оформительских решений, постановке задачи на проектирование.

Результатом концептуального проектирования является концептуальная модель базы данных, представленная на рисунке 3.1.

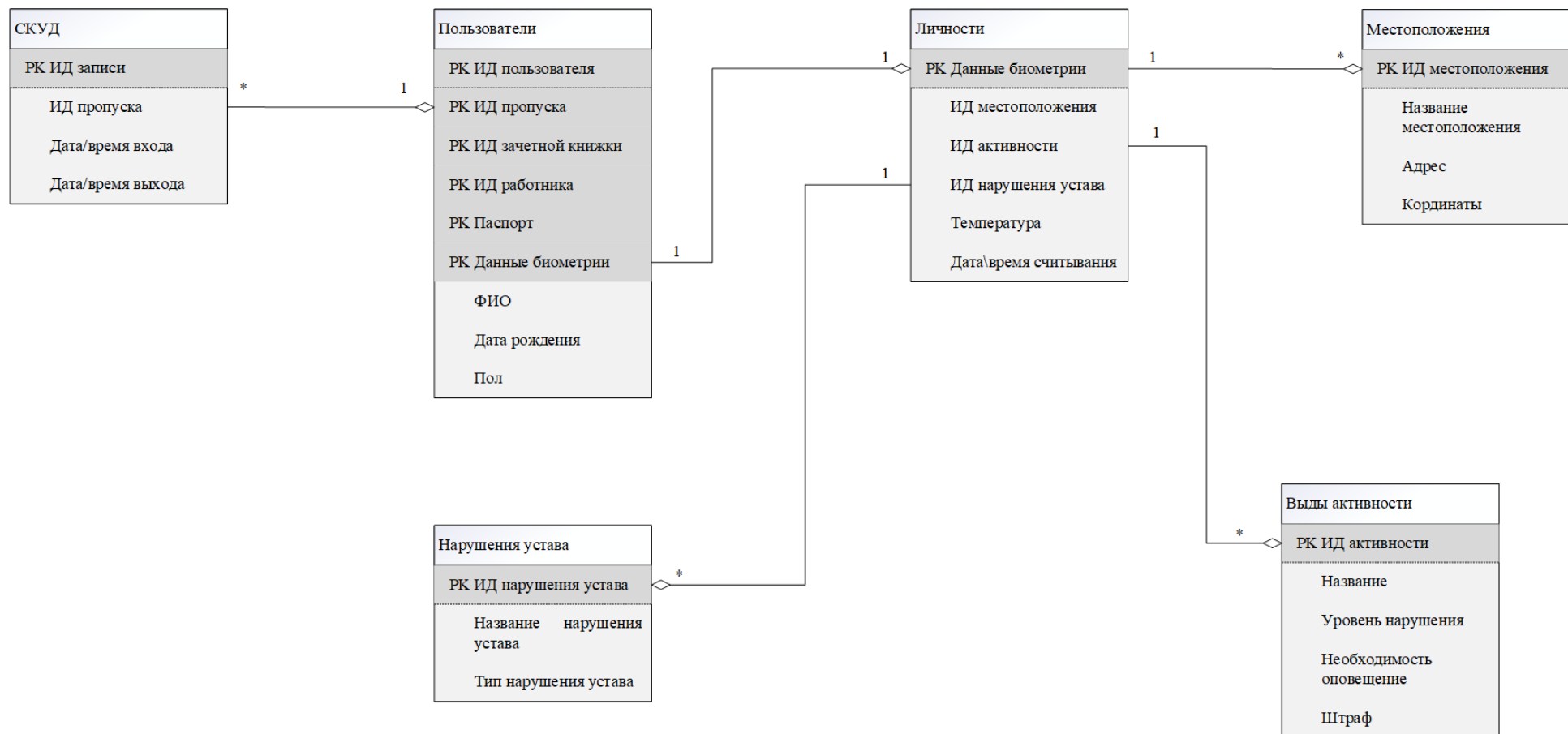


Рисунок 3.1 – Концептуальная модель базы данных информационной системы цифрового двойника ЮРГПУ (НПИ)

3.2 Конструирование логической модели

Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области.

Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой [2].

Сущности описывают объекты, являющиеся предметом деятельности предметной области, и субъекты, осуществляющие деятельность в рамках предметной области. Свойства объектов и субъектов реального мира описываются с помощью атрибутов.

Взаимоотношения между сущностями иллюстрируются с помощью связей. Правила и ограничения взаимоотношений описываются с помощью свойств связей. Обычно связи определяют либо зависимости между сущностями, либо влияние одной сущности на другую.

Результатом логического проектирования базы данных подсистемы хранения и обработки данных цифрового двойника ЮРГПУ(НПИ) является логическая модель базы данных, представленная на рисунке 3.2.

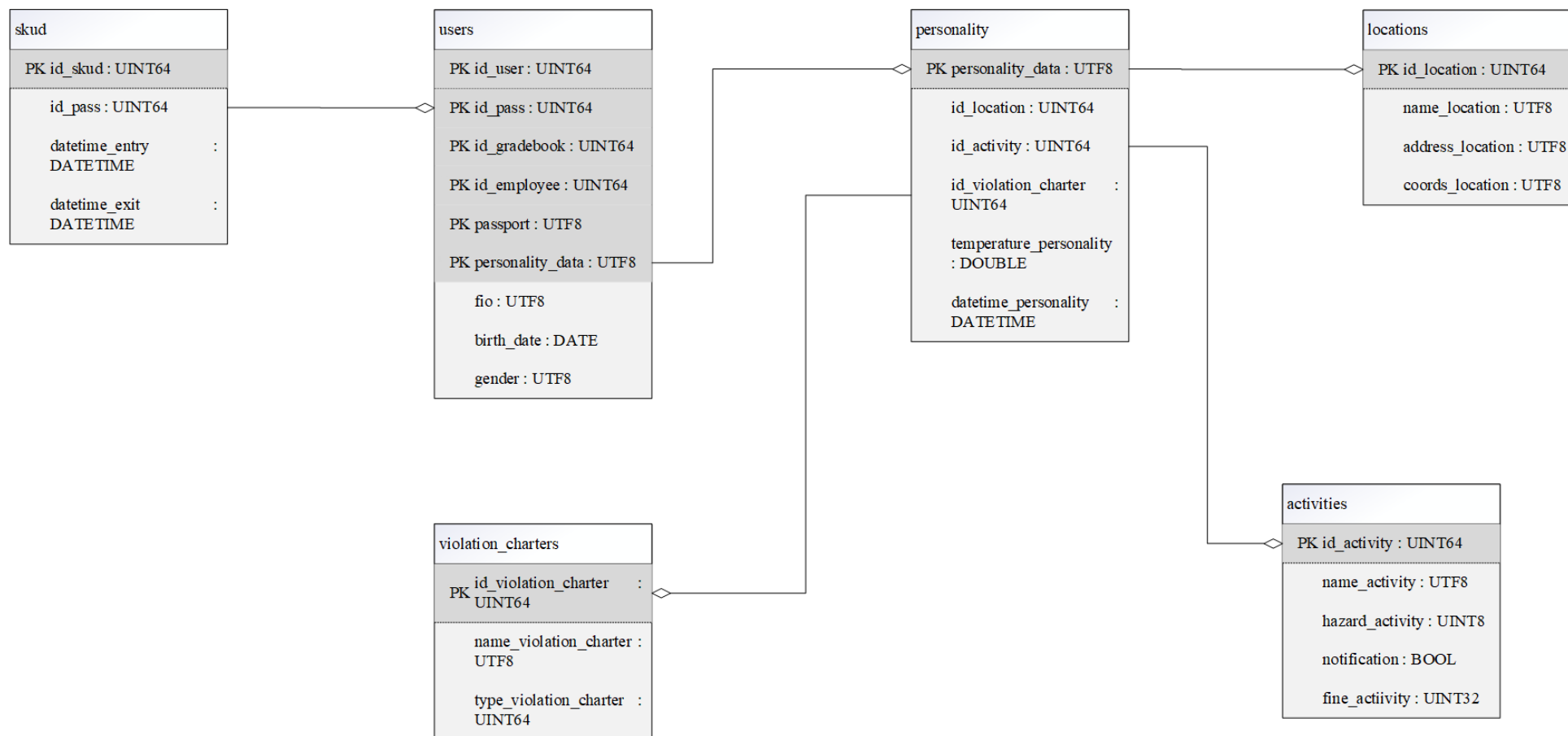


Рисунок 3.2 – Логическая модель базы данных информационной системы цифрового двойника ЮРГПУ (НПИ)

3.3 Описание физической реализации БД

Физическое проектирование базы данных — процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

Физическая модель данных (ФМД) – это модель данных, описанная с помощью средств конкретной СУБД. ФМД строится на базе даталогической путем добавления особенностей конкретной СУБД. К таким особенностям могут относиться поддерживаемые СУБД типы данных, соглашения о присвоении имен таблицам, атрибутам и т.д. ФМД фактически является готовым заданием на создание БД, имея которое можно реализовать БД в выбранной СУБД.

В результате проектирования физической модели данных получились следующие таблицы (таблицы 3.1 – 3.6).

Таблица 3.1 – Описание полей таблицы “personality”

Имя поля	Тип и размер данных	Описание	Тип ключа
personality_data	Utf8	Данные биометрических показателей	РК
id_location	UInt64	Идентификатор местоположения	РК
id_activity	UInt64	Идентификатор активности	РК

id_violation_charter	UInt64	Идентификатор опасного объекта	РК
temperature_personality	Double	Температура	
datetime_personality	Datetime	Дата/Время	

Таблица 3.2 – Описание полей таблицы “locations”

Имя поля	Тип и размер данных	Описание	Тип ключа
id_location	UInt64	Идентификатор местоположения	РК
name_location	Utf8	Название местоположения	
address_location	Utf8	Адрес местоположения	
coords_location	Utf8	Координаты местоположения	

Таблица 3.3 – Описание полей таблицы “activities”

Имя поля	Тип и размер данных	Описание	Тип ключа
id_activity	UInt64	Идентификатор активности	РК
name_activity	Utf8	Название активности	
hazard_activity	Utf8	Уровень нарушения	
notification	Bool	Необходимость оповещения	
fine_activity	UInt32	Штраф	

Таблица 3.4 – Описание полей таблицы “violation_charters”

Имя поля	Тип и размер данных	Описание	Тип ключа
id_violation_charter	Uint64	Идентификатор опасного объекта	РК
name_violation_charter	Utf8	Название опасного объекта	
type_violation_charter	Uint64	Тип опасного объекта	

Таблица 3.5 – Описание полей таблицы “users”

Имя поля	Тип и размер данных	Описание	Тип ключа
id_user	Uint64	Идентификатор пользователя	РК
id_pass	Uint64	Идентификатор электронного пропуска	РК
id_gradebook	Uint64	Идентификатор зачетной книжки	РК
id_employee	Uint64	Идентификатор работника	РК
passport	Utf8	Паспорт	РК
personality_data	Utf8	Данные биометрических показателей	РК
fio	Utf8	Фамилия, имя, отчество	
birth_date	Date	Дата рождения	
gender	Utf8	Пол	

Таблица 3.6 – Описание полей таблицы “skud”

Имя поля	Тип и размер данных	Описание	Тип ключа
id_skud	UInt64	Идентификатор записи СКУД	РК
id_pass	UInt64	Идентификатор электронного пропуска	РК
datetime_entry	Datetime	Дата/Время входа	
datetime_exit	Datetime	Дата/Время выхода	

Физическая модель данных приведена на рисунке 3.3

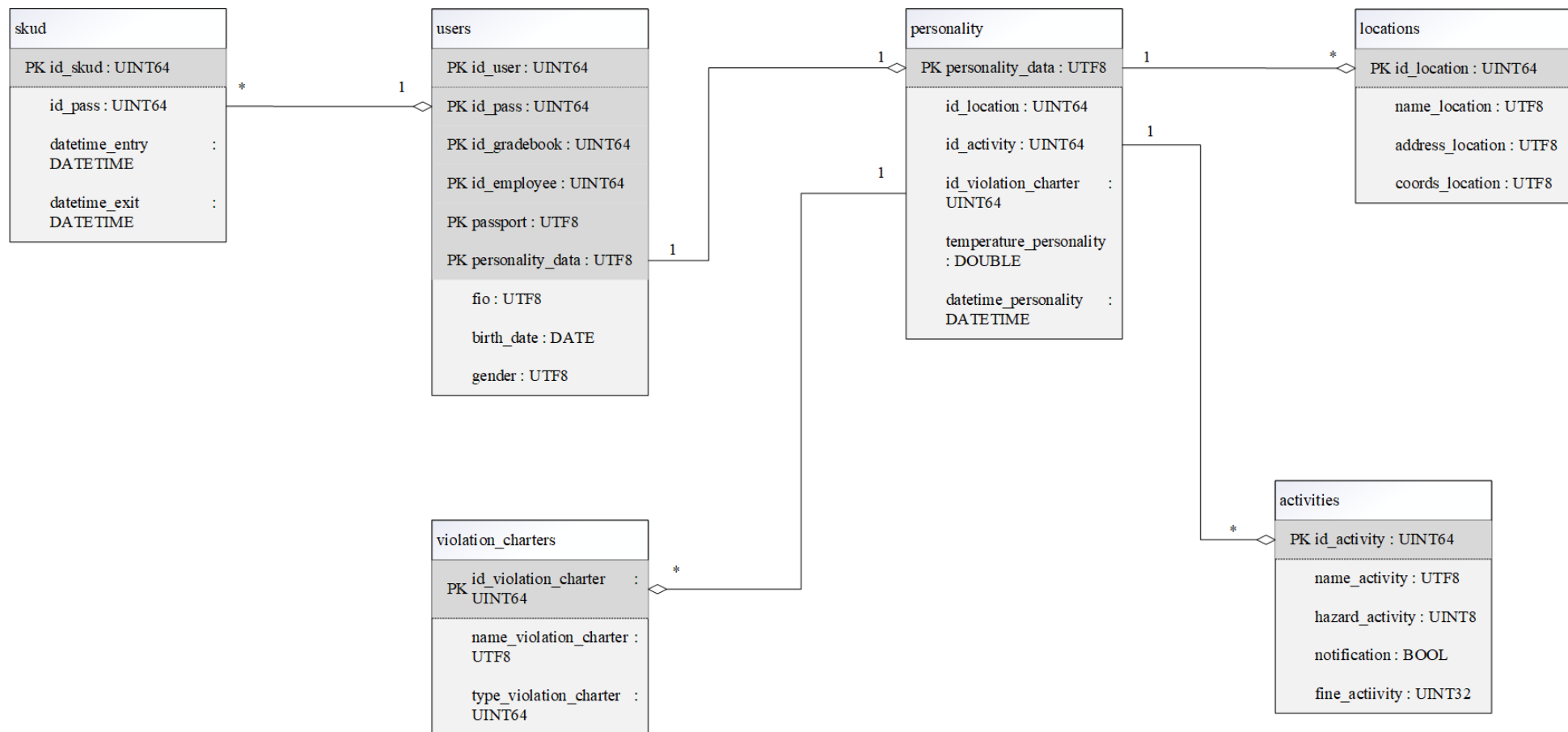


Рисунок 3.3 – Физическая модель базы данных

4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПОДСИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ ЦИФРОВОГО ДВОЙНИКА ЮРГПУ(НПИ)

4.1 Описание структуры программного обеспечения

Программа будет проектироваться по клиент-серверной архитектуре с использованием модульного принципа проектирования. Основной средой web-разработки выступает редактор WebStorm. В качестве сервера баз данных будет выступать система управления базами данных YandexDB [3].

Yandex Database (YDB) — то распределенная отказоустойчивая Distributed SQL СУБД. YDB обеспечивает высокую доступность, горизонтальную масштабируемость, а также строгую консистентность и поддержку ACID-транзакций. Для запросов используется диалект SQL (YQL) [4].

WebStorm (WS) — это интегрированная среда для разработки на JavaScript и связанных с ним технологиях. Как и другие IDE JetBrains, WebStorm позволяет автоматизировать рутинную работу и легко справляться со сложными задачами, делая разработку более увлекательной [5].

В данной дипломной работе используются следующие средства разработки:

- Yandex Database
- Node.js
- Express.js
- HTML вер.5.0 и выше

Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки,

написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Express.js — это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

Express предоставляет тонкий слой фундаментальных функций веб-приложений, которые не мешают вам работать с давно знакомыми и любимыми вами функциями Node.js.

HTML (от англ. HyperText Markup Language – «язык гипертекстовой разметки») – стандартизированный язык разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора. Элементы HTML являются строительными блоками HTML страниц. С помощью HTML разные конструкции, изображения и другие объекты такие как интерактивная веб-формы могут быть встроены в отображаемую страницу. HTML предоставляет средства для создания заголовков, абзацев, списков, ссылок, цитат и других элементов. Элементы HTML выделяются тегами, записанными с использованием угловых скобок.

Структурная схема проекта представлена на рисунке 4.1.

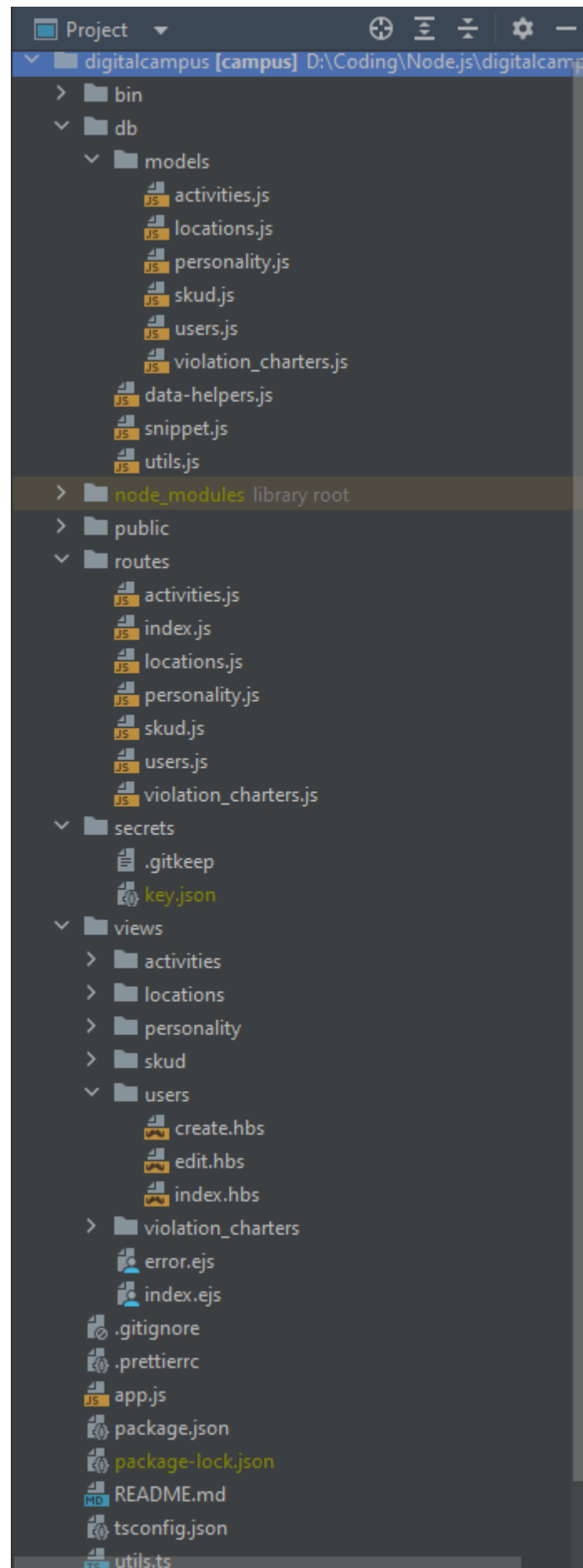


Рисунок 4.1 – Экранная форма структуры проекта

Структура проекта выглядит следующим образом:

- в корневом каталоге «digitalcampus» находятся:
 1. .gitignore – файл, говорящий Git, какие файлы не надо копировать системой управления версий,
 2. .prettierrc – файл настроек модуля PrettierRC,
 3. App.js – ядро, которое служит для обработки различных запросов,
 4. Utils.ts – файл настроек модуля Utils,
 5. package.json и ему подобный package-lock.json – файлы для обработки, скачивания и использования пакетов npm,
 6. другие директории.
- В каталоге «views» лежат файлы, которые впоследствии обработаются в html страницы.
- В каталоге «secrets» лежит файл .gitkeep, который нужен для обработки пустой директории, потому что файл с секретным ключом, нужным для авторизации, не передается.
- В каталоге «routes» лежат маршруты для обработки страниц: главной и для каждой из таблиц базы данных
- В каталоге «db» находятся 3 файла: один для авторизации, второй для обработки запросов запуска, третий для автоматического заполнения базы данных и генерации сущностей таблиц
- В каталоге «db/models» описана сущность для каждой из таблиц

В соответствии с рисунком 4.1 построим структурную схему программной реализации ИС (рисунок 4.2).

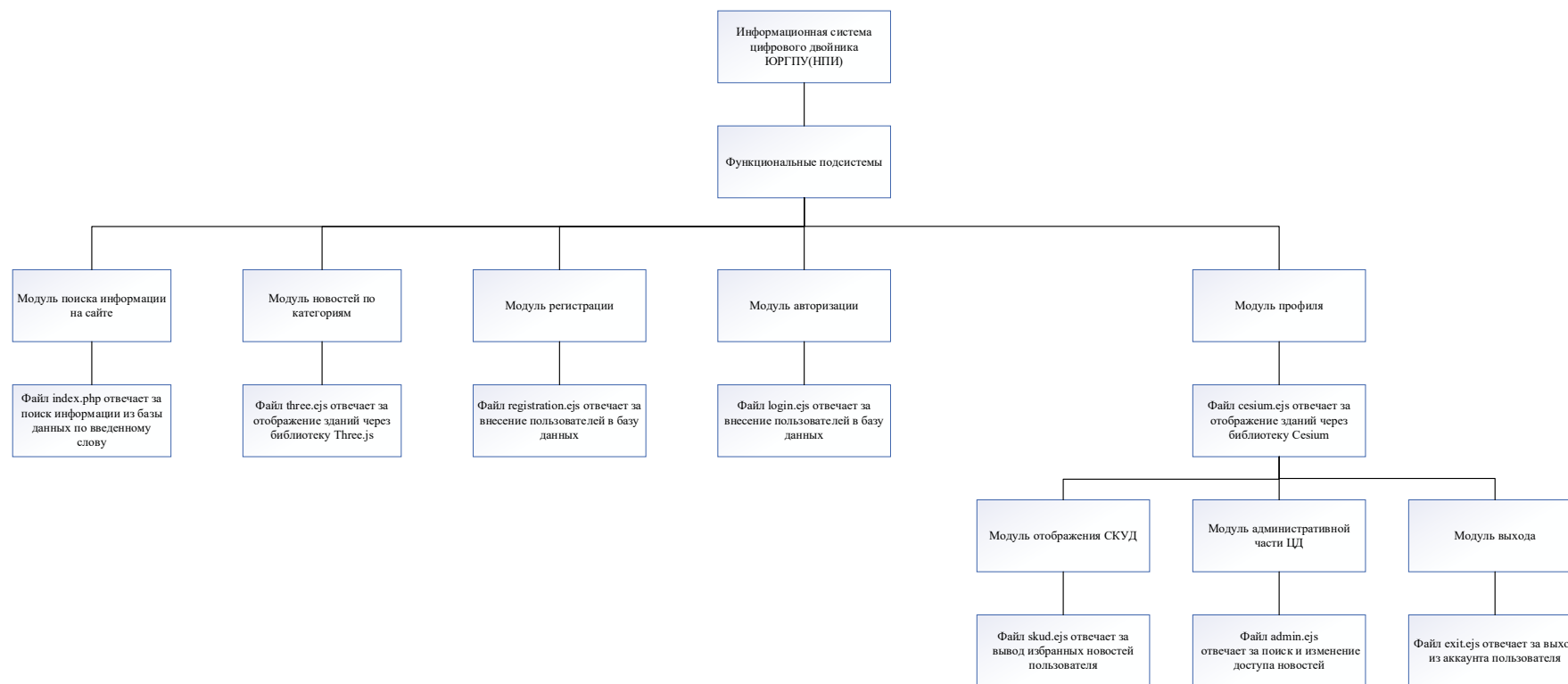


Рисунок 4.2 – Структурная схема программной реализации информационной системы цифрового двойника ЮРГПУ (НПИ)

4.2 Алгоритмизация типовых информационных запросов

Диаграмма деятельности – UML-диаграмма, на которой показаны действия, состояния которых описаны на диаграмме состояний. Под деятельностью (англ. activity) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов – вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Основные цели диаграмм деятельности аналогичны четырем другим диаграммам. Он фиксирует динамическое поведение системы. Другие четыре диаграммы используются для отображения потока сообщений от одного объекта к другому, но диаграмма действий используется для отображения потока сообщений от одного действия к другому.

Построим диаграмму деятельности для информационной системы цифрового двойника ЮРГПУ (НПИ) на базе уникального интерфейса новостных систем (рисунок 4.3).

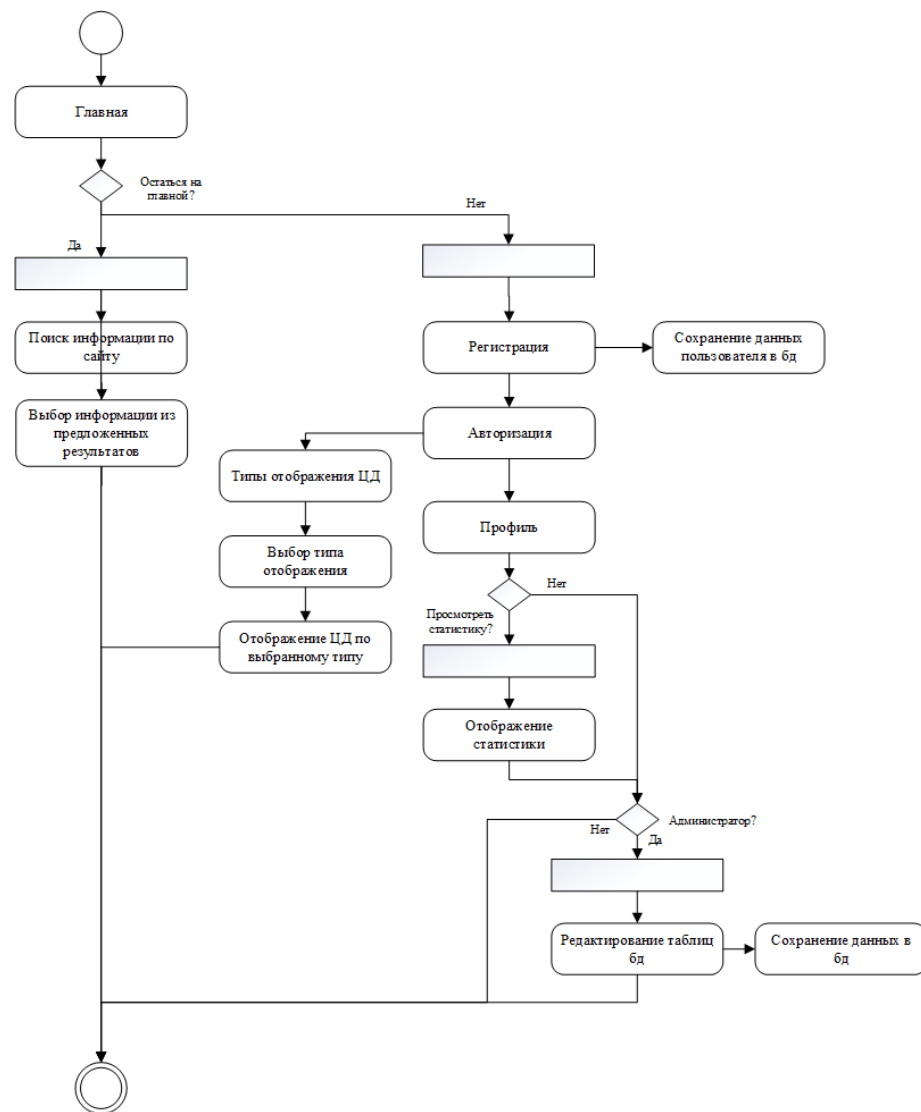


Рисунок 4.3 – Диаграмма деятельности информационной системы цифрового двойника ЮРГПУ (НПИ)

Диаграмма последовательности – UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) информационной системы в рамках прецедента.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

Построим диаграмму последовательности для информационной системы цифрового двойника ЮРГПУ (НПИ) (рисунок 4.4).

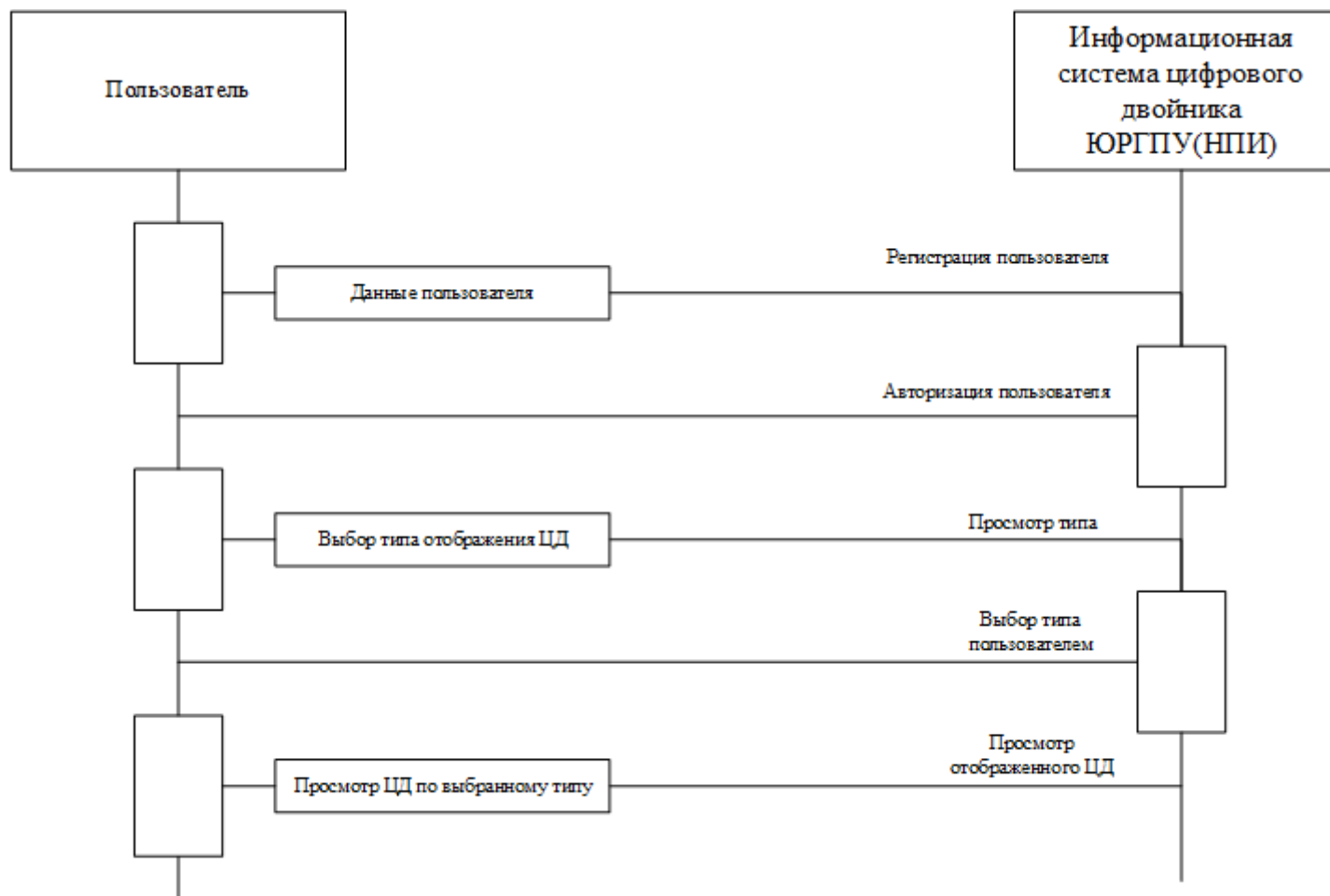
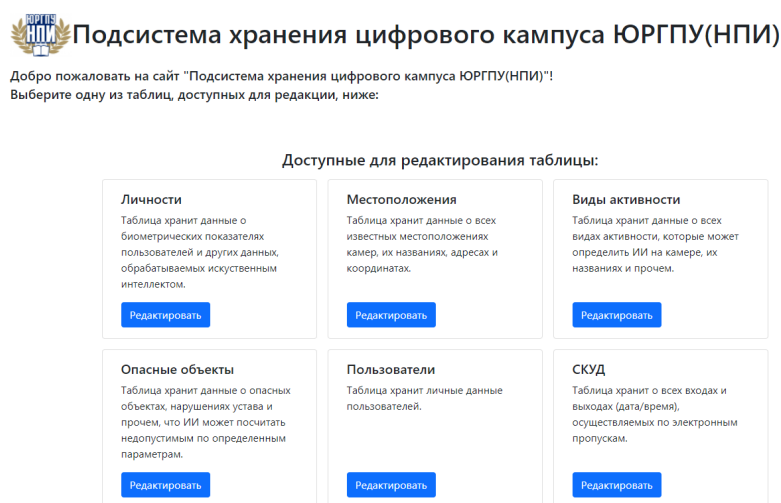


Рисунок 4.4 – Диаграмма последовательности для информационной системы цифрового двойника ЮРГПУ (НПИ)

4.3 Описание пользовательского интерфейса

При создании сайтов один из важнейших этапов разработки – это дизайн сайта. Важно не только содержание и наполнение, но и качественный дизайн, который дополнит и подчеркнет структуру страницы. Пренебрежение последним может свести всю остальную работу на нет, ведь визуальный эффект сайта можно сравнить со встречей по одежке, по которой большинство оценивают уровень компании, выдают кредит доверия

При попадании на сайт пользователя, ему открывается главная страница подсистемы хранения и обработки данных цифрового двойника ЮРГПУ(НПИ), так как учитывается то, что авторизация была произведена в самой информационной системе, которая предназначена только для определённого круга лиц. Вид главной страницы подсистемы представлен на рисунке 4.5



Copyright (c) 2022 | Коновалов Никита Александрович

Рисунок 4.5 – Вид главной страницы подсистемы

При переходе на страницу, нам предлагается выбрать, какую таблицу мы хотим отредактировать. При нажатии на кнопку, происходит открытие страницы с данными этой таблицы. Вид страницы таблицы «Пользователи» представлен на рисунке 4.6

Список пользователей

[На главную](#) | [Добавить пользователя](#)






Идентификатор пользователя	Идентификатор эл. пропуска	Идентификатор зачетной книжки	Идентификатор работника	Паспорт	Биометрические данные	ФИО	Дата рождения	Пол	
1	150833946	1832123	13021	3225 764451	32838118972656	Абрамов Николай Федорович	1996-10-22	М	
2	100833333	0	14231	6021 114451	51116387928887	Анисимова Анастасия Александровна	1970-03-11	Ж	
3	150833945	1842223	0	9021 212333	44942411679400	Кикиморов Георгий Михайлович	1996-11-22	М	
4	112833232	1123423	0	3212 112452	35210933387152	Полякова Мария Валентиновна	1998-12-23	Ж	
5	151233453	2123623	0	5652 112452	40057957250489	Георова Тамара Петровна	2001-07-22	Ж	

Рисунок 4.6 – Вид страницы таблицы «Пользователи»

При нажатии кнопки «Добавить пользователя» нам открывается страница, где необходимо ввести данные. Они будут прочитаны и обработаны базой данных, а после отобразятся и у нас на сайте. Вид страницы «Добавить пользователя» для таблицы «Пользователи» представлен на рисунке 4.7

Добавление пользователя

Идентификатор пользователя:	<input type="text"/>
Идентификатор эл. пропуска:	<input type="text"/>
Идентификатор зачетной книжки:	<input type="text"/>
Идентификатор работника:	<input type="text"/>
Паспорт:	<input type="text" value="0000 000000"/>
Биометрические данные:	<input type="text"/>
ФИО:	<input type="text" value="Иванов Иван Иванович"/>
Дата рождения:	<input type="text" value="дд.мм.гггг"/>
Пол:	<input type="text" value="Мужской"/>

Отправить

[К списку пользователей](#)

Рисунок 4.7 – Вид страницы «Добавить пользователя» для таблицы «Пользователи»

Попробуем внести свои данные и посмотрим, как сервер и база данных отреагируют на это. Вид страницы «Добавить пользователя» для таблицы «Пользователи» с заполненными данными представлен на рисунке 4.9. Вид

страницы таблицы «Пользователи» после обработки отправленных данных представлен на рисунке 4.10.


Добавление пользователя

Идентификатор пользователя:	100
Идентификатор эл. пропуска:	100
Идентификатор зачетной книжки:	100
Идентификатор работника:	100
Паспорт:	6060 100100
Биометрические данные:	32833100100100
ФИО:	Коновалов Никита Александрович
Дата рождения:	20.10.2000
Пол:	Мужской

Отправить

[К списку пользователей](#)

Рисунок 4.9 – Вид страницы «Добавить пользователя» для таблицы «Пользователи» с заполненными данными

 **Список пользователей**

[На главную](#) | [Добавить пользователя](#)



Идентификатор пользователя	Идентификатор эл. пропуска	Идентификатор зачетной книжки	Идентификатор работника	Паспорт	Биометрические данные	ФИО	Дата рождения	Пол	
1	150833946	1832123	13021	3225 764451	32838118972656	Абрамов Николай Федорович	1996-10-22	М	
2	100833333	0	14231	6021 114451	51116387928887	Анисимова Анастасия Александровна	1970-03-11	Ж	
3	150833945	1842223	0	9021 212333	44942411679400	Кикиморов Георгий Михайлович	1996-11-22	М	
4	112833232	1123423	0	3212 112452	35210933387152	Полякова Мария Валентиновна	1998-12-23	Ж	
5	151233453	2123623	0	5652 112452	40057957250489	Георова Тамара Петровна	2001-07-22	Ж	
100	100	100	100	6060 100100	32833100100100	Коновалов Никита Александрович	2000-10-20	М	

Рисунок 4.10 – страницы таблицы «Пользователи» после обработки отправленных данных

При нажатии кнопки «Изменить» (помечена зеленым цветом) происходит открытие страницы «Редактирование пользователя». Вид страницы «Редактирование пользователя» для таблицы «Пользователи» представлен на рисунке 4.11

Редактирование пользователя

Идентификатор пользователя:	4
Идентификатор эл. пропуска:	112833232
Идентификатор зачетной книжки:	1123423
Идентификатор работника:	0
Паспорт:	3212 112452
Биометрические данные:	35210933387152
ФИО:	Полякова Мария Валентиновна
Дата рождения:	23.12.1998
Пол:	Женский

Отправить

[К списку пользователей](#)

Рисунок 4.11 – Вид страницы «Редактирование пользователя» для таблицы «Пользователи»

При нажатии на кнопку «Удалить» (помечена красным цветом) происходит удаление записи из таблицы по первичному ключу. Вид страницы таблицы «Пользователи» после удаления одной записи из таблицы представлен на рисунке 4.12

 **Список пользователей**

[На главную](#) | [Добавить пользователя](#)








Идентификатор пользователя	Идентификатор эл. пропуска	Идентификатор зачетной книжки	Идентификатор работника	Паспорт	Биометрические данные	ФИО	Дата рождения	Пол	
1	150833946	1832123	13021	3225 764451	32838118972656	Абрамов Николай Федорович	1996-10-22	М	 
2	100833333	0	14231	6021 114451	51116387928887	Анисимова Анастасия Александровна	1970-03-11	Ж	 
3	150833945	1842223	0	9021 212333	44942411679400	Кикиморов Георгий Михайлович	1996-11-22	М	 
5	151233453	2123623	0	5652 112452	40057957250489	Георова Тамара Петровна	2001-07-22	Ж	 
100	100	100	100	6060 100100	32833100100100	Коновалов Никита Александрович	2000-10-20	М	 

Рисунок 4.12 – Вид страницы таблицы «Пользователи» после удаления одной записи из таблицы

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломной работы была спроектирована подсистема хранения и обработки данных для информационной системы цифрового двойника ЮРГПУ (НПИ). Были изучены её функции, описан смысл операций и ограничений. На основе выделенных информационных объектов и их атрибутов построена концептуальная и логическая модель, т.е. проведено проектирование баз данных: концептуальное, логическое и физическое. Было определено требуемое техническое и программное обеспечение необходимое для эксплуатации данного продукта.

Подсистема хранения и обработки данных цифрового двойника ЮРГПУ (НПИ) им. М.И. Платова спроектирована посредством среды разработки WebStorm. Написана программа с помощью языка гипертекстовой разметки HTML, с использованием каскадных таблиц стилей CSS, программной платформы Node.js. Физическая модель данных представлена реляционными таблицами, вся информация которых содержится в базе данных Yandex Database. Также спроектирована и реализована база данных, описанная созданием базы и запросов на языке SQL.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Модель электронного (смарт) кампуса университета – 2022 г / ВГУЭС [Электронный ресурс] URL: https://e-campus.vvsu.ru/latest/article/10134139/odel_elektronnogo_smart_kampusa/
2. Черноморов Г.А. Базы данных в среде промышленных СУБД: учебное пособие для студентов высших учебных заведений, обучающихся по специальности "Прикладная информатика (по областям)" / Г. А. Черноморов. – Новочеркасск: Южно-Российский гос.
3. Yandex Managed Service for YDB – 2022 г / YandexCloud [Электронный ресурс] URL: <https://cloud.yandex.ru/docs/ydb/>
4. SQL запросы быстро. Часть 1 – 2006–2021 гг / Хабр. [Электронный ресурс] URL: <https://habr.com/ru/post/480838/>
5. WebStorm – 2000–2022 гг / WebStorm [Электронный ресурс] URL: <https://www.jetbrains.com/ru-ru/webstorm/>

ПРИЛОЖЕНИЕ А – Фрагменты листинга программы

1) routes/index.js

```
const express = require('express');
const router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Подсистема хранения цифрового кампуса
ЮРГПУ(НПИ)' });
});

module.exports = router;
```

2) routes/users.js

```
const express = require('express');
const router = express.Router();
express().set('view engine', 'hbs');
let Handlebars = require('hbs');
let moment = require('moment');

const readTableByName = require('../db/snippet').readTableByName;
const editSimpleByName = require('../db/snippet').editSimpleByName;
const selectSimpleByName = require('../db/snippet').selectSimpleByName;
const deleteSimpleByName = require('../db/snippet').deleteSimpleByName;

const USERS_TABLE = 'users';

const urlencodedParser = express.urlencoded({ extended: false });

// получение списка пользователей
router.get('/', async function(req, res) {
```

```

let data = (await readTableByName(USERS_TABLE));
// console.log(data);
res.render('users/index.hbs', {
  title: "Список пользователей",
  users: data,
});
Handlebars.registerHelper('formatTime', function (date, format) {
  let mmnt = moment(date);
  return mmnt.format(format);
});
});

// возвращаем форму для добавления данных
router.get('/create', function(req, res) {
  res.render('users/create.hbs', {
    title: "Добавление пользователя",
  });
});

// получаем отправленные данные и добавляем их в БД
router.post("/create", urlencodedParser, async function (req, res) {
  if(!req.body) return res.sendStatus(400);

  const id_user = req.body.idUser;
  const id_pass = req.body.idPass;
  const id_gradebook = req.body.idGradebook;
  const id_employee = req.body.idEmployee;
  const passport = req.body.passport;
  const personality_data = req.body.personalityData;
  const fio = req.body.fio;

```

```

const birth_date = req.body.birthDate;
const gender = req.body.gender;

console.log(`${id_user}, ${id_pass}, ${id_gradebook}, ${id_employee},
"${passport}",    "${personality_data}",    "${fio}",    Date(${birth_date}),
"${gender}");

try {
    let table = await selectSimpleByName(USERS_TABLE, 'id_user',
id_user);
    if (table.length === 0){
        await editSimpleByName(USERS_TABLE, `insert`, `${id_user},
${id_pass},    ${id_gradebook},    ${id_employee},    "${passport}",
"${personality_data}", "${fio}", Date("${birth_date}"), "${gender}");
        res.redirect("/users");
    } else if (
        parseInt(table[0].idUser) !== parseInt(id_user) &&
        parseInt(table[0].idPass) !== parseInt(id_pass) &&
        parseInt(table[0].idGradebook) !== parseInt(id_gradebook) &&
        parseInt(table[0].idEmployee) !== parseInt(id_employee) &&
        table[0].passport.toString() !== passport.toString() &&
        table[0].personalityData.toString() !== personality_data.toString()
    ) {
        await editSimpleByName(USERS_TABLE, `insert`, `${id_user},
${id_pass},    ${id_gradebook},    ${id_employee},    "${passport}",
"${personality_data}", "${fio}", Date("${birth_date}"), "${gender}");
        res.redirect("/users");
    } else {
        console.log("It is not possible to add data to the table: the primary key
has already been used (Error 409)");
    }
}

```



```

        res.status(409).send("Невозможно добавить данные в таблицу:
первичный ключ уже был использован");
    }
}
catch (err) {
    console.log("It is not possible to add data to the table: the primary key has
already been used (Error 409)");
    console.log(err);
    res.status(409).send("Невозможно добавить данные в таблицу:
первичный ключ уже был использован");
}
});

```

```

// получаем id удаляемого пользователя и удаляем его из бд
router.post("/delete/:id", async function(req, res){

```

```

    const id = req.params.id;

```

```

    try {
        await deleteSimpleByName(USERS_TABLE, 'id_user', id);
        res.redirect("/users");
    }
    catch (err) {
        console.log(err);
    }

```

```

});

```

```

// возвращаем форму для добавления данных
router.get('/edit', function(req, res) {

```

```
res.redirect(`/users`);  
});
```

// получем id редактируемого пользователя, получаем его из бд и отправляем с формой редактирования

```
router.get("/edit/:id", async function(req, res){  
  const id = req.params.id;  
  
  try {  
    let data = (await selectSimpleByName(USERS_TABLE, 'id_user', id));  
    console.log(data);  
    res.render("users/edit.hbs", {  
      title: "Редактирование пользователя",  
      user: data[0]  
    });  
    Handlebars.registerHelper('formatTime', function (date, format) {  
      let mmnt = moment(date);  
      return mmnt.format(format);  
    });  
    Handlebars.registerHelper('isSelected', function (first, second) {  
      return first === second ? 'selected' : "";  
    });  
  }  
  catch (err) {  
    console.log(err);  
  }  
});
```

```
router.post("/edit/:id", urlencodedParser, async function (req, res) {  
  if(!req.body) return res.sendStatus(400);
```

```

const id_user = req.params.id;

//const id_user = req.body.idUser;
const id_pass = req.body.idPass;
const id_gradebook = req.body.idGradebook;
const id_employee = req.body.idEmployee;
const passport = req.body.passport;
const personality_data = req.body.personalityData;
const fio = req.body.fio;
const birth_date = req.body.birthDate;
const gender = req.body.gender;

//console.log(`${id_user}, ${id_pass}, ${id_gradebook}, ${id_employee},
"${passport}",    "${personality_data}",    "${fio}",    Date(`${birth_date}`),
"${gender}");

try {
    await deleteSimpleByName(USERS_TABLE, 'id_user', id_user);
    await    editSimpleByName(USERS_TABLE,    `insert`,`${id_user}`,
`${id_pass}`,    `${id_gradebook}`,    `${id_employee}`,    "${passport}",
"${personality_data}", "${fio}", Date("${birth_date}"), "${gender}");
    res.redirect(`/users`);
}
catch (err) {
    console.log(err);
    //res.status(409).send("Невозможно добавить данные в таблицу:
первичный ключ уже был использован");
}
});

```

```
module.exports = router;
```

3) db/models/users.js

```
"use strict";
```

```
Object.defineProperty(exports, "__esModule", { value: true });
```

```
const ydbSDK = require("ydb-sdk");
```

```
let __decorate = (this && this.__decorate) || function (decorators, target, key, desc) {  
    var c = arguments.length, r = c < 3 ? target : desc === null ? desc =  
Object.getOwnPropertyDescriptor(target, key) : desc, d;  
    if (typeof Reflect === "object" && typeof Reflect.decorate === "function")  
r = Reflect.decorate(decorators, target, key, desc);  
    else for (var i = decorators.length - 1; i >= 0; i--) if (d = decorators[i]) r =  
(c < 3 ? d(r) : c > 3 ? d(target, key, r) : d(target, key)) || r;  
    return c > 3 && r && Object.defineProperty(target, key, r), r;  
};
```

```
class Users extends ydbSDK.TypedData {  
    constructor(data) {  
        super(data);  
        this.idUser = data.idUser;  
        this.idPass = data.idPass;  
        this.idGradebook = data.idGradebook;  
        this.idEmployee = data.idEmployee;  
        this.passport = data.passport;  
        this.personalityData = data.personalityData;  
        this.fio = data.fio;  
        this.birthDate = data.birthDate;  
        this.gender = data.gender;  
    }  
}
```

```

        static create(idUser, idPass, idGradebook, idEmployee, passport,
personalityData, fio, birthDate, gender) {
            return new this({ idUser, idPass, idGradebook, idEmployee, passport,
personalityData, fio, birthDate, gender });
        }
    };

    __decorate([
        ydbSDK.declareType(ydbSDK.Types.UINT64)
    ], Users.prototype, "idUser", void 0);

    __decorate([
        ydbSDK.declareType(ydbSDK.Types.UINT64)
    ], Users.prototype, "idPass", void 0);

    __decorate([
        ydbSDK.declareType(ydbSDK.Types.UINT64)
    ], Users.prototype, "idGradebook", void 0);

    __decorate([
        ydbSDK.declareType(ydbSDK.Types.UINT64)
    ], Users.prototype, "idEmployee", void 0);

    __decorate([
        ydbSDK.declareType(ydbSDK.Types.UTF8)
    ], Users.prototype, "passport", void 0);

    __decorate([
        ydbSDK.declareType(ydbSDK.Types.UTF8)
    ], Users.prototype, "personalityData", void 0);

    __decorate([
        ydbSDK.declareType(ydbSDK.Types.UTF8)
    ], Users.prototype, "fio", void 0);

    __decorate([
        ydbSDK.declareType(ydbSDK.Types.DATE)
    ], Users.prototype, "birthDate", void 0);

```

```

__decorate([
  ydbSDK.declareType(ydbSDK.Types.UTF8)
], Users.prototype, "gender", void 0);
Users = __decorate([
  ydbSDK.withTypeOptions({ namesConversion:
ydbSDK.snakeToCamelCaseConversion })
], Users);
module.exports = Users;

```

4) db/data-helpers.js

```

'use strict';
Object.defineProperty(exports, '__esModule', { value: true });

exports.getPersonalityData = exports.getLocationsData =
exports.getActivitiesData = exports.getViolationChartersData =
exports.getUsersData = exports.getSkudData = void 0;

```

```

const Personality = require('./models/personality');
const Locations = require('./models/locations');
const Activities = require('./models/activities');
const Violation_Charters = require('./models/violation_charters');
const Users = require('./models/users');
const Skud = require('./models/skud');

/**
 * @desc randomTodayTimeGenerator - generates random time intervals
from the date and time of function generation in a given range
 */
function rttGen(min, max) {
  function getRandomInt(miN, maX) {
    miN = Math.ceil(miN);

```

```

    maX = Math.floor(maN);
    return Math.floor(Math.random() * (maX - miN)) + miN;
}
let d = new Date()
d.setHours(d.getHours() + getRandomInt(min, max),d.getMinutes() +
getRandomInt(min, max) + getRandomInt(min, max));
return d;
}

```

```

function getPersonalityData() {
    return Personality.asTypedCollection([
        Personality.create("51116387928887", 1, 1, 1,36.6, rttGen(-15,5)),
        Personality.create("32838118972656", 3, 3, 1,35.6, rttGen(-15,5)),
        Personality.create("35210933387152", 2, 1, 1,36.4, rttGen(-15,5)),
        Personality.create("44942411679400", 1, 2, 3,36.5, rttGen(-15,5)),
        Personality.create("40057957250489", 3, 3, 1,35.9, rttGen(-15,5)),
    ]);
}

```

```

exports.getPersonalityData = getPersonalityData;

```

```

function getLocationsData() {
    return Locations.asTypedCollection([
        Locations.create(1,"Главный корпус","346428, Ростовская обл., г.
Новочеркасск, ул. Просвещения, 132","47.416614, 40.086592"),
        Locations.create(2,"Горный корпус","346428, Ростовская обл., г.
Новочеркасск, ул. Просвещения, 132","47.417595, 40.083563"),
        Locations.create(3,"Общежитие 11","346428, Ростовская обл., г.
Новочеркасск, ул. Энгельса, 85А","47.416812, 40.074263"),
    ]);
}

```

```
exports.getLocationsData = getLocationsData;
```

```
function getActivitiesData() {  
    return Activities.asTypedCollection([  
        Activities.create(1,"Идет",0, false, 0),  
        Activities.create(2,"Сидит",0, false, 0),  
        Activities.create(3,"Курит",5, true, 1500),  
    ]);  
}  
exports.getActivitiesData = getActivitiesData;
```

```
function getViolationChartersData() {  
    return Violation_Charters.asTypedCollection([  
        Violation_Charters.create(1,"Все хорошо",0),  
        Violation_Charters.create(2,"Неудовлетворяющая одежда",1),  
        Violation_Charters.create(3,"Холодное оружие",3),  
    ]);  
}  
exports.getViolationChartersData = getViolationChartersData;
```

```
function getUsersData() {  
    return Users.asTypedCollection([  
        Users.create(1,150833946,1832123,13021,"3225  
764451","32838118972656","Абрамов Николай Федорович",new Date("1996-  
10-22"),"М"),  
        Users.create(2,100833333,0,14231,"6021  
114451","51116387928887","Анисимова Анастасия Алексадровна",new  
Date("1970-03-11"),"Ж"),  
    ]);  
}
```



```

        Users.create(3,150833945,1842223,0,"9021
212333","44942411679400","Кикиморов      Георгий      Михайлович",new
Date("1996-11-22"),"M"),
        Users.create(4,112833232,1123423,0,"3212
112452","35210933387152","Полякова Мария Валентиновна",new Date("1998-
12-23"),"Ж"),
        Users.create(5,151233453,2123623,0,"5652
112452","40057957250489","Георова Тамара Петровна",new Date("2001-07-
22"),"Ж"),
    ];
}
exports.getUsersData = getUsersData;

```

```

function getSkudData() {
    return Skud.asTypedCollection([
        Skud.create(1,150833946, rttGen(-15,5), rttGen(0,8)),
        Skud.create(2,150833946, rttGen(-15,5), rttGen(0,8)),
        Skud.create(3,150833946, rttGen(-15,5), rttGen(0,8)),
        Skud.create(4,112833232, rttGen(-15,5), rttGen(0,8)),
        Skud.create(5,112833232, rttGen(-15,5), rttGen(0,8)),
        Skud.create(6,150833946, rttGen(-15,5), rttGen(0,8)),
        Skud.create(7,112833232, rttGen(-15,5), rttGen(0,8)),
        Skud.create(8,151233453, rttGen(-15,5), rttGen(0,8)),
        Skud.create(9,150833946, rttGen(-15,5), rttGen(0,8)),
        Skud.create(10,151233453, rttGen(-15,5), rttGen(0,8)),
    ]);
}
exports.getSkudData = getSkudData;

```

5) db/spippet.js

```
'use strict';
```

```

Object.defineProperty(exports, '__esModule', { value: true });
exports.run = void 0;
const ydbSDK = require('ydb-sdk');
const utils = require('./utils');
const data_helpers = require('./data-helpers');

const PERSONALITY_TABLE = 'personality';
const LOCATIONS_TABLE = 'locations';
const ACTIVITIES_TABLE = 'activities';
const VIOLATION_CHARTERS_TABLE = 'violation_charters';
const USERS_TABLE = 'users';
const SKUD_TABLE = 'skud';

const Personality = require('./models/personality');
const Locations = require('./models/locations');
const Activities = require('./models/activities');
const Violation_Charters = require('./models/violation_charters');
const Users = require('./models/users');
const Skud = require('./models/skud');

async function createTables(session, logger) {
  logger.info('Dropping old tables...');
  await session.dropTable(PERSONALITY_TABLE);
  await session.dropTable(LOCATIONS_TABLE);
  await session.dropTable(ACTIVITIES_TABLE);
  await session.dropTable(VIOLATION_CHARTERS_TABLE);
  await session.dropTable(USERS_TABLE);
  await session.dropTable(SKUD_TABLE);
  logger.info('Creating tables...');

```

```

        await session.createTable(PERSONALITY_TABLE, new
ydbSDK.TableDescription()
            .withColumn(new ydbSDK.Column('personality_data',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))
            .withColumn(new ydbSDK.Column('id_location',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
            .withColumn(new ydbSDK.Column('id_activity',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
            .withColumn(new ydbSDK.Column('id_violation_charter',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
            .withColumn(new ydbSDK.Column('temperature_personality',
ydbSDK.Types.optional(ydbSDK.Types.DOUBLE)))
            .withColumn(new ydbSDK.Column('datetime_personality',
ydbSDK.Types.optional(ydbSDK.Types.DATETIME)))
            .withPrimaryKey('personality_data'));
        await session.createTable(LOCATIONS_TABLE, new
ydbSDK.TableDescription()
            .withColumn(new ydbSDK.Column('id_location',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
            .withColumn(new ydbSDK.Column('name_location',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))
            .withColumn(new ydbSDK.Column('address_location',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))
            .withColumn(new ydbSDK.Column('coords_location',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))
            .withPrimaryKey('id_location'));
        await session.createTable(ACTIVITIES_TABLE, new
ydbSDK.TableDescription()
            .withColumn(new ydbSDK.Column('id_activity',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))

```

```

        .withColumn(new ydbSDK.Column('name_activity',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))
        .withColumn(new ydbSDK.Column('hazard_activity',
ydbSDK.Types.optional(ydbSDK.Types.UINT8)))
        .withColumn(new ydbSDK.Column('notification',
ydbSDK.Types.optional(ydbSDK.Types.BOOL)))
        .withColumn(new ydbSDK.Column('fine_activity',
ydbSDK.Types.optional(ydbSDK.Types.UINT32)))
        .withPrimaryKey('id_activity'));
    await session.createTable(VIOLATION_CHARTERS_TABLE, new
ydbSDK.TableDescription()
        .withColumn(new ydbSDK.Column('id_violation_charter',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
        .withColumn(new ydbSDK.Column('name_violation_charter',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))
        .withColumn(new ydbSDK.Column('type_violation_charter',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
        .withPrimaryKey('id_violation_charter'));
    await session.createTable(USERS_TABLE, new
ydbSDK.TableDescription()
        .withColumn(new ydbSDK.Column('id_user',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
        .withColumn(new ydbSDK.Column('id_pass',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
        .withColumn(new ydbSDK.Column('id_gradebook',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
        .withColumn(new ydbSDK.Column('id_employee',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
        .withColumn(new ydbSDK.Column('passport',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))

```

```

        .withColumn(new ydbSDK.Column('personality_data',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))
        .withColumn(new ydbSDK.Column('fio',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))
        .withColumn(new ydbSDK.Column('birth_date',
ydbSDK.Types.optional(ydbSDK.Types.DATE)))
        .withColumn(new ydbSDK.Column('gender',
ydbSDK.Types.optional(ydbSDK.Types.UTF8)))
        .withPrimaryKeys('id_user', 'id_pass', 'id_gradebook', 'id_employee',
'passport', 'personality_data'));
    await session.createTable(SKUD_TABLE, new
ydbSDK.TableDescription()
        .withColumn(new ydbSDK.Column('id_skud',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
        .withColumn(new ydbSDK.Column('id_pass',
ydbSDK.Types.optional(ydbSDK.Types.UINT64)))
        .withColumn(new ydbSDK.Column('datetime_entry',
ydbSDK.Types.optional(ydbSDK.Types.DATETIME)))
        .withColumn(new ydbSDK.Column('datetime_exit',
ydbSDK.Types.optional(ydbSDK.Types.DATETIME)))
        .withPrimaryKey('id_skud'));
    logger.info('Tables has been created');
}

```

```

async function fillTablesWithData(session, logger) {
    const query = `
${utils.SYNTAX_V1}

```

```

DECLARE $personalityData AS List<Struct<
    personality_data: Utf8,

```

```

    id_location: Uint64,
    id_activity: Uint64,
    id_violation_charter: Uint64,
    temperature_personality: Double,
    datetime_personality: Datetime>>;
DECLARE $locationsData AS List<Struct<
    id_location: Uint64,
    name_location: Utf8,
    address_location: Utf8,
    coords_location: Utf8>>;
DECLARE $activitiesData AS List<Struct<
    id_activity: Uint64,
    name_activity: Utf8,
    hazard_activity: Uint8,
    notification: Bool,
    fine_activity: Uint32>>;
DECLARE $violationChartersData AS List<Struct<
    id_violation_charter: Uint64,
    name_violation_charter: Utf8,
    type_violation_charter: Uint64>>;
DECLARE $usersData AS List<Struct<
    id_user: Uint64,
    id_pass: Uint64,
    id_gradebook: Uint64,
    id_employee: Uint64,
    passport: Utf8,
    personality_data: Utf8,
    fio: Utf8,
    birth_date: Date,
    gender: Utf8>>;

```

```

DECLARE $skudData AS List<Struct<
    id_skud: UInt64,
    id_pass: UInt64,
    datetime_entry: Datetime,
    datetime_exit: Datetime>>;

REPLACE INTO ${PERSONALITY_TABLE}
SELECT
    personality_data,
    id_location,
    id_activity,
    id_violation_charter,
    temperature_personality,
    datetime_personality
FROM AS_TABLE($personalityData);

REPLACE INTO ${LOCATIONS_TABLE}
SELECT
    id_location,
    name_location,
    address_location,
    coords_location
FROM AS_TABLE($locationsData);

REPLACE INTO ${ACTIVITIES_TABLE}
SELECT
    id_activity,
    name_activity,
    hazard_activity,
    notification,

```

```

        fine_activity
FROM AS_TABLE($activitiesData);

REPLACE INTO ${VIOLATION_CHARTERS_TABLE}
SELECT
    id_violation_charter,
    name_violation_charter,
    type_violation_charter
FROM AS_TABLE($violationChartersData);

REPLACE INTO ${USERS_TABLE}
SELECT
    id_user,
    id_pass,
    id_gradebook,
    id_employee,
    passport,
    personality_data,
    fio,
    birth_date,
    gender
FROM AS_TABLE($usersData);

REPLACE INTO ${SKUD_TABLE}
SELECT
    id_skud,
    id_pass,
    datetime_entry,
    datetime_exit
FROM AS_TABLE($skudData);

```



```

`;

async function fillTable() {
  logger.info('Inserting data to tables, preparing query...');
  const preparedQuery = await session.prepareQuery(query);
  logger.info('Query has been prepared, executing...');
  await session.executeQuery(preparedQuery, {
    '$personalityData': data_helpers.getPersonalityData(),
    '$locationsData': data_helpers.getLocationsData(),
    '$activitiesData': data_helpers.getActivitiesData(),
    '$violationChartersData': data_helpers.getViolationChartersData(),
    '$usersData': data_helpers.getUsersData(),
    '$skudData': data_helpers.getSkudData()
  });
  logger.info('Executing completed');
}

await ydbSDK.withRetries(fillTable);
}

// async function select1Simple(session, logger) {
//   const query = `
//     ${utils.SYNTAX_V1}
//     SELECT e.title, s.title
//     FROM episodes AS e
//     FULL JOIN seasons AS s
//     USING (series_id);`;
//   logger.info('Making a simple select...');
//   const { resultSets } = await session.executeQuery(query);
//   const result = data_helpers.Table.createNativeObjects(resultSets[0]);
//   console.log(`selectSimple result: ${JSON.stringify(result, null, 2)}`);
// }

```

```

    async function editSimple(table_name, edit_mode, table_col, session) {
        let table_rows;

        table_name === PERSONALITY_TABLE ? table_rows =
`personality_data,      id_location,      id_activity,      id_violation_charter,
temperature_personality, datetime_personality` : ";

        table_name === LOCATIONS_TABLE ? table_rows = `id_location,
address_location, coords_location, name_location` : ";

        table_name === ACTIVITIES_TABLE ? table_rows = `id_activity,
name_activity, hazard_activity, notification, fine_activity` : ";

        table_name === VIOLATION_CHARTERS_TABLE ? table_rows =
`id_violation_charter, name_violation_charter, type_violation_charter` : ";

        table_name === USERS_TABLE ? table_rows = `id_user, id_pass,
id_gradebook, id_employee, passport, personality_data, fio, birth_date, gender` : ";

        table_name === SKUD_TABLE ? table_rows = `id_skud, id_pass,
datetime_entry, datetime_exit` : ";

        const query = `
        ${utils.SYNTAX_V1}
        ${edit_mode.toUpperCase()} INTO ${table_name} (${table_rows})
VALUES (${table_col});
        `;

        console.log(`Making an ${edit_mode} in table "${table_name}" values
        (${table_col})...`);

        await session.executeQuery(query);
        console.log('Upsert completed.');
```

```

    }

    async function selectSimple(table_name, table_col_name, table_col_value,
session) {
        let res;

```

```

    const query = `
    ${utils.SYNTAX_V1}
    SELECT * FROM ${table_name} WHERE ${table_col_name} ==
    ${table_col_value};
    `;

    console.log(`Making a simple select in table "${table_name}" where
    ${table_col_name}=${table_col_value}...`);

    const { resultSets } = await session.executeQuery(query);

    table_name      ===      PERSONALITY_TABLE      ?      res      =
    Personality.createNativeObjects(resultSets[0]) : "";

    table_name      ===      LOCATIONS_TABLE      ?      res      =
    Locations.createNativeObjects(resultSets[0]) : "";

    table_name      ===      ACTIVITIES_TABLE      ?      res      =
    Activities.createNativeObjects(resultSets[0]) : "";

    table_name      ===      VIOLATION_CHARTERS_TABLE      ?      res      =
    Violation_Charters.createNativeObjects(resultSets[0]) : "";

    table_name      ===      USERS_TABLE      ?      res      =
    Users.createNativeObjects(resultSets[0]) : "";

    table_name      ===      SKUD_TABLE      ?      res      =
    Skud.createNativeObjects(resultSets[0]) : "";

    console.log(`Select simple result: ${JSON.stringify(res, null, 2)} `);
    return res;
}

async function deleteSimple(table_name, table_col_name, table_col_value,
session) {

    const query = `

```

```

    ${utils.SYNTAX_V1}
    DELETE FROM ${table_name} WHERE ${table_col_name} ==
    ${table_col_value};
    `;

    console.log(`Delete in table "${table_name}" where
    ${table_col_name}=${table_col_value}...`);
    await session.executeQuery(query);
    console.log('Delete completed.');
```

```

    }

    async function readTable(table_name, session, settings) {
        let res;
        await session.streamReadTable(table_name, (result) => {
            const resultSet = result.resultSet;
            if (resultSet) {
                table_name === PERSONALITY_TABLE ? res =
                Personality.createNativeObjects(resultSet) : "";
                table_name === LOCATIONS_TABLE ? res =
                Locations.createNativeObjects(resultSet) : "";
                table_name === ACTIVITIES_TABLE ? res =
                Activities.createNativeObjects(resultSet) : "";
                table_name === VIOLATION_CHARTERS_TABLE ? res =
                Violation_Charters.createNativeObjects(resultSet) : "";
                table_name === USERS_TABLE ? res =
                Users.createNativeObjects(resultSet) : "";
                table_name === SKUD_TABLE ? res =
                Skud.createNativeObjects(resultSet) : "";
            }
        }, settings);
        console.log(`Data output from the table "${table_name}"`);
    }

```

```

    return res;
}

let driver;

async function run(logger, endpoint, database, args) {
    logger.info('Driver initializing...');
    const saKeyFile = args.serviceAccountKeyFile;
    const saCredentials = ydbSDK.getSACredentialsFromJson(saKeyFile);
    const authService = new ydbSDK.IamAuthService(saCredentials);
    driver = new ydbSDK.Driver({ endpoint, database, authService });
    const timeout = 10000;
    if (!await driver.ready(timeout)) {
        logger.fatal(`Driver has not become ready in ${timeout}ms!`);
        process.exit(1);
    }
    logger.info('Done');

    await driver.tableClient.withSession(async (session) => {
        await createTables(session, logger);
        await fillTablesWithData(session, logger);
    });

    // await driver.tableClient.withSession(async (session) => {
    //     await deleteSimple(LOCATIONS_TABLE, 0, session);
    // });

    // await driver.tableClient.withSession(async (session) => {

```

```

        // console.log(await selectSimple(LOCATIONS_TABLE, `id_location`,
1, session));
        // });

        //editSimpleByName(SKUD_TABLE, `upsert`,`6, "346428, Ростовская
обл., г. Новочеркасск, ул. Энгельса, 85А", "47.416812, 40.074263",
"Общежитие 10");

        // deleteSimpleByName(LOCATIONS_TABLE, 'id_location',4);

        // await driver.tableClient.withSession(async (session) => {
        // //logger.info('Read whole table, unsorted:');
        //
        // // s.forEach((users) => {
        // // console.log(`# Users, Id: ${users.idUser}, Login: ${users.fio}`);
        // // });
        //
        // // await readTable(session, logger);
        //
        // // await readTable('users', session, logger);
        // // await selectSimple(session, logger);
        // });

        await driver.destroy();
    }

    exports.run = run;

    function readTableByName(table_name) {
        return driver.tableClient.withSession( (session) => readTable(table_name,
session));
    }

```

```

    }

    exports.readTableByName = readTableByName;

    async function editSimpleByName(table_name, edit_mode, table_col) {
        await driver.tableClient.withSession( async (session) => {
            await editSimple(table_name, edit_mode, table_col, session);
        });
    }

    exports.editSimpleByName = editSimpleByName;

    function      selectSimpleByName(table_name,      table_col_name,
table_col_value) {
        return      driver.tableClient.withSession(      (session)      =>
selectSimple(table_name, table_col_name, table_col_value, session));
    }

    exports.selectSimpleByName = selectSimpleByName;

    async  function  deleteSimpleByName(table_name,  table_col_name,
table_col_value) {
        await driver.tableClient.withSession( async (session) => {
            console.log(`Delete in table ${table_name}...`);
            await  deleteSimple(table_name,  table_col_name,  table_col_value,
session);
            console.log('Delete completed. ');
        });
    }

    exports.deleteSimpleByName = deleteSimpleByName;

    exports.options = [{
        key: 'serviceAccountKeyFile',

```

```

    name: 'service-account-key-file',
    description: 'service account key file for YDB authenticate',
  }];
5) db/utils.js
"use strict";
let __importDefault = (this && this.__importDefault) || function (mod) {
  return (mod && mod.__esModule) ? mod : { "default": mod };
};
Object.defineProperty(exports, "__esModule", { value: true });
exports.SYNTAX_V1 = exports.main = void 0;
const ydbSDK = require("ydb-sdk");
const yargs = __importDefault(require("yargs"));
async function main(runner, options) {
  const optionsUsage = options && options.length > 0 ?
options.map((option) => `--${option.name}`).join(") : ";
  const argsBuilder = yargs.default
    .usage(`Usage: $0 (--db <database> --endpoint <endpoint> or --
connection-string <connection_string>)${optionsUsage}`);
  argsBuilder.options({
    'db': { describe: 'YDB database name', type: 'string' },
    'endpoint': { describe: 'YDB database endpoint', type: 'string' },
    'connection-string': { describe: 'YDB connection string', type: 'string' },
  });
  options === null || options === void 0 ? void 0 : options.forEach((option)
=> {
    argsBuilder.option(option.name, {
      describe: option.description,
      type: 'string',
      demandOption: true,
    });
  });

```



```

    });
    const args = argsBuilder.argv;
    const endpointParam = args.endpoint;
    const dbParam = args.db;
    const connectionStringParam = args.connectionString;
    let endpoint;
    let db;
    if (connectionStringParam) {
        const parsedConnectionString =
ydbSDK.parseConnectionString(connectionStringParam);
        endpoint = parsedConnectionString.endpoint;
        db = parsedConnectionString.database;
    }
    else if (endpointParam && dbParam) {
        endpoint = endpointParam;
        db = dbParam;
    }
    else {
        throw new Error('Either --connection-string <connection_string> or --db
<database> --endpoint <endpoint> arguments are required');
    }
    const cliParams = {};
    options === null || options === void 0 ? void 0 : options.forEach((option)
=> {
        cliParams[option.key] = args[option.key];
    });
    const logger = ydbSDK.getLogger();
    logger.info(`Running basic-example script against endpoint '${endpoint}'
and database '${db}'.`);
    try {

```

```

    await runner(logger, endpoint, db, cliParams);
  }
  catch (error) {
    logger.error(error);
  }
}

exports.main = main;
exports.SYNTAX_V1 = '--!syntax_v1';

```

6) views/index.ejs

```

<!DOCTYPE html>

<html>

<head>

  <title><%= title %></title>

  <!-- <link rel='stylesheet' href='public/stylesheets/style.css' />-->

  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"

    integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">

  </head>

  <body>

    <div class='container mt-3'>

      <div class=" col-12 mb-3">

        <div class="d-flex align-items-center ">

          <div class="mr-3 safari-logo-width ">

          </div>

          <h1 class='pl-2'><%= title %></h1>

```

</div>

</div>

<h5 class='mb-1'>Добро пожаловать на сайт "<%= title %>"!</h5>

<h5 class='mb-4'>Выберите одну из таблиц, доступных для редакции,
ниже:</h5>

<h4 class='mb-3 text-center'>Доступные для редактирования
таблицы:</h4>

<div class='row justify-content-center mx-auto mb-3'>

<div class="card col-3 me-3">

<div class="card-body">

<h5 class="card-title">Личности</h5>

<p class="card-text">Таблица хранит данные о биометрических
показателях пользователей и других данных, обрабатываемых искусственным
интеллектом.</p>

Редактировать

</div>

</div>

<div class="card col-3 me-3">

<div class="card-body">

<h5 class="card-title">Местоположения</h5>

<p class="card-text">Таблица хранит данные о всех известных
местоположениях камер, их названиях, адресах и координатах.</p>

Редактировать

</div>

</div>

<div class="card col-3 me-3">

```

<div class="card-body">
  <h5 class="card-title">Виды активности</h5>
  <p class="card-text">Таблица хранит данные о всех видах
активности, которые может определить ИИ на камере, их названиях и
прочем.</p>
  <br>
  <a href="#" class="btn btn-primary">Редактировать</a>
</div>
</div>
<div class='row justify-content-center mx-auto mb-3'>
  <div class="card col-3 me-3">
    <div class="card-body">
      <h5 class="card-title">Опасные объекты</h5>
      <p class="card-text">Таблица хранит данные о опасных
объектах, нарушениях устава и прочем, что ИИ может посчитать
недопустимым по определенным параметрам.</p>
      <a href="#" class="btn btn-primary">Редактировать</a>
    </div>
  </div>
  <div class="card col-3 me-3">
    <div class="card-body">
      <h5 class="card-title">Пользователи</h5>
      <p class="card-text">Таблица хранит личные данные
пользователей.</p>
      <br><br><br>
      <a href="/users" class="btn btn-primary">Редактировать</a>
    </div>
  </div>
</div>
<div class="card col-3 me-3">

```

```

    <div class="card-body">
      <h5 class="card-title">СКУД</h5>
      <p class="card-text">Таблица хранит о всех входах и выходах
(дата/время), осуществляемых по электронным пропускам.</p>
      <br>
      <a href="#" class="btn btn-primary">Редактировать</a>
    </div>
  </div>
</div>
<br><br>
<hr>
<p>Copyright (c) 2022 | Коновалов Никита Александрович</p>
</div>
</body>
</html>

```

7) views/users/index.hbs

```

<!DOCTYPE html>
<html lang='ru'>
<head>
  <title>{{title}}</title>
  <meta charset="utf-8" />
  <!-- <link rel='stylesheet' href='/public/stylesheets/style.css' />-->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-
1BmE4kWBq78iYhF1dvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
</head>
<body>

```

```

<div class='container mt-3'>
  <div class=" col-12 mb-3">
    <div class="d-flex align-items-center ">
      <div class="mr-3 safari-logo-width ">
        
      </div>
      <h1 class='pl-2'>{{title}}</h1>
    </div>
  </div>
  <p><a href="/">На главную</a> | <a href="/users/create">Добавить
пользователя</a></p>
  <table class="table">
    <thead>
      <tr>
        <th scope="col">Идентификатор пользователя</th>
        <th scope="col">Идентификатор эл. пропуска</th>
        <th scope="col">Идентификатор зачетной книжки</th>
        <th scope="col">Идентификатор работника</th>
        <th scope="col">Паспорт</th>
        <th scope="col">Биометрические данные</th>
        <th scope="col">ФИО</th>
        <th scope="col">Дата рождения</th>
        <th scope="col">Пол</th>
        <th scope="col"></th>
      </tr>
    </thead>
    <tbody>
      {{#each users}}
        <tr>

```

```

<th scope="row">{{this.idUser}}</th>
<td>{{this.idPass}}</td>
<td>{{this.idGradebook}}</td>
<td>{{this.idEmployee}}</td>
<td>{{this.passport}}</td>
<td>{{this.personalityData}}</td>
<td>{{this.fio}}</td>
<td>{{formatTime this.birthDate "YYYY-MM-DD"}}</td>
<td>{{this.gender}}</td>
<td>
<a class="btn btn-success btn-sm mb-1"
href="/users/edit/{{this.idUser}}" role="button">✎</a>
<br>
<form action="/users/delete/{{this.idUser}}" method="POST"
style="display:inline;">
<input class="btn btn-danger btn-sm" type="submit"
value="✕" />
</form>
</td>
</tr>
{{/each}}
</tbody>
</table>
</div>
</body>
</html>

```

8) views/users/create.hbs

```

<!DOCTYPE html>
<head>

```

```

<title>{{title}}</title>
<meta charset="utf-8" />
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jI
W3" crossorigin="anonymous">
</head>
<body>
<div class="container">
  <br>
  <h1>{{title}}</h1>
  <br>
  <form method="POST">
    <div class="input-group flex-nowrap mb-3">
      <span      class="input-group-text      col-3"      id="addon-
wrapping">Идентификатор пользователя:</span>
      <input  required  name="idUser"  type="number"  class="form-
control">
    </div>
    <div class="input-group flex-nowrap mb-3">
      <span      class="input-group-text      col-3"      id="addon-
wrapping">Идентификатор эл. пропуска:</span>
      <input  required  name="idPass"  type="number"  class="form-
control">
    </div>
    <div class="input-group flex-nowrap mb-3">
      <span      class="input-group-text      col-3"      id="addon-
wrapping">Идентификатор зачетной книжки:</span>

```



```

        <input required name="idGradebook" type="number" class="form-
control">
    </div>
    <div class="input-group flex-nowrap mb-3">
        <span class="input-group-text col-3" id="addon-
wrapping">Идентификатор работника:</span>
        <input required name="idEmployee" type="number" class="form-
control">
    </div>
    <div class="input-group flex-nowrap mb-3">
        <span class="input-group-text col-3" id="addon-
wrapping">Паспорт:</span>
        <input required name="passport" type="text" class="form-control"
placeholder="0000 000000">
    </div>
    <div class="input-group flex-nowrap mb-3">
        <span class="input-group-text col-3" id="addon-
wrapping">Биометрические данные:</span>
        <input required name="personalityData" type="text" class="form-
control">
    </div>
    <div class="input-group flex-nowrap mb-3">
        <span class="input-group-text col-3" id="addon-
wrapping">ФИО:</span>
        <input required name="fio" type="text" class="form-control"
placeholder="Иванов Иван Иванович">
    </div>
    <div class="input-group flex-nowrap mb-3">
        <span class="input-group-text col-3" id="addon-wrapping">Дата
рождения:</span>

```

```

        <input required name="birthDate" type="date" class="form-control"
placeholder="Username">
    </div>
    <div class="input-group mb-3">
        <label class="input-group-text col-3">Пол:</label>
        <select required name="gender" class="form-select">
            <option selected value="М">Мужской</option>
            <option value="Ж">Женский</option>
        </select>
    </div>
    <br><br>
    <input class="form-control" type="submit" value="Отправить" />
</form>
<br>
<a href="/users">К списку пользователей</a>
</div>
</body>
<html>

```

ПРИЛОЖЕНИЕ Б – Презентация



Рисунок Б.1 – Первый слайд

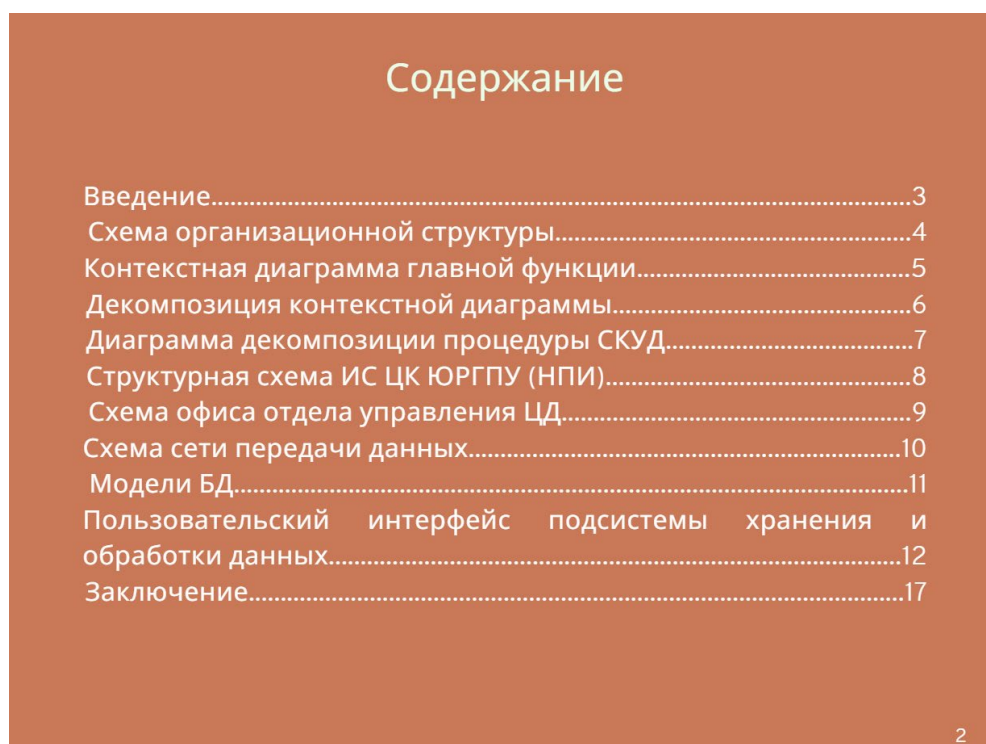


Рисунок Б.2 – Второй слайд

Введение

Постепенно, интернет трансформируется из громадной игрушки для развитых интеллектуальных личностей в полноценный источник разнообразной и одновременно полезной информации для любого слоя населения планеты Земля, поэтому он стал наиболее эффективным средством продвижения и обмена информацией и является одним из значительных элементов современной цивилизации. Я считаю, что поэтому одной из популярных тем на сегодняшний день является создание и поддержание сайтов.

Они, то есть сайты, позволяют обрабатывать, передавать, преобразовывать, хранить, продавать различные типы информации. И для всего этого не нужно отходить от экрана компьютера. Наиболее важным является то, что сайты позволяют подать эту информацию в красочном и отчетливом виде, снабдить анимацией, видеороликами, графикой, звуком, ссылками. Я уверен, что при правильном планировании выполняемой работы, возможно продемонстрировать эту правильно поданную информацию тысячам других пользователей интернета.

Web-сайт цифрового двойника ЮРГПУ (НПИ) будет создан для того, чтобы через сеть Интернет обеспечивает доступ пользователям к оперативной информации, которая происходит в университете. Доступ к информации требует регистрации пользователя, так как данная информация является не общедоступной.

В настоящей дипломной работе представлен Web-сайт подсистемы хранения цифрового двойника ЮРГПУ (НПИ) им. М.И. Платова, который взаимодействует с подключенной к нему базой данных Yandex Database.

3

Рисунок Б.3 – Третий слайд

Схема организационной структуры



4

Рисунок Б.4 – Четвертый слайд

Контекстная диаграмма главной функции

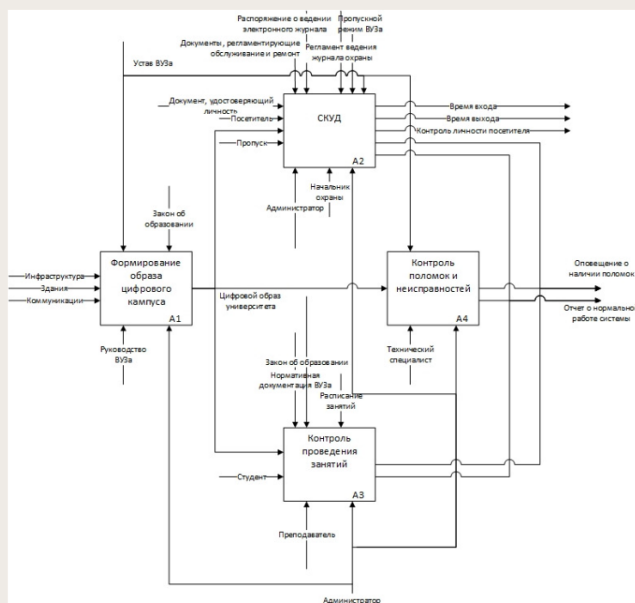


Для описания бизнес-процессов используются диаграммы IDEF0.
Модель в IDEF0 представлена совокупностью иерархически упорядоченных и логически связанных диаграмм.

5

Рисунок Б.5 – Пятый слайд

Декомпозиция контекстной диаграммы

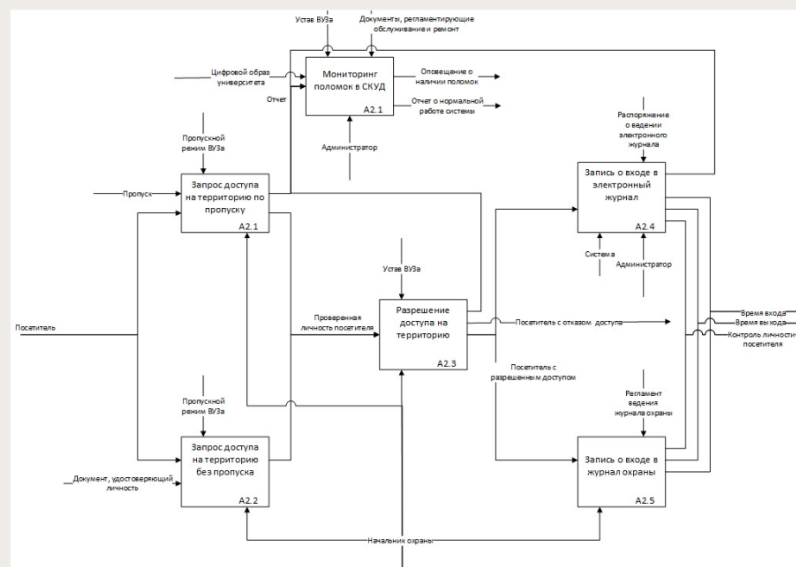


В ходе декомпозиции контекстной диаграммы, должны быть получены основные функции цифрового двойника ЮРГПУ(НПИ)

6

Рисунок Б.6 – Шестой слайд

Диаграмма декомпозиции процедуры "СКУД"



В ходе выполнения декомпозиции процесса "СКУД" были выявлены его подфункции

7

Рисунок Б.7 – Седьмой слайд

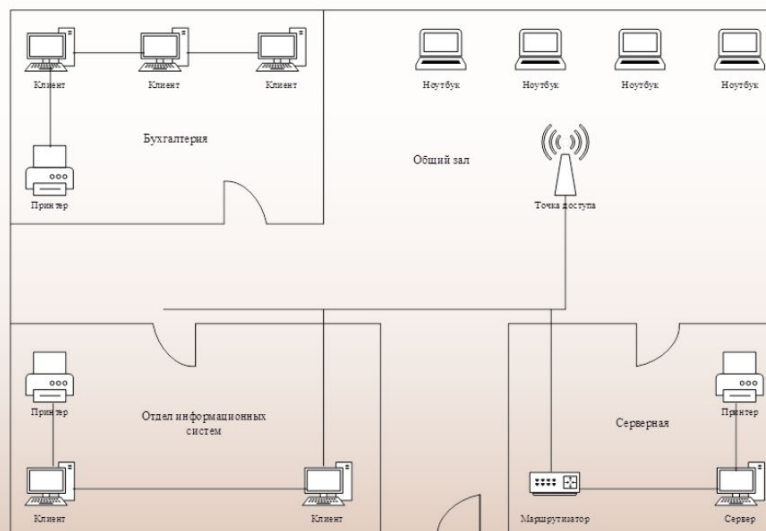
Структурная схема информационной системы цифрового двойника ЮРГПУ (НПИ)



8

Рисунок Б.8 – Восьмой слайд

Схема офиса отдела управления ЦД



9

Рисунок Б.9 – Девятый слайд

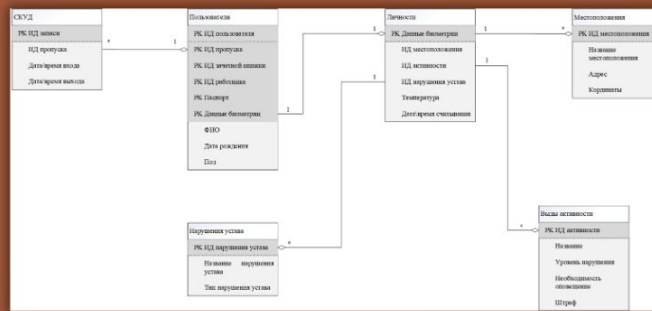
Схема сети передачи данных



10

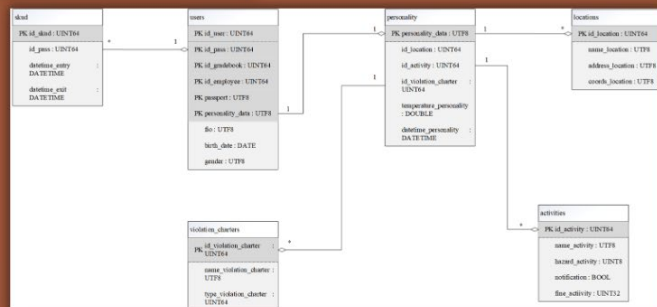
Рисунок Б.10 – Десятый слайд

Модели БД



- Концептуальная модель

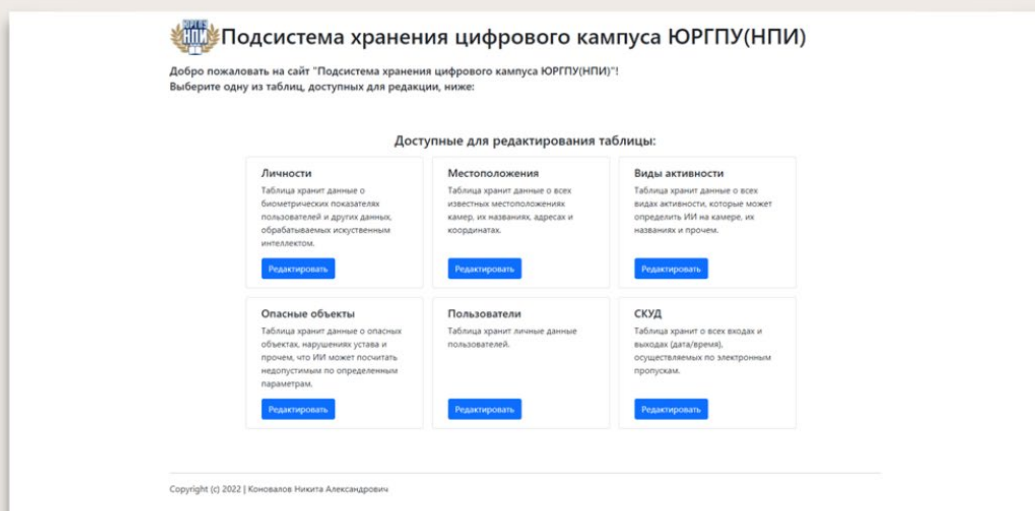
Физическая модель -



11

Рисунок Б.11 – Одинадцатый слайд

Пользовательский интерфейс подсистемы хранения и обработки данных




Главная страница подсистемы

12

Рисунок Б.12 – Двенадцатый слайд

Пользовательский интерфейс подсистемы хранения и обработки данных

 **Список пользователей**

[На главную](#) | [Добавить пользователя](#)

Идентификатор пользователя	Идентификатор зл. пропуска	Идентификатор зачетной книжки	Идентификатор работника	Паспорт	Биометрические данные	ФИО	Дата рождения	Пол
1	150833946	1832123	13021	3225 764451	32838118972656	Абрамов Николай Федорович	1996-10-22	М
2	100833333	0	14231	6021 114451	51116387928887	Анисимова Анастасия Александровна	1970-03-11	Ж
3	150833945	1842223	0	9021 212333	44942411679400	Кичилюрова Георгий Михайлович	1996-11-22	М
4	112833232	1123423	0	3212 112452	35210933387152	Полесова Мария Валентиновна	1998-12-23	Ж
5	151233453	2123623	0	5652 112452	40057957230489	Георова Тамара Петровна	2001-07-22	Ж

Страница таблицы "Пользователи"

13

Рисунок Б.13 – Тринадцатый слайд

Пользовательский интерфейс подсистемы хранения и обработки данных

Добавление пользователя

Идентификатор пользователя:

Идентификатор зл. пропуска:

Идентификатор зачетной книжки:

Идентификатор работника:

Паспорт:

Биометрические данные:

ФИО:

Дата рождения:

Пол:

[К списку пользователей](#)

Страница "Добавление пользователя" для таблицы "Пользователи"

14

Рисунок Б.14 – Четырнадцатый слайд

Пользовательский интерфейс подсистемы хранения и обработки данных

Редактирование пользователя

Идентификатор пользователя:	4
Идентификатор з/л пропуска:	112833232
Идентификатор зачетной книжки:	1123423
Идентификатор работника:	0
Паспорт:	3212 112452
Биометрические данные:	3521093387152
ФИО:	Полокова Мария Валентиновна
Дата рождения:	23.12.1998
Пол:	Женский


[К списку пользователей](#)

Страница "Редактирование пользователя" для таблицы "Пользователи"






15

Рисунок Б.15 – Пятнадцатый слайд

Пользовательский интерфейс подсистемы хранения и обработки данных

 **Список пользователей**

[на главную](#) | [добавить пользователя](#)

Идентификатор пользователя	Идентификатор з/л пропуска	Идентификатор зачетной книжки	Идентификатор работника	Паспорт	Биометрические данные	ФИО	Дата рождения	Пол	
1	150833946	1832123	13021	3225 764451	32838118972656	Абрамов Николай Федорович	1996-10-22	М	
2	100823333	0	14231	6021 114451	51116387928887	Алексимова Анастасия Александровна	1970-03-11	Ж	
3	150833945	1842223	0	9021 212333	44942411679400	Клиширов Георгий Михайлович	1996-11-22	М	
5	151233453	2123623	0	5652 112452	40057957250489	Георова Тамара Петровна	2001-07-22	Ж	
100	100	100	100	6060 100100	32833100100100	Коньвалов Никита Александрович	2000-10-20	М	

Страница таблицы "Пользователи" после добавления и удаления одной записи

16

Рисунок Б.16 – Шестнадцатый слайд

Заключение

В рамках дипломной работы были выполнены следующие задачи:

- проведен анализ предметной области;
- проведен анализ функций организационной структуры;
- построены диаграммы описания бизнес-процессов в нотации *IDEFO* и диаграмма потоков данных в нотации *DFD*;
- разработана концепция и архитектура построения ИС;
- разработаны концептуальная, логическая и физическая модели БД;
- спроектирована подсистема хранения и обработки данных цифрового двойника ЮРГПУ(НПИ) им. М.И. Платова;
- разработан прототип веб-сайта подсистемы хранения и обработки данных цифрового двойника ЮРГПУ(НПИ) им. М.И. Платова.

17

Рисунок Б.17 – Семнадцатый слайд

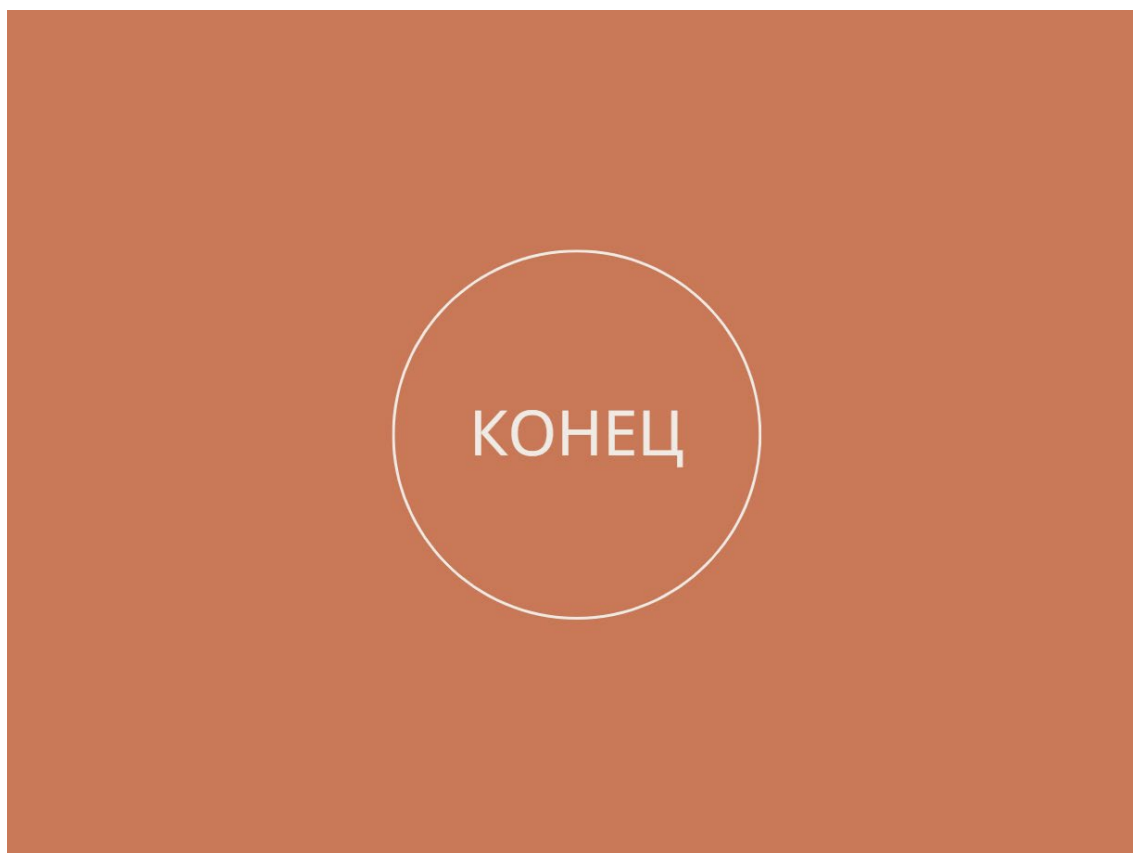


Рисунок Б.18 – Восемнадцатый