

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Специальность:

09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»

Специализация:

«Эксплуатация вычислительных машин,
комплексов, систем и сетей специального
назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

**ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС ДЛЯ
ОБРАБОТКИ И ХРАНЕНИЯ ВИДЕОКОНТЕНТА**

Студент А.И. 6.06.2023 Маслов Артем Юрьевич
(подпись, дата) (ФИО полностью)

Руководитель Д.В. Дубравин А.В.
(подпись, дата) (фамилия, инициалы) 7.06.2023

Нормоконтролёр А.С. Бычков А.С.
(подпись, дата) (фамилия, инициалы) 7.06.2023

Рецензент А.С. Васильев Васильев А.В.
(должность, место работы) (подпись, дата) (фамилия,
инициалы) «ИПП» «Р.Д.И.»

Работа допущена к защите (протокол зас. кафедры от 2.06. 2023 г. № 14)

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

Работа защищена с отметкой удовлетворительно
(протокол заседания ГЭК от 19.06.2023 № 5)

Секретарь ГЭК

Е.
(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

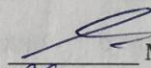
Факультет

Кафедра

Вычислительной техники

Вычислительная техника

«Утверждаю»
Заведующий кафедрой


Митрохин М.А.
«26» апреля 2023 г.

ЗАДАНИЕ НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Маслов Артем Юрьевич гр. 18ВОЭ1 факультета вычислительной техники.

2. Тема работы: «Программно-аппаратный комплекс для обработки и хранения видеоконтента».

Тема утверждена приказом ПГУ № 458/о от 26.04.2023 г.

3. Руководитель работы к.т.н., доцент каф. ВТ «ПГУ» Дубравин А.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

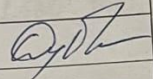
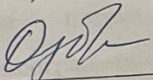
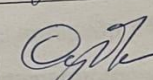
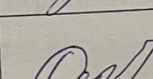
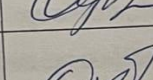
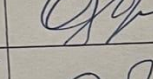
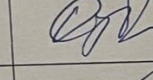
Назначение разработки – Программно аппаратный комплекс предназначен для формирования файлов видеоконтента на внутреннем хранилище видео регистратора с последующей передачей видеоконтента в основное облачное хранилище. В разработке необходимо предусмотреть кроссплатформенное приложение для работы с устройствами и созданными файлами видеоконтента. Система должна работать в рамках локальной и глобальной сети передачи данных.

5. Перечень подлежащих разработке вопросов.

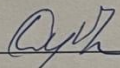
Требуется произвести анализ предметной области, определить функциональный состав программно-аппаратного комплекса, выбрать стек технологий для реализации комплекса, разработать алгоритм работы программной части, разработать UML диаграммы с последующим

написанием программы и формирования дистрибутива встроенной операционной системы для корректной работы частей системы в едином комплексе.

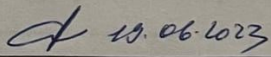
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объ ем рабо ты	Срок выполне ния	Подпись руководит еля
1.	Анализ предметной области	10	26.04.2023 - 28.04.2023	
2.	Выбор частей системы для реализации комплекса	10	29.04.2023 - 30.04.2023	
3.	Изучение работы ОС Linux Embedded	20	31.04.2023 - 03.05.2023	
4.	Изучение работы облачных хранилищ видео контента	10	04.05.2023 - 08.05.2023	
5.	Проектирование и реализация программной части комплекса	10	09.05.2023 - 13.05.2023	
6.	Сбор единой системы и ее функциональное тестирование	10	16.05.2023 - 23.05.2023	
7.	Оформление выпускной работы	30	24.05.2023 - 02.06.2023	

Дата выдачи задания «26» 04 2023 года.

Руководитель дипломного проекта  Дубравин А.В.
(подпись, дата)

Задание к исполнению принял студент  Маслов А.Ю.
(подпись, дата)

Работу к защите допустить
Декан факультета  10.06.2023 Фионова Л.Р.
(подпись, дата)

Реферат

Работа содержит 105 листа, 33 рисунков, 18 таблиц, 5 источников, 4 приложения

ПРОГРАММА, ВСТРОЕННАЯ ОПЕРАЦИОННАЯ СИСТЕМА, СЕТЕВОЙ ВИДЕОРЕГИСТРАТОР, WEB СТРАНИЦА

Объектом разработки является программно-аппаратный комплекс, обеспечивающий видеофиксацию с помощью камер видео наблюдения, с сохранением видео контента на SSD накопитель. Программная часть комплекса представляет собой WEB приложение на базе стека LAMP. WEB приложение позволяет: смотреть, удалять видео контент.

В результате проведённой работы разработан проект, удовлетворяющий поставленной задаче, а именно, создан программно-аппаратный комплекс, в состав которого вошёл сетевой видеорегистратор на базе x86 архитектуры, на базе которого развернута встроенная операционная система и LAMP сервер для записи видеоконтента с видео камер.

Данный проект рекомендуется использовать для организации мобильных систем видеонаблюдения за трудовой деятельностью сотрудников.

					ПГУ1.090501.07.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программно-аппаратный комплекс для обработки и хранения видеоконтента	Лит.	Лист	Листов
Разраб.		Маслов А.Ю.						
Провер.		Дубравин А.В.					2	105
Н. Контр.		Бычков А.С.				ФВТ, каф. ВТ		
Утверд.		Митрохин М.А.				гр.18ВОЭ1		
					Пояснительная записка			

(подпись, дата)

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Программно-аппаратный комплекс для обработки и хранения видеоконтента	8
1.1 Предпроектный анализ предметной области.....	8
1.1.1 Анализ предметной области	8
1.1.2 Техническое задание на разработку программного обеспечения	13
1.1.2.1 Основание для разработки.....	13
1.1.2.2 Назначение разработки	13
1.1.2.3 Требования к программе.....	13
1.1.2.4 Требования к надежности.....	14
1.1.2.5 Требования к составу и параметрам технических средств	14
1.1.2.6 Стадии и этапы разработки	14
1.1.3 Выбор стека технологий и средств разработки.....	15
1.1.3.1 Выбор аппаратной платформы.....	15
1.1.3.2 Выбор версии ОС Linux Embedded.....	20
1.1.3.3 Выбор языка программирования для реализации приложения.....	21
1.1.3.4 Выбор фреймворка для реализации приложения.....	22
1.1.4 Выводы	23
1.2 Реализация программно-аппаратного комплекса для обработки и хранения видео контента	24
1.2.1 Реализация аппаратной части проекта.....	24
1.2.1.1 Инсталляция ОС Linux и периферийного оборудования	25
1.2.2 Реализация программного приложения	31
1.2.2.1 Разработка базы данных	32
1.2.2.2 Разработка серверной логики приложения.....	40
1.2.2.3 Разработка интерфейсов клиентской части приложения	43
1.2.3 Выводы	51
2 Технология установки и настройки сервера LAMP для комплекса обработки и хранения видео контента.....	52

					ПГУ1.09.05.01.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

2.1 Установка сервера Apache	52
2.2 Установка MySQL	54
2.3 Установка Python	55
2.4 Выводы по разделу	55
3 Экономическое обоснование разработки	56
4 Безопасность жизнедеятельности. Эргономические требования к рабочему месту программиста.....	62
Заключение	68
Список использованных источников	69
Приложение А - UML - диаграммы	70
Приложение А.1 – Диаграмма развертывания.....	71
Приложение А.2 – Диаграмма развертывания.....	72
Приложение А.3 – Диаграмма развертывания.....	73
Приложение А.4 – Диаграмма последовательности	74
Приложение А.5 – Диаграмма последовательности	75
Приложение Б - Физическая модель базы данных	786
Приложение В - Листинг приложения.....	78
Приложение Г -Презентация.....	96

					ПГУ1.09.05.01.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ПЗ – пояснительная записка

ПО – программное обеспечение

СУБД – система управления базами данных

API (Application Programming Interface) – программный интерфейс приложения

WSGI(Web Server Gateway Interface) - стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб сервером

WEB - система доступа к связанным между собой документам на различных компьютерах, подключённых к Интернету

LAMP - акроним, обозначающий набор (комплекс) серверного программного обеспечения

IP - это уникальный адрес, идентифицирующий устройство в интернете или локальной сети

SoC - Security Operations Center

UML (Unified Modeling Language) – унифицированный язык моделирования

					ПГУ1.09.05.01.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В настоящий момент существует огромное количество девелоперских проектов требующих использования огромного количества рабочих различных специальностей и направлений. Даже для реализации ремонтных работ одной квартиры требуется порядка 25 человек различных специальностей и три месяца работы. Помимо самой работы каждый этап требует надзора и приемки выполненных работ.

И как показывает практика, на каждом этапе возникают вопросы, требующие вмешательства руководителя бригады, проектного менеджера, надзорных органов и представителя заказчика.

В связи с тем, что процесс ремонта или иного вида девелоперского проекта растянут во времени основные и спорные моменты фиксируются в виде актов, и как всегда не хватает видео фиксации фактов, которые доказывают правоту той или иной стороны процесса девелопмента.

Иногда крупные девелоперы и смежные исполнители прибегают к организации стационарного наблюдения, с подключением облачной системы видео наблюдения, используя услуги провайдеров облачных видео регистраторов: АО «Эр-Телеком» или ПАО «Ростелеком».

В связи с ограниченной статьей бюджета средние девелоперы не могут пойти на данный шаг, и вынуждены фиксировать свою работу на видео камеру и активировать деятельность в присутствии заказчика или иных законных представителей.

Интернет исследования показывают потребность в альтернативе дорогим стационарным или облачным системам видеонаблюдения.

Целью данного дипломного проекта является разработка мобильной бюджетной системы видеонаблюдения, которая позволит сотрудникам среднего и малого девелопмента:

- быстро развернуть систему на время реализации проекта;
- на всем протяжении проекта проводить видео фиксацию работ;

					ПГУ1.09.05.01.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

- иметь возможность просмотра архива видео фиксации;
- по окончании работ демонтировать систему и развернуть ее на новом объекте.

Дипломный проект программно-аппаратного комплекса для обработки и хранения видео контента реализован на базе классической трехзвенной архитектуры:

- интерфейсом данного комплекта является *web* приложение, которое отображает записи видео фиксации и позволяет их просмотр;
- ядром данного комплекса является компактный видео сервер, с Ethernet портами и возможностью подключения *SSD* хранилища;
- верхним слоем ядра является *ОС Linux Embedded* с возможностью развертывания сервера *LAMP*;
- хранилищем видео файлов является *SSD* накопитель с функциями: чтение, запись, удаление файлов;
- фиксация видео осуществляется *IP* видео камерами, подключенными к периферийным устройствам *IP* видео сервера.

Дипломный проект состоит четырёх разделов:

- первый раздел включает в себя анализ предметной области и стека технологий для реализации проекта; описывает реализацию ядра системы, включая интеграцию *ОС Linux Embedded* в аппаратную часть проекта; реализацию *web* приложения с возможностями отображения списка видео записей;
- второй раздел посвящён технологии отладки *web* приложения;
- в третьем разделе показано экономическое обоснование проекта;
- четвёртый раздел посвящён безопасности жизнедеятельности оператора ПК.

					ПГУ1.09.05.01.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

1 Программно-аппаратный комплекс для обработки и хранения видеоконтента

1.1 Предпроектный анализ предметной области

1.1.1 Анализ предметной области

Современная бюджетная система видео наблюдения состоит из следующих компонентов (рисунок 1.1):

- видео регистратор с жестким диском;
- набор видеокамер;
- сетевой коммутатор;
- облачная коммутационная среда;
- устройства воспроизведения видео файлов или потока (мобильный телефон или компьютер).

Самым дорогим компонентом данной системы является видео регистратор, более того видео регистратор произведенный сторонней организацией является «черным ящиком» для технического персонала производящего монтаж системы (рисунок 1.1).

В связи с вышеизложенным, возникла потребность в производстве не готовых решений «под ключ», а гибких программно-аппаратных комплексов для обработки и хранения видео контента, с возможностью замены функционала видео регистратора, программы воспроизведения видео файлов и самой операционной системы.

Для этого, взамен классического видео регистратора представленного на базе архитектуры x86 предлагается использовать SoC системы и предлагать конечным пользователям (техническим специалистам по монтажу и настройке видео систем) не только готовые решения, но и программные модули с функционалом, требуемым для конкретного

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

проекта. Данный подход делает реализуемый комплекс «гибким», бюджетным техническим решением.



Рисунок 1.1 – Классическая бюджетная система видеонаблюдения

На этапе анализа предметной области требуется рассмотреть аналогичные решения и выявить их достоинства и недостатки, которые будут учтены при реализации дипломного проекта.

В таблице 1.1 представлены существующие программно-аппаратные комплексы для обработки и хранения видео контента с функцией видеонаблюдения.

Анализ существующих решений показал, что основным ограничением данных комплексов является то, что они предлагают готовые решения, изменить логику работы которых может только производитель или специализированный центр. Логикой такого решения является максимальная коммерциализация проекта.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 1.1 – Сравнительные характеристики аналогов

Наименование системы	Плюсы	Минусы
комплекс <i>Okta Vision</i>	<ul style="list-style-type: none"> -имеет полную комплектность; -высокая скорость обработки видео контента; -наличие облачного хранилища и сервера обработки данных; -техническая поддержка по требованию. 	<ul style="list-style-type: none"> -реализован на x86 архитектуре; -проприетарное программное обеспечение; -облачное хранилище платное.
комплекс <i>HD-TVI Hiwatch</i>	<ul style="list-style-type: none"> -имеет полную комплектность; -средняя скорость обработки видео контента; -наличие облачного хранилища данных; -техническая поддержка по требованию. 	<ul style="list-style-type: none"> -реализован на SoC архитектуре с защищенной от изменений логикой; -проприетарное программное обеспечение.
комплекс <i>Tantos Amelie</i>	<ul style="list-style-type: none"> -имеет полную комплектность; -высокая скорость обработки видео контента; -техническая поддержка 5/12/365. 	<ul style="list-style-type: none"> -реализован на SoC архитектуре с защищенной от изменений логикой; -проприетарное программное обеспечение.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Целью дипломного проекта является разработка бюджетного варианта программно-аппаратного комплекса для обработки и хранения видео контента с функцией видеонаблюдения на базе открытых свободно распространяемых решений. В результате конечные потребители и технические специалисты получают готовое работоспособное решение, в которое они могут добавить собственный функционал, изменив программное обеспечение системы самостоятельно или с помощью разработчиков системы на бесплатной основе.

Схема реализуемого комплекта представлена на рисунке 1.2 в виде диаграммы развертывания.

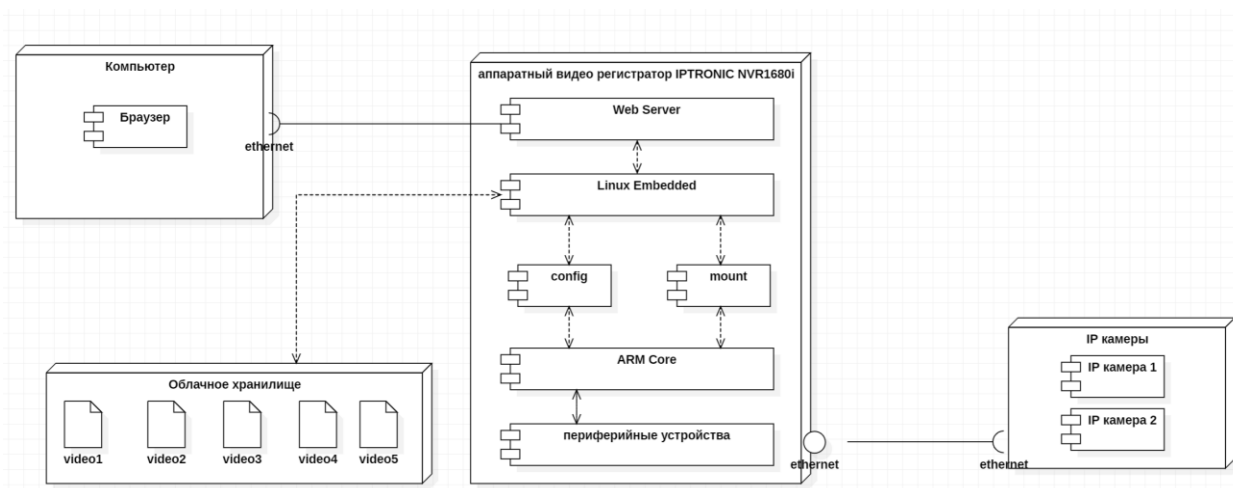


Рисунок 1.2 – Программно-аппаратного комплекс для обработки и хранения видеоконтента (диаграмма развертывания)

Все представленные на рисунке 1.2 аппаратные и программные части являются или бесплатными или представляют собой бюджетный вариант аппаратуры находящейся в свободном доступе. Более того программирование и настройка аппаратных частей комплекса имеет подробную мультязычную инструкцию.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Таким образом, задача разработки программно-аппаратного комплекса для обработки и хранения видео контента, решаемая путем разработки новой системы, с учетом недостатков уже существующих систем является актуальной.

1.1.2 Техническое задание на разработку программного обеспечения

1.1.2.1 Основание для разработки

Основанием для разработки является задание на дипломное проектирование.

1.1.2.2 Назначение разработки

Программно-аппаратный комплекс для обработки и хранения видеоконтента с функцией видео наблюдения предназначен для организации мобильного поста видеонаблюдения при проведении строительных и отделочных работ. Система должна иметь клиентсерверную архитектуру. Клиентская часть отвечает за пользовательский интерфейс и реализована в виде *WEB* приложения, серверная часть реализована в виде программно аппаратной платформы на базе серверной платформы *Proto-X*.

1.1.2.3 Требования к программе

Клиентская часть программно-аппаратного комплекса должна обеспечивать выполнение следующих функций:

- формирование списка файлов видео фиксации;
- возможность просмотра файлов видео фиксации в браузере;
- возможность удаления файла видео фиксации.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Серверная часть программно-аппаратного комплекса должна обеспечивать выполнение следующих функций:

- сбор видео трафика с камер по протоколу *Ethernet*;
- сохранение архива видео файлов на *SSD* накопителе;
- наличие *LAMP* сервера для доступа к архиву файлов с помощью технологии «тонкий» клиент.
- Входными данными являются:
- видео трафик с камер видео наблюдения;
- архив видео файлов на *SSD* накопителе.

1.1.2.4 Требования к надежности

При функционировании клиентской части системы должен осуществляться контроль входной и выходной информации. В случае возникновения ошибки должны выдаваться соответствующие сообщения и предоставляться возможность корректного выхода из программы.

При функционировании серверной части и периферийных устройств системы должен осуществляться контроль входного и выходного потоков информации. В случае возникновения ошибки на программном и/или аппаратном уровне должны выдаваться соответствующие сообщения и предоставляться возможность корректного перезапуска системы.

1.1.2.5 Требования к составу и параметрам технических средств

Клиентское приложение предназначено для работы на *IBM*-совместимых персональных компьютерах с минимальной конфигурацией: процессор *Intel Core 2 Duo 2.0 ГГц*, оперативная память 8 Гбайт, жесткий диск 500 Мбайт, монитор.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Серверное приложение предназначено для работы на платформе IPTRONIC NVR1680i или аналог, с установленной *OC Linux Embedded*.

1.1.2.6 Стадии и этапы разработки

Таблица 1.2- Стадии и этапы разработки

Наименование этапов работы	Объем работы	Срок выполнения
Анализ предметной области	10%	
Составление технического задания	4%	
Выбор стека технологий для реализации проекта	10%	
Проектирование системы	10%	
Разработка серверной части	30%	
Разработка клиентской части	20%	
Выполнение задания по экономике	6%	
Выполнение задания по БЖД	6%	
Оформление пояснительной записки	4%	

1.1.3 Выбор стека технологий и средств разработки

1.1.3.1 Выбор аппаратной платформы

Для реализации проекта требуется выбрать основной и центральный компонент системы – аппаратную платформу с вычислительным ядром и периферией, на базе которой будет реализован проект.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

При выборе аппаратной платформы требуется принять принципиальное решение. Реализовать проект на базе ОС Linux Embedded или реализовать его полностью схемотехническим инструментарием .

В таблице 1.3 приведен сравнительный анализ данных подходов с обоснованием выбора способа на базе *ОС Linux Embedded*.

Таблица 1.3.- Сравнительный анализ способов реализации SoC проектов.

Критерий анализа	Способы реализации SoC проектов	
	Аппаратно ориентированные платформы (bare-metal)	Платформы на базе ОС Linux Embedded
Время разработки	Системы без операционной системы требуют больше времени на разработку, чем встроенные системы Linux, поскольку разработчикам приходится писать каждый компонент с нуля	Основаны на библиотеках и драйверах с открытым исходным кодом, которые можно использовать практически без изменений
Использование памяти	Системы на аппаратной платформе способны работать на микроконтроллерах с низким уровнем ресурсов	Проекты ресурсоемкие и требуют больше памяти, чем системы на аппаратной платформе
Гибкость разработки	Разрабатываются специально для конкретной задачи, что делает их негибкими	Обладают широкими возможностями настройки благодаря большому количеству доступных библиотек, что делает их адаптируемыми и гибкими для различных задач.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Критерий анализа	Способы реализации SoC проектов	
	Аппаратно ориентированные платформы (bare-metal)	Платформы на базе ОС Linux Embedded
Безопасность использования	Безопасность средствами аппаратуры или средствами производителя устройства	Используют преимущества встроенных функций безопасности и механизмов безопасности программного обеспечения VPN, брандмауэры и программные исправления.
Простота отладки	Сложнее отлаживать по сравнению со встроенными системами Linux	Разработчики имеют доступ к широкому спектру инструментов отладки и могут воспользоваться журналами ошибок и средствами трассировки.
Стоимость	Дешевле разрабатывать, так как требуют меньше памяти и меньше времени на разработку.	Дороже из-за более высоких требований к памяти и затрат на разработку.

Исходя из сравнительного анализа, приведенного в таблице 1.3 в связи с ограниченностью времени реализации проекта и отсутствием в команде инженера-системотехника выбор был сделан на выбор платформы на базе ОС *Linux Embedded*.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Следующим этапом выбора аппаратной платформы для проекта является выбор конкретного аппаратного решения с необходимыми функциями в сегменте бюджетных моделей (таблица 1.4).

Таблица 1.4- Сравнительный анализ плат SoC

Критерий выбора	IPTRONIC NVR1680i	J2000- NVR04mt	Smartec STNR-0441
Цена до 100 000 рублей за единицу	да	да	да
Поддержка обработки видео на аппаратной уровне	да	нет	да
Количество отсеков для жестких дисков	2	1	1
Встроенная работа с облачными решениями	да	нет	нет
Наличие <i>Ethernet</i>	да	да	да
Скорость <i>Ethernet</i> , Мбит/сек	1000	1000	100
Поддержка системы интернета вещей	да	нет	нет

Исходя из сравнительного анализа характеристик аппаратных платформ, приведенного в таблице 1.4, выбор сделан в пользу *IPTRONIC NVR1680i* (рисунок 1.3).

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Платформа представляет собой комбинацию аппаратного ядра, периферии и *OS Linux Embedded*. Данное сочетание позволит реализовать проект программно-аппаратный комплекс для обработки и хранения видео контента в полном объеме.



Рисунок 1.3 – Аппаратная часть проекта. Видео регистратор *IPTRONIC NVR1680i*

Ядром платформы являются вычислительная платформа, представленная многоядерным *ARM* процессором *ARM Cortex A7* (рисунок 1.4) и стек аппаратно-программной среды содержащей инструменты для аппаратной и программной реализации проекта.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

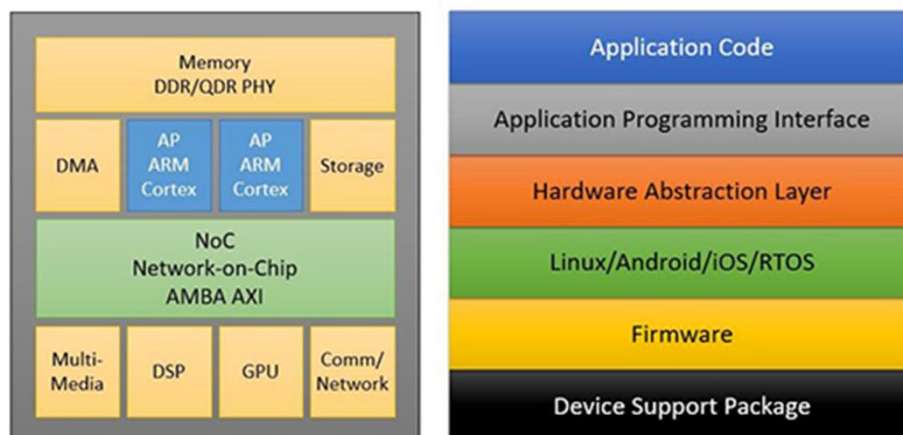


Рисунок 1.4 –Модуль ядра и стек среды *IPTRONIC NVR1680i*

1.1.3.2 Выбор версии *OC Linux Embedded*

Для реализации проекта требуется выбрать тип и версию операционной системы, которая является готовым к использованию дистрибутивом *Linux* на базе *Ubuntu* для *ARM*, с поддержкой периферии и *SSH* доступа к системе.

Выбор производится между двумя дистрибутивами *OC PetaLinux* и *OC Xillinux*, в связи с тем, что исполнитель имеет опыт реализации проектов на данных ОС.

Время разработки программного и аппаратного слоя системы: *OC PetaLinux* быстрый и эффективный инструмент разработки по сравнению с *OC Xillinux*. *OC PetaLinux* предлагает обширную документацию, руководства и учебные пособия, что сокращает время разработки по сравнению с *OC Xillinux*.

Настройка программного и аппаратного слоя системы: *OC PetaLinux* предоставляет больше возможностей для настройки, чем *OC Xillinux*. Разработчики могут настраивать *OC PetaLinux* и создавать свои собственные типы серверов или системы на основе приложений. *OC*

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Xillinux предоставляет меньше возможностей для настройки и более специализирован.

Поддерживаемые устройства: *OC PetaLinux* имеет более полный список поддерживаемых устройств, чем *OC Xillinux*, что облегчает разработчикам поиск совместимого оборудования для своих систем.

Инструменты разработки: *OC PetaLinux* поставляется с предустановленными отладчиками, включая *GNU Debugger*, *IDE Eclipse*, что упрощает отладку кода и отслеживание ошибок. В *OC Xillinux* отсутствуют многие из этих инструментов.

Производительность встроенных систем: *OC Xillinux* оптимизирован для высокопроизводительных систем, так как *OC Xillinux* использует специализированные системные утилиты, которые требуют меньше системных ресурсов, чем в системе *OC PetaLinux*.

Поддержка сообщества: *OC PetaLinux* и *OC Xillinux* имеют активную поддержку производителей и разработчиков.

Так как на базе *OC Linux Embedded* будет реализован *LAMP* сервер, выбран *OC PetaLinux*, в котором данная возможность предусмотрена по умолчанию.

1.1.3.3 Выбор языка программирования для реализации приложения

Язык *Python* используется для программирования встроенных систем *Linux*. *Python* является высокоуровневым языком, который позволяет быстро и просто разрабатывать код. Основные преимущества использования *Python* для разработки встроенных систем *Linux*:

- предлагает простой и понятный синтаксис, что делает его простым в использовании и снижает порог вхождения в *Embedded* разработку;

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

- имеет обширную библиотеку, в которой есть много полезных инструментов для работы с различными аппаратными средствами: *GPIO, Ethernet, SPI, I2C* и др.;
- является кроссплатформенным языком программирования, что означает, что код, написанный на *Python*, можно легко перенести на любую другую платформу;
- обладает хорошо структурированной системой для обработки ошибок, что упрощает написание кода на *Python* и повышает его надежность;
- интегрируется с другими языками программирования, что делает его гибким в использовании в различных встроенных системах *Linux*.

Язык *Python* поддерживает различные фреймворки и инструменты для создания *Python*-приложений: *Flask, Django, PyInstaller*.

1.1.3.5 Выбор фреймворка для реализации приложения

Flask - фреймворк, который позволяет быстро создавать веб-приложения. *Flask* предлагает множество возможностей для создания веб-сайтов, включая шаблонизацию, валидацию данных, управление пользователями и многое другое. *Flask* ориентирован на использование во встроенных системах.

Django - полноценный фреймворк для веб-программирования на *Python*. *Django* предлагает веб-интерфейс, который обеспечивает множество возможностей, высокую масштабируемость и большое количество модулей. *Django* имеет активное сообщество, которое обеспечивает профессиональную техническую поддержку.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

PyInstaller - инструмент для создания автономных приложений, которые могут быть запущены на любой платформе без установки дополнительных компонентов. *PyInstaller* позволяет упаковать *Python*-код, библиотеки и зависимости в один исполняемый файл, что упрощает развертывание приложения и избавляет от необходимости устанавливать *Python* на конечную машину.

Выбор сделан в пользу фреймворка *Flask*, так как он ориентирован на использование в *OC Linux Embedded*.

1.1.4 Выводы

В первом разделе показана актуальность выбранной темы, выбраны инструменты реализации, описан функционал реализуемого проекта.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

1.2 Реализация программно-аппаратного комплекса для обработки и хранения видео контента

1.2.1 Реализация аппаратной части проекта

Ядром аппаратной части проекта является видеорегистратор *IPTRONIC NVR1680i* и размещенный на ней *SSD* накопитель на 256 Гбайт для хранения видео информации и расширение *Ethernet FMC* на 4 интерфейса Ethernet (рисунок 2.1).

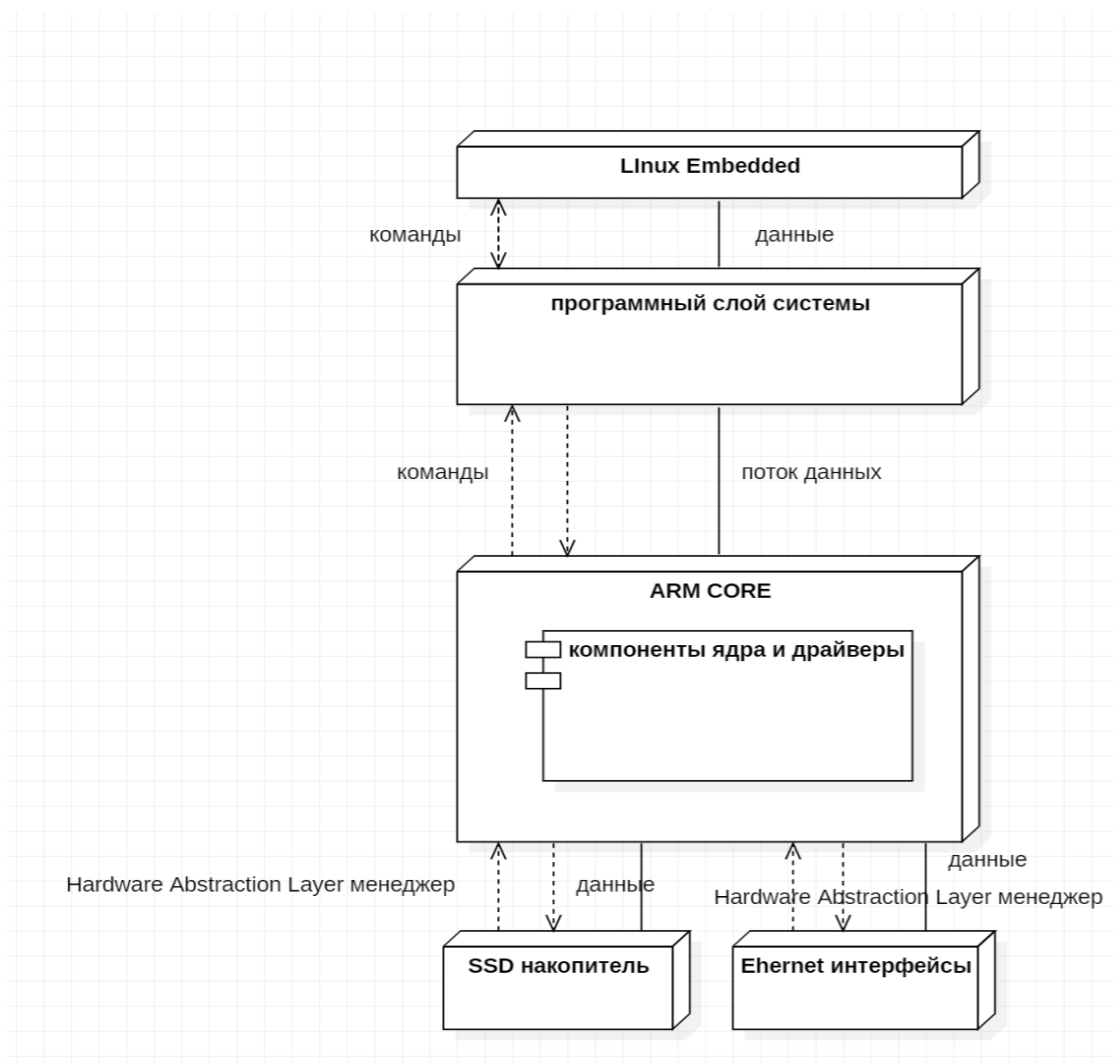


Рисунок 2.1 – Аппаратная часть комплекса с установленным *SSD* накопителем (диаграмма развертывания)

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Для работы данной схемы требуется:

- инсталляция *ОС PetaLinux*;
- установка связи между аппаратных слоев и слоев операционной системы;
- программное монтирование *SSD* накопителя в *ОС PetaLinux*, для доступа к файлам *SSD* диска со стороны web интерфейса;
- настройка конфигурационных файлов *ОС PetaLinux* для корректной работы в соответствии с техническим заданием.

1.2.1.1 Инсталляция *ОС Linux* и периферийного оборудования.

Для инсталляции *Linux Embedded* на аппаратную платформу требуется создать проект *PetaLinux* с помощью команды:

petalinux-create --type project --template prj --name petalinux_prj.

Затем требуется перейти в каталог *petalinux_prj* в командном терминале. Все последующие команды будут выполняться в рамках каталога *PetaLinux*:

- произведем импорт настроек оборудования, в проект *PetaLinux* с помощью команды: *petalinux-config --get-hw-description ../aximm_pcie/aximm_pcie.sdk/* ;
- после чего автоматически откроется диалоговое окно конфигурации системы *Linux*, в котором требуется нажать кнопку «Сохранить» (рисунок 2.8).

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

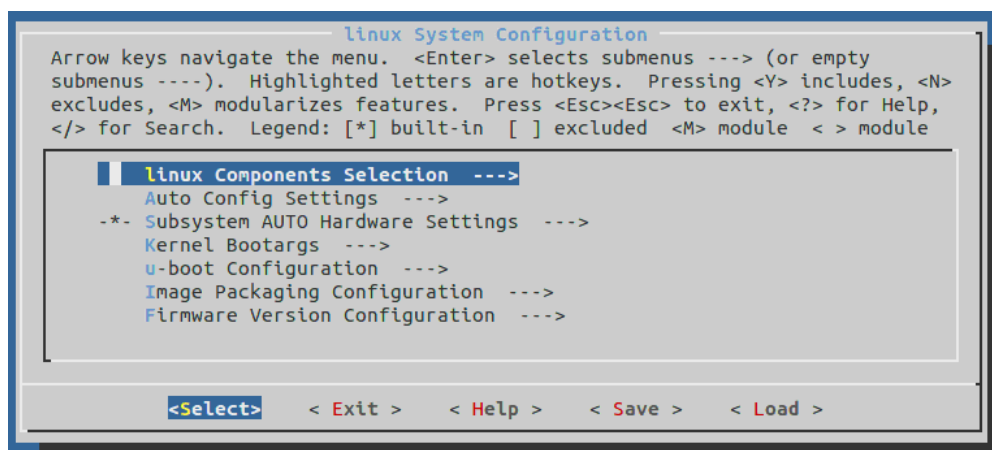


Рисунок 2.8 – Выбор компонентов инсталляции *PetaLinux*.

Настройку ядра *Linux*, осуществляемая с помощью команды *petalinux-config -c kernel*(рисунок 2.9)

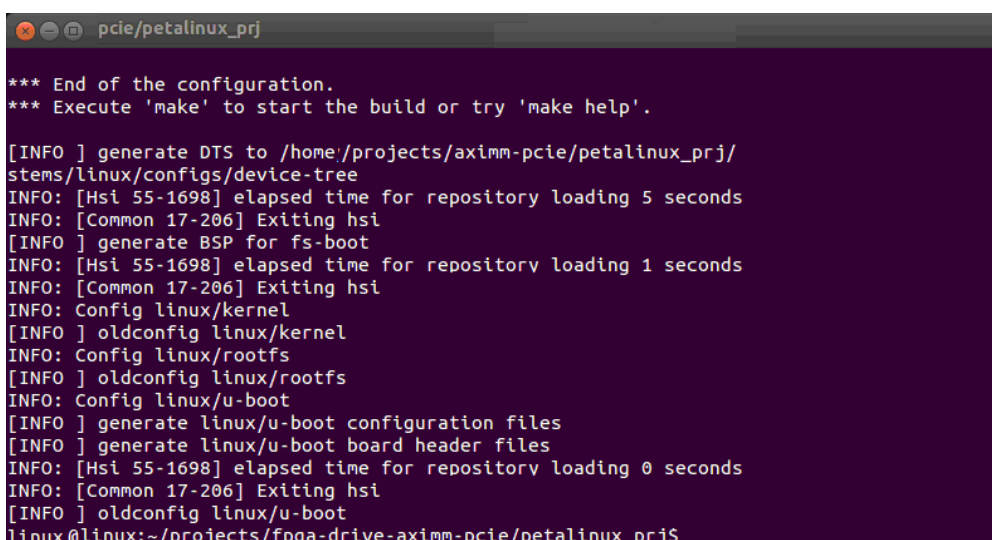


Рисунок 2.9 – Выбор компонентов инсталляции *PetaLinux*.

При выборе команды откроется диалоговое окно, в нем требуется выбрать типы периферийных интерфейсов, которые будут доступны в ОС *Linux Embedded* по умолчанию (рисунок 2.10).

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

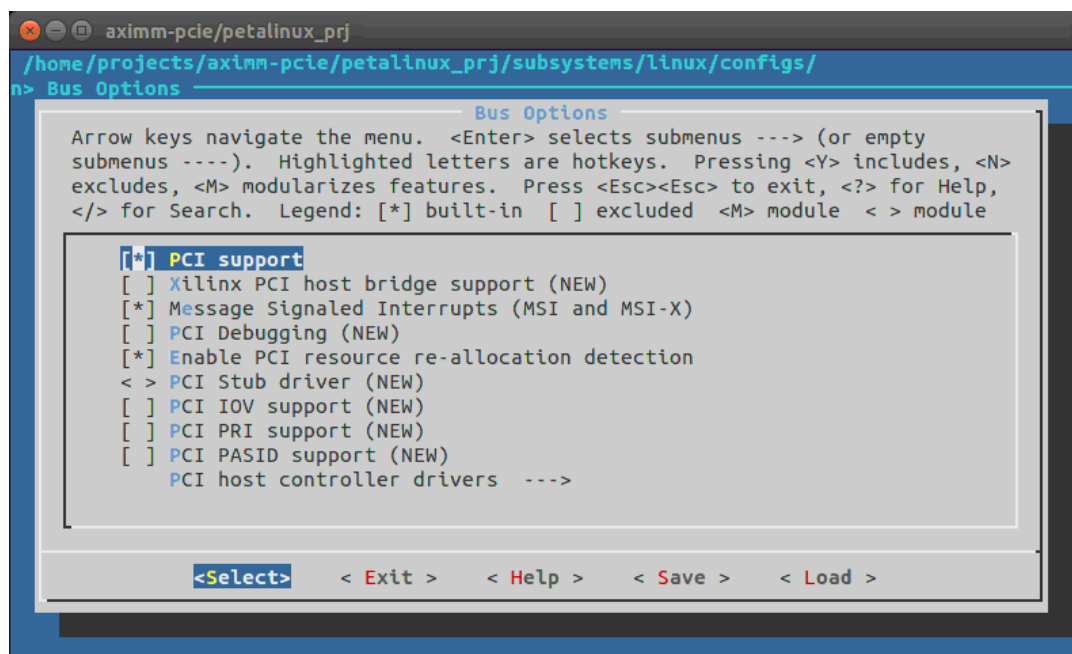


Рисунок 2.10 – Выбор типов периферийных интерфейсов *PetaLinux*.

В меню конфигурации ядра, необходимо включить поддержку SSD накопителя и платы расширения *Ethernet* путем включения поддержки соответствующих драйверов.(рисунок 2.11)

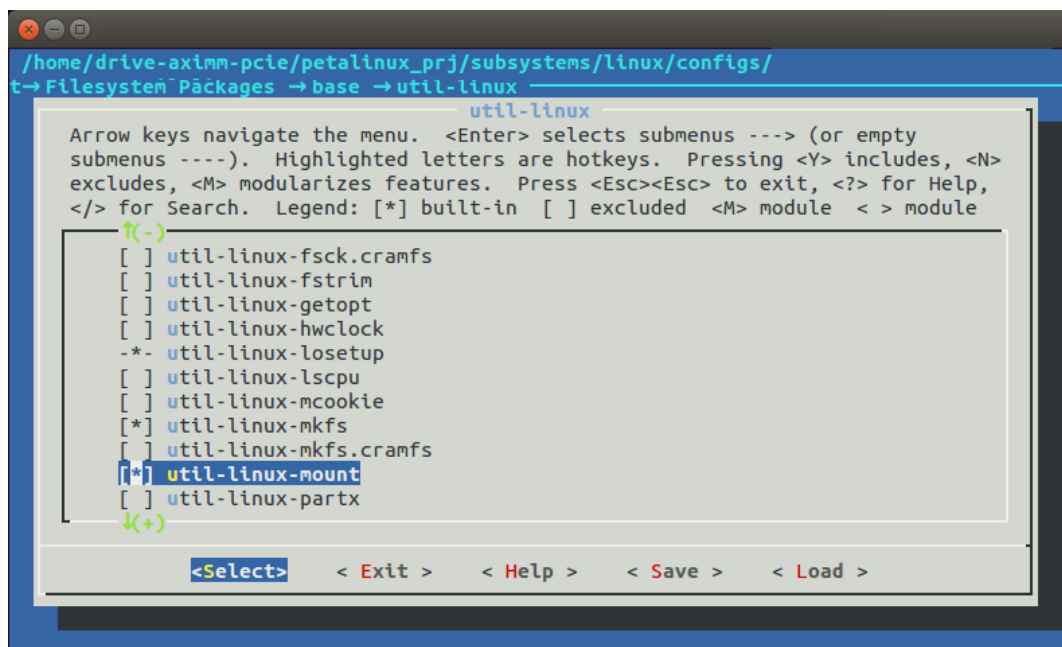


Рисунок 2.11 – Точка включения файловой системы *PetaLinux*

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

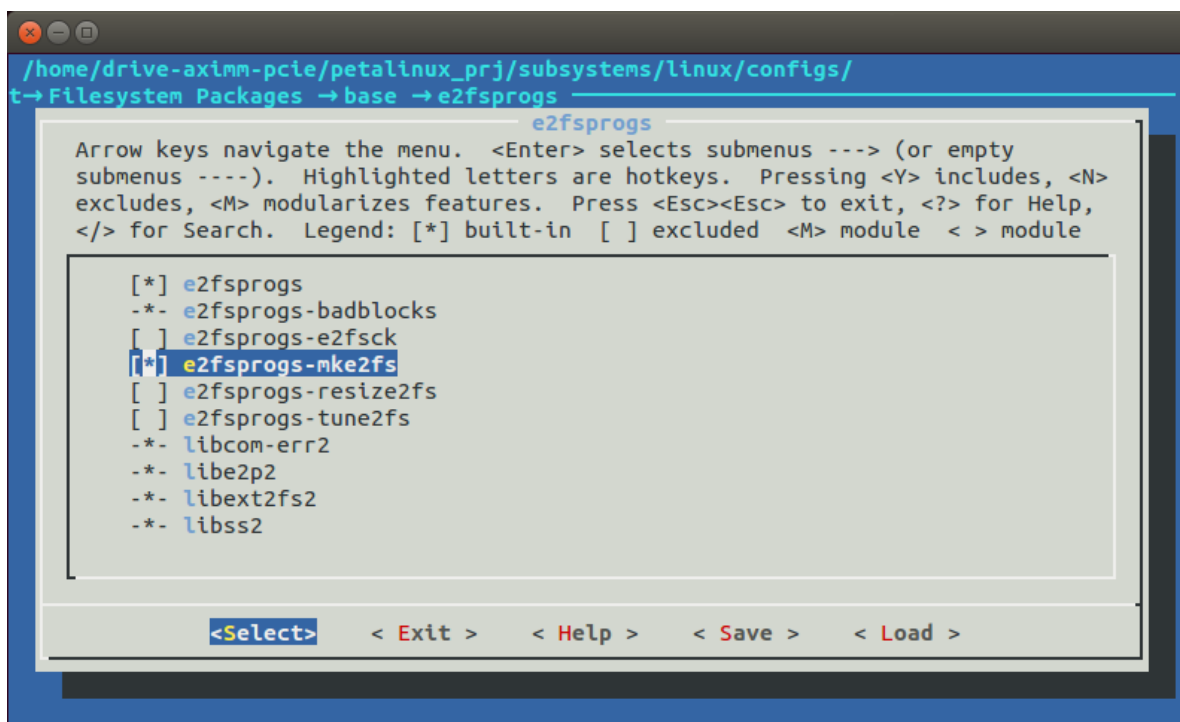


Рисунок 2.12 – Выбор типа файловой системы *PetaLinux*

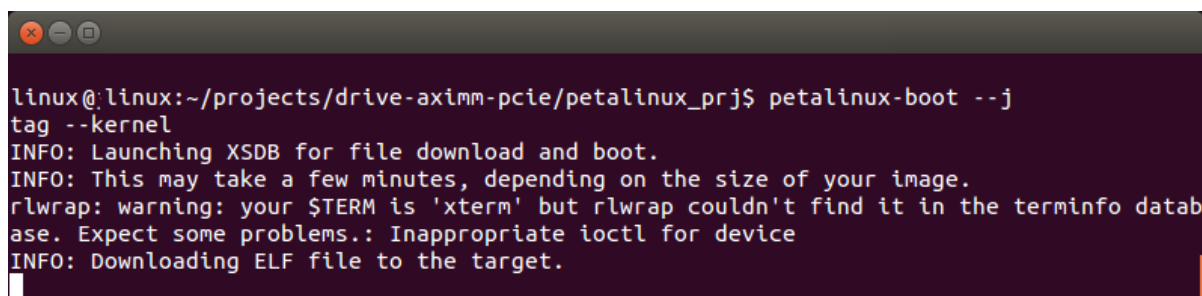
В результате выбора последовательности команд приведенных на рисунках 2.10 – 2.12 будет сформирована конфигурация ОС *Linux* (*PetaLinux*). Для формирования файла дистрибутива и создания физической сборки ОС *PetaLinux* требуется выполнить команду в командной строке: *petalinux-build*.

После окончания сборки дистрибутива операционной системы *PetaLinux*, его требуется развернуть на базе платформы видеорегистратора (на SSD накопителе с функцией автозагрузки ОС при подаче питания). Для этого подключив компьютер через интерфейс *USB* к платформе *IPTRONIC NVR1680i* требуется выполнить следующие команды:

```
petalinux-package --boot --fsbl
./images/linux/zynq_fsbl.elf/impl_1/am_ethernet_prj.bit --uboot -force
petalinux-package --prebuilt //impl_1/ am_ethernet_prj.bit
petalinux-boot --usb --prebuilt 3 -- /impl_1/ am_ethernet_prj.bit
```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Как только *ELF* файл содержащие ядро и периферийные настройки *ОС PetaLinux* будут загружены в ПЗУ платформы видеорегистратора, командная строка компьютера напечатает в командной строке сообщение об успешной загрузке ядра (рисунок 2.13).



```
linux@linux:~/projects/drive-aximm-pcie/petalinux_prj$ petalinux-boot --jtag --kernel
INFO: Launching XSDb for file download and boot.
INFO: This may take a few minutes, depending on the size of your image.
rlwrap: warning: your $TERM is 'xterm' but rlrwrap couldn't find it in the terminfo database. Expect some problems.: Inappropriate ioctl for device
INFO: Downloading ELF file to the target.
```

Рисунок 2.13 – Загрузка ядра в ПЗУ платформы видеорегистратора

После загрузки *ОС Linux Embedded* на плату в командной строке можно проверить корректность работы требуемого оборудования, подключением к платформе *IPTRONIC NVR1680i* в командной строке (рисунок 2.14 - 2.15).

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

pci 0000:01:00.0: BAR 2: no space for [io size 0x0100]
pci 0000:01:00.0: BAR 2: failed to assign [io size 0x0100]
pci 0000:00:00.0: PCI bridge to [bus 01]
pci 0000:00:00.0: bridge window [mem 0x60000000-0x600fffff]
Serial: 8250/16550 driver, 4 ports, IRQ sharing disabled
console [ttyS0] disabled
44a10000.serial: ttyS0 at MMIO 0x44a11000 (irq = 6, base_baud = 7812500) is a 16
550A
console [ttyS0] enabled
brd: module loaded
pci 0000:00:00.0: enabling device (0000 -> 0002)
nvme 0000:01:00.0: enabling device (0000 -> 0002)
xilinx_spi 44a20000.axi_quad_spi: at 0x44a20000 mapped to 0xf40e0000, irq=3
xilinx_emaclite 40e00000.ethernet: Device Tree Probing
xilinx_emaclite 40e00000.ethernet: Failed to register mdio bus.
xilinx_emaclite 40e00000.ethernet: error registering MDIO bus
xilinx_emaclite 40e00000.ethernet: MAC address is now 00:0a:35:00:22:01
xilinx_emaclite 40e00000.ethernet: Xilinx Emaclite at 0x40e00000 mapped to 0xf41
0000, irq=2
i2c /dev entries driver
TCP: cubic registered
NET: Registered protocol family 17
xilinx-pcie 50000000.axi-pcie: Trying to free unused MSI#128
nvme0n1: p1
Key type encrypted registered
Freeing unused kernel memory: 5792K (c04de000 - c0a86000)
INIT: version 2.88 booting
Creating /dev/flash/* device nodes
random: dd urandom read with 45 bits of entropy available
Starting internet superserver: inetd.
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
Removing any system startup links for run-postinsts ...
/etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpd (v1.23.1) started
Sending discover...
Sending discover...
Sending discover...
No lease, forking to background
done.

Built with PetaLinux (Yocto 1.8) petalinux_prj /dev/ttyS0
login:

```

Рисунок 2.14 – Список периферийного оборудования аппаратной платформы

```

root@linux_prj:~#
NAME      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
nvme0n1 259:0    0 238.5G  0 disk
root@linux_prj:

```

Рисунок 2.15 – Проверка доступности SSD накопителя аппаратной платформы

В результате выше проделанной работы аппаратная часть проекта подготовлена для инсталляции *web* сервера *LAMP* и размещения разработанного приложения для обработки и хранения видео контента.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

1.2.2 Реализация программного приложения.

Реализуемое программное приложение реализовано на базе клиент – серверной архитектуры (рисунок 2.16).

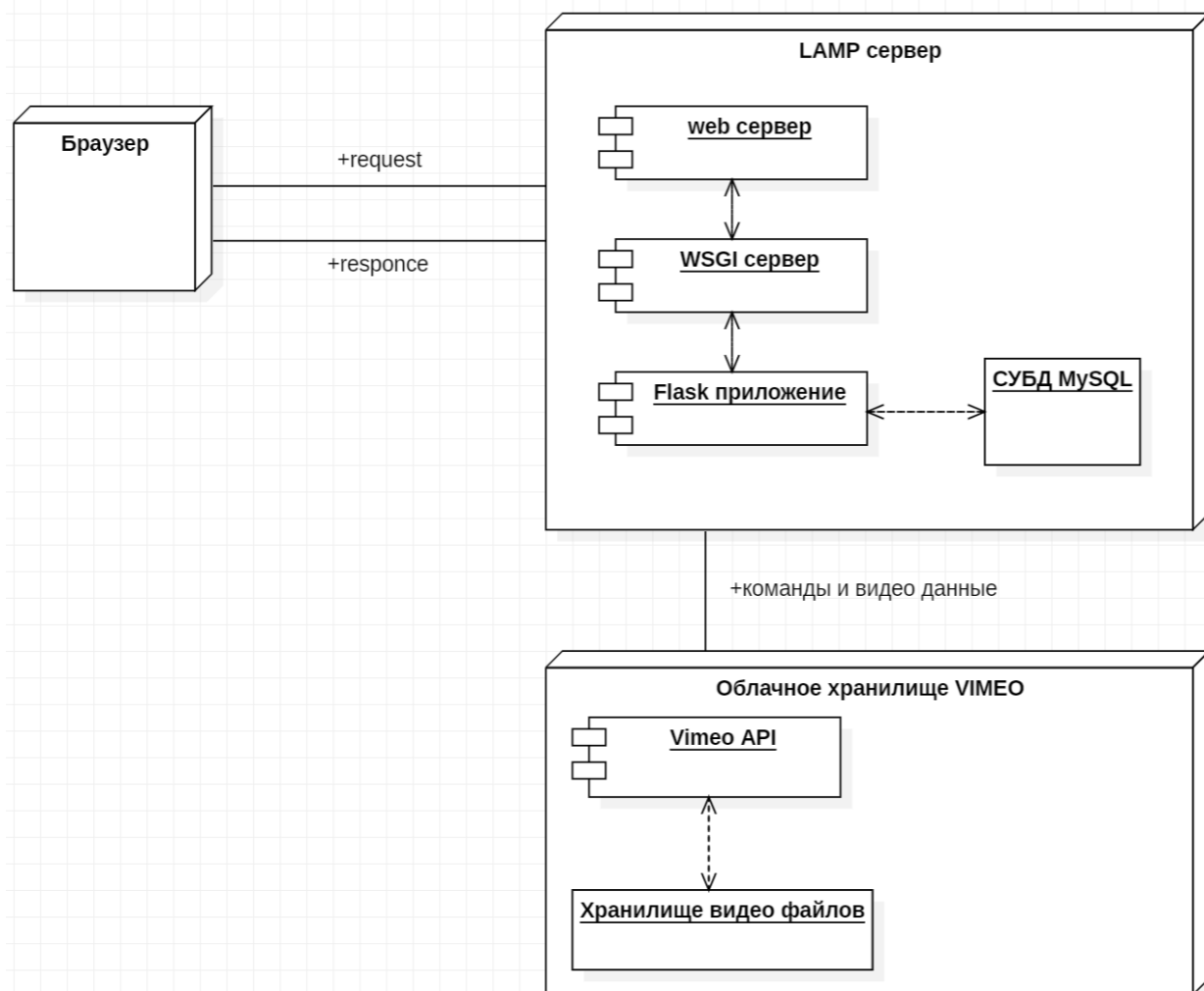


Рисунок 2.16 – Программная часть системы

«Тонкий» клиент (браузер) по определенным IP адресам может сделать запрос *LAMP* серверу.

Для формирования функциональных свойств реализованного приложения потребовалось разработать: базу данных приложения, серверную логику приложения и интерфейс приложения.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

1.2.2.1 Разработка базы данных

Корректно спроектированная база данных предоставляет доступ к актуальной информации системы. Корректный дизайн необходим для достижения в работе с базой данных. В результате проектирования СУБД, система получает базу данных, которая соответствует потребностям пользователей, разработчиков и может быть изменена без изменений в слоях интерфейса или логики приложения.

В процессе проектирования БД существуют следующие принципы:

- дублирующаяся информация не допустима (избыточные данные);
- требуется полнота информации.

Исходя из выше представленных принципов, дизайн базы данных должен:

- разделять информацию на тематические таблицы, чтобы уменьшить избыточность данных.
- обеспечить доступ к информации, необходимой для объединения информации в таблицах по мере необходимости;
- поддерживать целостность информации в БД;
- соответствовать потребностям пользователей в обработке данных и визуализации отчетности.

Опишем реализованную БД в виде логической и физической моделей, в результате чего разработчик будет иметь полную информацию о сущностях, связях и типах хранимой в БД информации.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Логическая модель БД

Таблица 2.1 – Атрибуты сущности «Устройства»

Наименование атрибута	Тип	Обязательный
Идентификатор страны	Целый	Да
Наименование устройства	Строка	Да
Уникальный идентификатор устройства в сети	Целый	Да
Уникальный идентификатор, сетевого интерфейса	Целый	Да
Тип устройства	Строка	Нет

Таблица 2.2 – Атрибуты сущности «Пользователи»

Наименование атрибута	Тип	Обязательный
Идентификатор пользователя	Целый	Да
Имя пользователя	Строка	Да
Отчество пользователя	Строка	Нет
Фамилия пользователя	Строка	Да
Адрес электронной почты	Строка	Да

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Наименование атрибута	Тип	Обязательный
Пароль	Целое (хеширование)	Да
Описание пользователя	Строка	Нет
Количество каналов	Целый	Да

Таблица 2.3 – Атрибуты сущности «Видео»

Наименование атрибута	Тип	Обязательный
Идентификатор видео	Целый	Да
Ключ безопасного удалённого доступа к информационным ресурсам	Строка	Да
Признак наличия видео	Целый	Нет
Заголовок	Строка	Нет
Описание	Строка	Нет
Время создания видео	Временная метка	Да
Время изменения видео	Временная метка	Да
Ссылка на видео (URL)	Строка	Да

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 2.4 – Атрибуты сущности «Видео каналы»

Наименование атрибута	Тип	Обязательный
Идентификатор канала	Целый	Да
Наименование канала	Строка	Да
Счетчик видео файлов	Целый	Да
Время создания видео	Временная метка	Да
Время изменения видео	Временная метка	Да

Таблица 2.5 – Атрибуты сущности «Комментарии»

Наименование атрибута	Тип	Обязательный
Идентификатор комментария	Целый	Да
Текст комментария	Строка	Да
Время создания видео	Временная метка	Да
Время изменения видео	Временная метка	Да

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

В результате проектирования логической модели БД, сформированы сущности, которые позволят спроектировать и реализовать физическую модель БД (рисунок).

Физическая модель БД

Таблица 2.6 – Поля и описание таблицы «*Devices*»

Имя поля	Тип	Примечание
<i>id</i>	<i>int</i>	Первичный ключ, обязательный
<i>deviceName</i>	<i>varchar(255)</i>	Обязательный
<i>ipAdress</i>	<i>integer</i>	Обязательный
<i>MAC</i>	<i>integer</i>	Обязательный
<i>type</i>	<i>varchar(255)</i>	Необязательный
<i>userid</i>	<i>integer</i>	Внешний ключ
<i>videoid</i>	<i>integer</i>	Внешний ключ

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 2.7 – Поля и описание таблицы «*Users*»

Имя поля	Тип	Примечание
<i>id</i>	<i>int</i>	Первичный ключ, обязательный
<i>firstName</i>	<i>varchar(255)</i>	Обязательный
<i>secondName</i>	<i>varchar(255)</i>	Необязательный
<i>lastName</i>	<i>varchar(255)</i>	Обязательный
<i>e-mail</i>	<i>varchar(255)</i>	Обязательный
<i>userid</i>	<i>integer</i>	Внешний ключ
<i>password</i>	<i>integer</i>	Обязательный (хеширование)
<i>profile</i>	<i>text</i>	Необязательный
<i>channel</i>	<i>integer</i>	Обязательный

Таблица 2.8 – Поля и описание таблицы «*Videos*»

Имя поля	Тип	Примечание
<i>id</i>	<i>int</i>	Первичный ключ, обязательный
<i>streaming_url</i>	<i>text</i>	Обязательный
<i>key</i>	<i>varchar(255)</i>	Обязательный
<i>processed</i>	<i>boolean</i>	Обязательный

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Имя поля	Тип	Примечание
<i>title</i>	<i>varchar(255)</i>	Необязательный
<i>description</i>	<i>text</i>	Необязательный
<i>createdAt</i>	<i>timestamp</i>	Обязательный
<i>updatedAt</i>	<i>timestamp</i>	Обязательный

Таблица 2.9 – Поля и описание таблицы «*Channels*»

Имя поля	Тип	Примечание
<i>id</i>	<i>integer</i>	Первичный ключ, обязательный
<i>user_id</i>	<i>integer</i>	Внешний ключ
<i>name</i>	<i>varchar(255)</i>	Обязательный
<i>video_count</i>	<i>integer</i>	Обязательный
<i>createdAt</i>	<i>timestamp</i>	Обязательный
<i>updatedAt</i>	<i>timestamp</i>	Обязательный

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 2.10 – Поля и описание таблицы «*Comments*»

Имя поля	Тип	Примечание
<i>id</i>	<i>integer</i>	Первичный ключ, обязательный
<i>user_id</i>	<i>integer</i>	Внешний ключ
<i>comment</i>	<i>text</i>	Обязательный
<i>createdAt</i>	<i>timestamp</i>	Обязательный
<i>updatedAt</i>	<i>timestamp</i>	Обязательный

Реализованные таблицы полностью покрывают потребности приложения в данных и способны выполнять базовые функции СУБД *MySQL*: создание, чтение, модификация, удаление.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

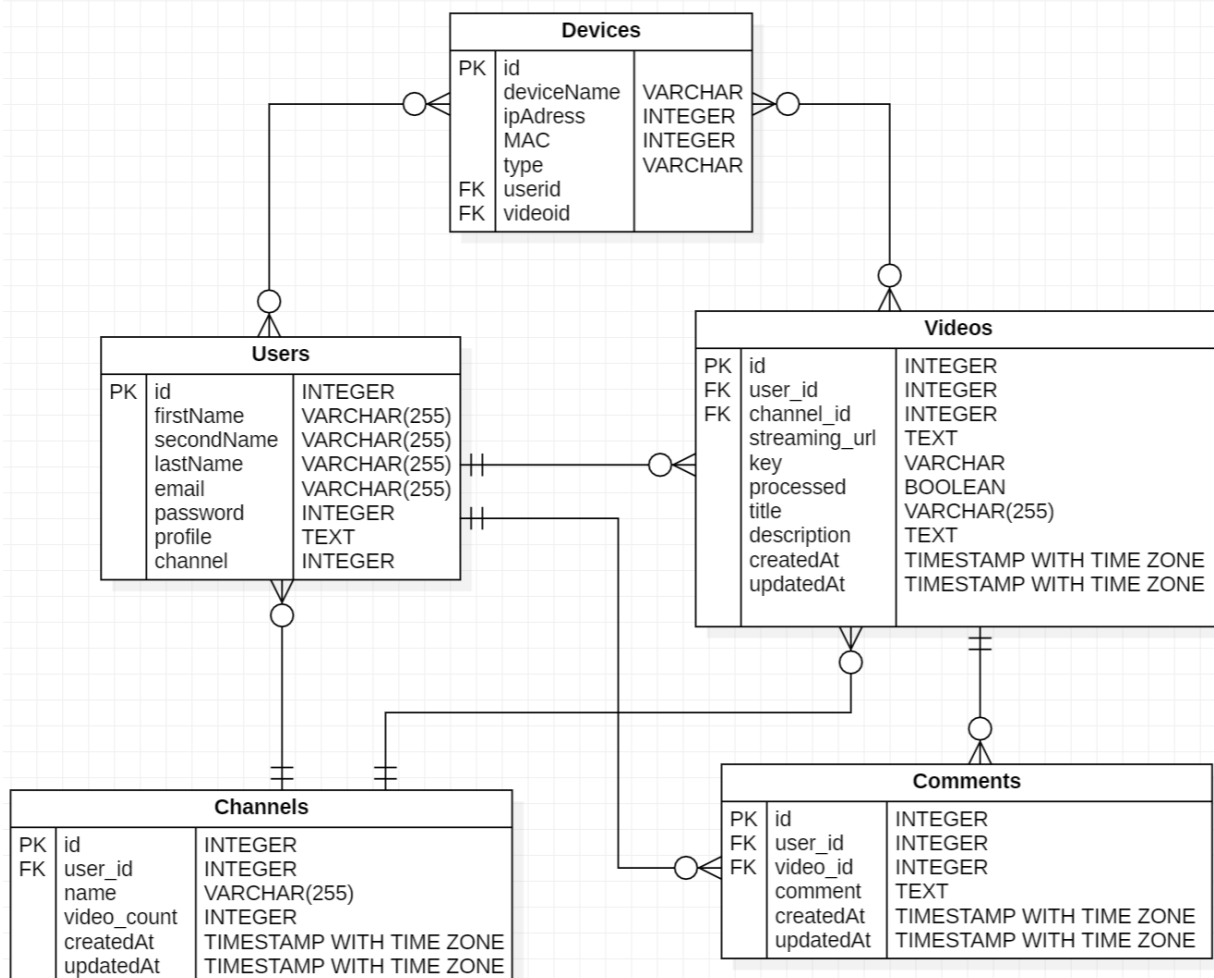


Рисунок 2.16 – Физическая модель БД

1.2.2.2 Разработка серверной логики приложения

В использовании *python* в качестве инструмента обработки данных и команд на стороне сервера есть определенная специфика, которая будет отражена в процессе описания реализации серверной части приложения программно-аппаратного комплекса для обработки и хранения видео контента.

Из браузера в сторону *web* сервера *Apache* отправляется запрос на запрос определенного *URL*. Веб-сервер, полученный запрос, направляет для обработки на сервер *WSGI*.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Web Server Gateway Interface, представляет собой интерфейс между *web* сервером и *web* приложением на основе *Python*. Это необходимо, поскольку веб-сервер не может напрямую взаимодействовать с приложением *Python*. Сервер *WSGI* сообщает приложению *Flask* запрос для обработки. Приложение *Flask* обрабатывает и генерирует ответ, который отправляется обратно на сервер *WSGI*, затем на веб-сервер и, в конечном счете, обратно в веб-браузер. Эти шаги описывают цикл «запрос-ответ», который является ключевой функцией того, как запрос обрабатывается через *web* сервер, сервер приложений *WSGI* и веб-приложение.

В соответствии с техническим заданием реализованы функции, которые предоставляют следующие возможности пользователю (рисунок 2.17) :

-*loginAction(login, password)* – вход в систему авторизованного пользователя;

-*viewVideo(listItem)* – отображение содержимого видео файла;

-*viewProcessVideo(url)* – просмотр видео контента в браузере;

-*deleteVideo(listItem)* – удаление видео из списка видео контента;

-*viewDevice(id)* – просмотр информации об устройстве;

-*getDeleteDevice(id)* – удаление устройства.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

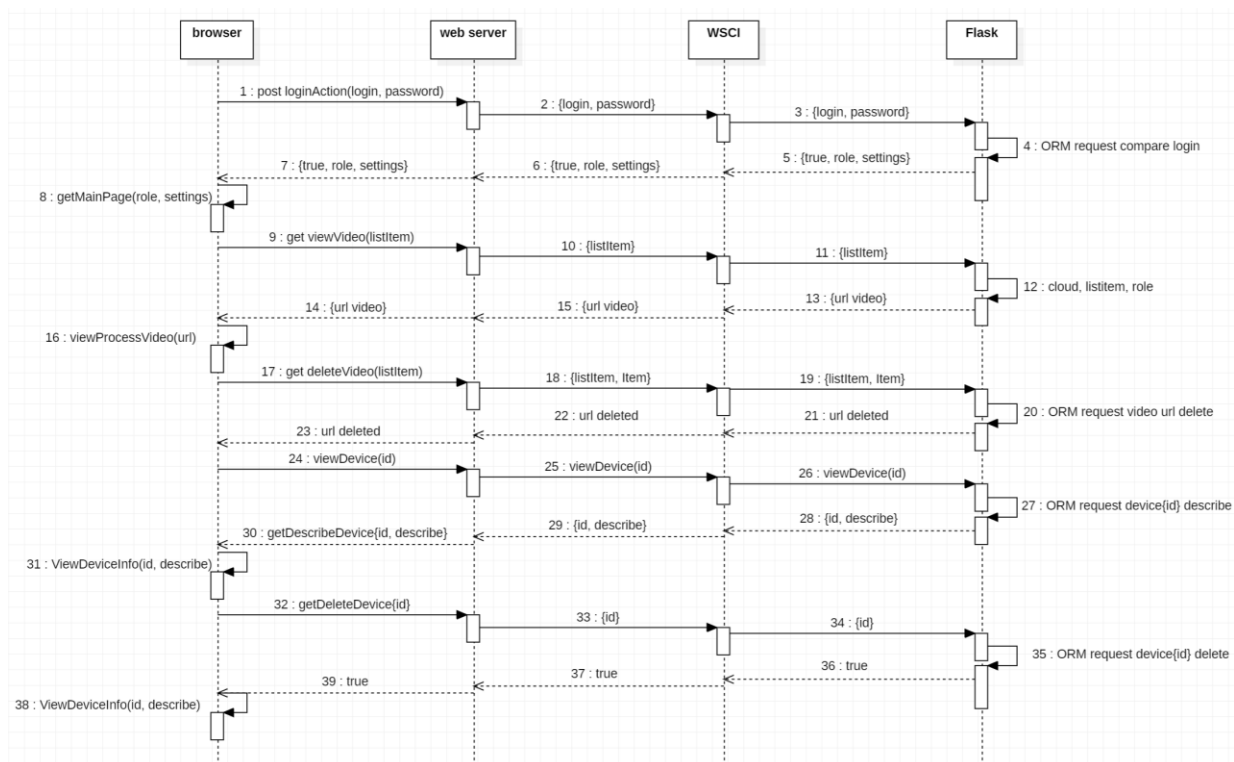


Рисунок 2.17 – Запросы к серверу и их исполнение (пользователь)

В соответствии с техническим заданием реализованы функции, которые предоставляют следующие возможности администратору системы (рисунок 2.18):

-loginAction(login, password) - вход в систему авторизованного пользователя

-addDevice(id, describe) – добавить новое устройство,

-addUser(id, describe) – добавить нового пользователя,

-addCloud(id, name, url) – добавиь новое облачное хранилище,

-updateUserInfo(id, describe) – обновить информацию о пользователе,

-deleteUser(id) – удалить пользователя.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

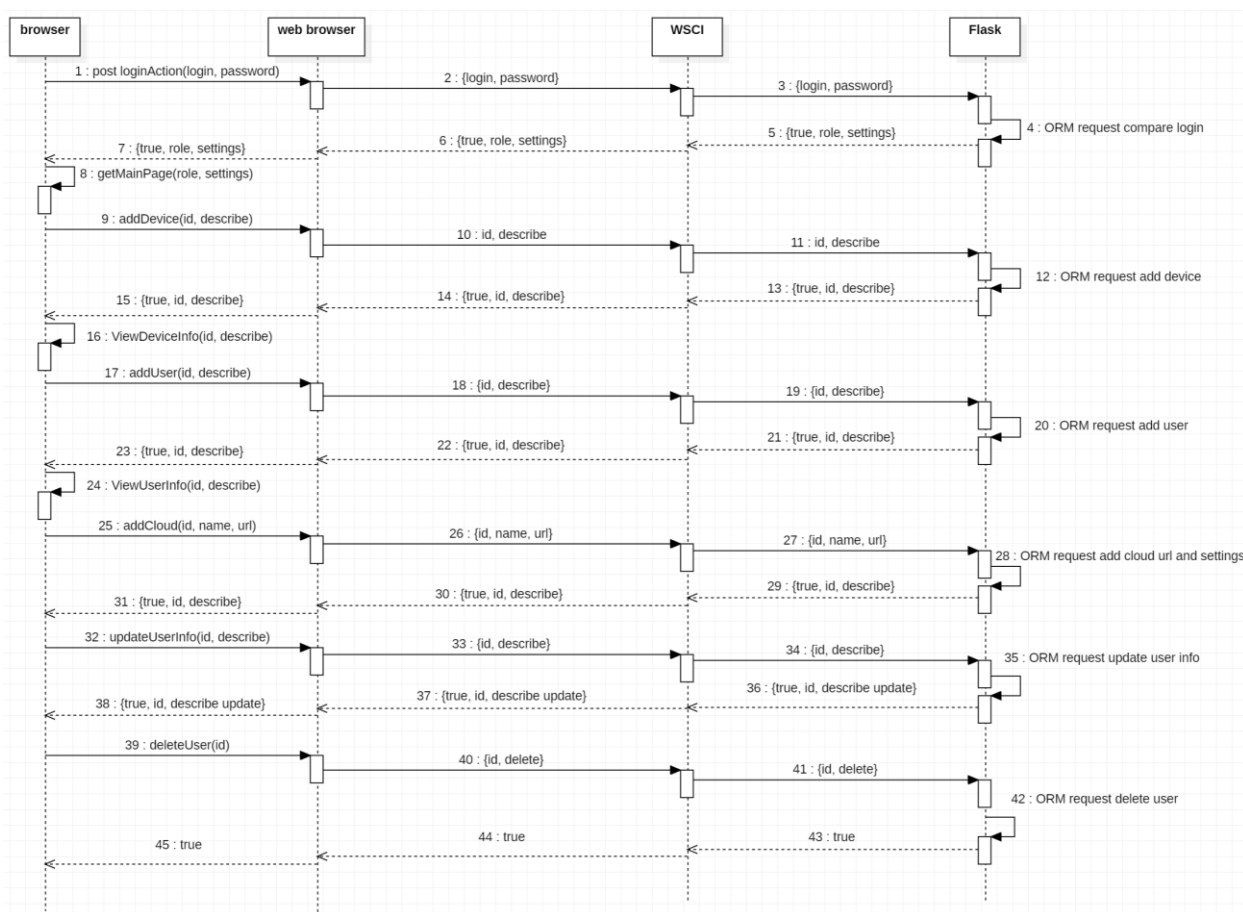


Рисунок 2.18 – Запросы к серверу и их исполнение (администратор)

Отдельно опишем процесс работы с облачным сервисом *Vimeo*. Для работы с данным сервисом через *API* требуется подключить функционал сервиса с помощью команды *import vimeo* требуется его установить в проект. Для этого импортируется пакет *PyVimeo* в *PyPI*. командой: *pip install PyVimeo*.

После чего необходимо добавить в код приложения функционал по работе с облачным сервисом:

Добавить видео на уже существующий канал:

```
video_url = v.upload('device3_25052023.mp4')
```

Добавить метаданные в видео перед его загрузкой в облачный сервис:

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```
video_uri = v.upload(' device3_25052023.mp4', data={'video25052023': 'Офис.
Camera 3', 'AMaslov. test.'})
```

Таким образом используя встроенные в облачный сервис *API* и стандартные инструменты фрейворка *Flask* реализована серверная часть приложения.

1.2.2.3 Разработка интерфейсов клиентской части приложения

Создание клиентской части приложения представляет собой процесс проектирования и реализацию пользовательского интерфейса, в том числе и создание структуры приложения.

Этапами реализации клиентской части являются: определение стиля приложения, определение цветовой палитры интерфейса, применение элементов управления и подсказок.

На основании структуры сайта (рисунки 2.19 и 2.20) ниже представлены страницы реализованного интерфейса программно-аппаратный комплекс для обработки и хранения видео контента.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

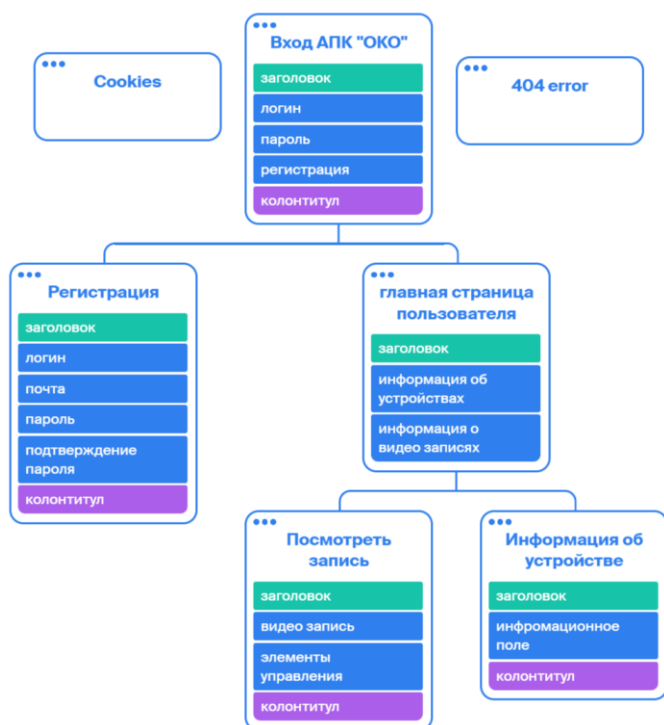


Рисунок 2.19 – Структура интерфейса (пользователь)

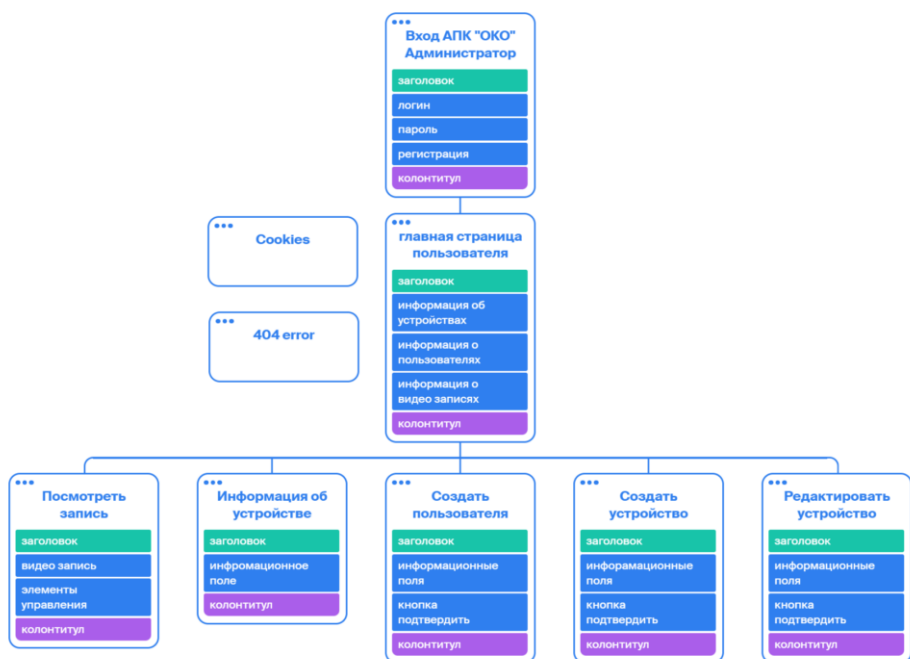


Рисунок 2.20 – Структура интерфейса (администратор)

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Элементами страницы регистрации являются следующие блоки (рисунок 2.21.):

- форма регистрации – тип *Form*;
- поле ввода логина - тип *Input type*;
- поле ввода e-mail - тип *Input type*;
- поле ввода пароля - тип *Input type*;
- поле ввода подтверждение пароля - тип *Input type*;
- поле ввода подтверждение пароля - тип *Input type*;
- кнопка регистрации - тип *Button*;
- ссылка на страницу авторизации - тип *Href*.

Регистрация АПК "ОКО"

Логин

Почта

Пароль

Подтверждение пароля

[Авторизация в системе](#)

Рисунок 2.21 – Страница регистрации пользователей

Элементами страницы авторизации пользователей являются следующие блоки (рисунок 2.22):

- форма авторизации – тип *Form*;
- поле ввода логина - тип *Input type*;
- поле ввода пароля - тип *Input type*;
- кнопка входа в систему - тип *Button*;

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

- кнопка входа в систему тип *Button*;
- ссылка на страницу регистрации - тип *Href*.

Рисунок 2.22 – Страница авторизации пользователей

Элементами главной страницы пользователя являются следующие блоки (рисунок 2.23.):

- форма страницы – тип *Form*;
- информационное поле - тип *Label* и *var*;
- таблица с информацией об устройствах - тип *table*;
- таблица с информацией о видео записях - тип *table*;
- кнопки управления устройствами - тип *Button*;
- кнопки управления видео записями - тип *Button*.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Главная страница пользователя user2

Выйти

Количество подключенных устройств: 2
Количество видео файлов: 232

Информация об устройствах

№	Подписанное устройство	Действие	Количество записей
1	HOST_Stream1	<div>Удалить</div> <div>Запись</div>	100
2	HOST_Stream3	<div>Удалить</div> <div>Запись</div>	123

Информация о записях

№	Запись	Устройство	Действие
1	video_room10.06.23	HOST_Stream1	<div>Удалить</div> <div>Просмотр</div>
2	video_room21.05.23	HOST_Stream1	<div>Удалить</div> <div>Просмотр</div>

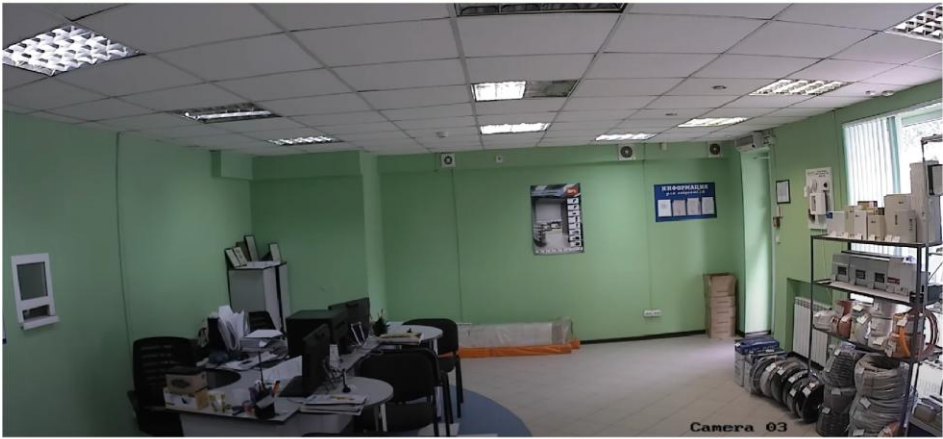
Рисунок 2.23 – Главная страница авторизованного пользователя

Элементами страницы просмотра видео файла являются следующие блоки (рисунок 2.24):

- форма страницы – тип *Form*;
- таблица с информацией о видео записях - тип *table*;
- кнопки управления просмотром - тип *Button*.

Просмотр записи video_room29.05.23

Назад



Старт

Стоп

Удалить

Рисунок 2.24 – Страница просмотра видео файла

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Элементами главной страницы администратора являются следующие блоки (рисунок 2.25):

- форма страницы – тип *Form*;
- информационное поле - тип *Label* и *var*;
- таблица с информацией об устройствах - тип *table*;
- таблица с информацией о пользователях - тип *table*;
- таблица с информацией о видео записях - тип *table*;
- кнопки управления устройствами - тип *Button*;
- кнопки управления пользователями - тип *Button*;
- кнопки управления видео записями - тип *Button*.

Главная страница администратора admin1

Выйти

Количество пользователей: 2
Количество подключенных устройств: 2
Количество видео файлов: 232

Информация об устройствах

№	Подписанное устройство	Действие		IP-адрес устройств
1	HOST_Stream1	Удалить	Редактировать	192.168.23.5
2	HOST_Stream3	Удалить	Редактировать	192.168.73.9

Добавить устройство

Информация о пользователях

№	Логин	Действие		Описание
1	user2	Удалить	Редактировать	Пользователь
2	admin1	Удалить	Редактировать	Администратор

Добавить пользователя

Информация о записях

№	Хранилище	Устройство	Действие	
1	WS_hosh_19	PYNQ_Stream1	Удалить	Просмотр
2	video_room29.05.23	PYNQ_Stream1	Удалить	Просмотр

Привязать хранилище

Рисунок 2.25 – Главная страница администратора

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Элементами страницы с информацией об устройстве являются следующие блоки (рисунок 2.26):

- форма страницы – тип *Form*;
- информационное поле - тип *Label*;
- поле ввода данных «имя устройства» - тип *input*;
- поле ввода данных «ip адрес» - тип *input*;
- поле ввода данных «URL хранилища видео файлов» - тип *input*;
- поле ввода данных «пользователь» - тип *input*;
- кнопка управления - тип *Button*.

Создание устройства		Назад
Информация о устройстве		
Имя устройства	<input type="text" value="HOST_Stream3"/>	
IP-адрес	<input type="text" value="192.168.56.3"/>	
Адрес хранилища	<input type="text" value="https://userid123456.vimeo.com/ru/build_hosh_25"/>	
Пользователь	<input type="text" value="user2"/>	
<input type="button" value="Применить"/>		

Рисунок 2.26 – Страница с информацией об устройстве (администратор)

Элементами страницы с информацией о пользователе являются следующие блоки (рисунок 2.27):

- форма страницы – тип *Form*;
- информационное поле - тип *Label*;
- поле ввода данных «пользователь» - тип *input*;
- поле ввода данных «роль» - тип *input*;

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

- поле ввода данных «*URL* хранилища видео файлов» - тип *input*;
- поле ввода данных «устройство» - тип *input*;
- кнопка управления - тип *Button*.

Создание пользователя		Назад
Информация о пользователе		
Имя пользователя	<input type="text" value="user3"/>	
Роль	<input type="text" value="пользователь"/>	
Адрес хранилища	<input type="text" value="https://userid123456.vimeo.com/ru/build_hosh_25"/>	
Устройство	<input type="text" value="HOST_Stream3"/>	
<input type="button" value="Применить"/>		

Рисунок 2.27 – Страница с информацией о пользователе (администратор)

1.2.3 Выводы

Во втором подразделе показана реализация аппаратной части проекта, реализация серверной и клиентской частей приложения.

Все части системы реализованы и собраны в единую систему.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

2. Технология установки и настройки сервера *LAMP* для комплекса обработки и хранения видео контента

The LAMP stack - это набор инструментов с открытым исходным кодом, используемых для разработки веб-приложений.

Чтобы *web* приложение работало, оно должно включать в себя серверную систему, *web* сервер, базу данных и язык программирования. Каждый уровень программного обеспечения необходим для создания динамичного веб-сайта, основанного на базе данных.

LAMP - это набор из четырех компонентов, составляющих полностью функциональную среду *web* разработки. Аббревиатура *LAMP* содержит инициалы названий компонентов: *OS Linux Embedded*, *HTTP* - сервер *Apache HTTP Server*, СУБД *MySQL*, язык программирования *Python*.

Выполните следующие действия, чтобы установить каждый инструмент для реализации программно-аппаратный комплекс для обработки и хранения видео контента.

2.1 Установка сервера *Apache*

HTTP-сервер *Apache* - это *web* сервер, работающий в среде *OS Linux* в стеке *LAMP*. Сервер использует *HTTP* для обработки запросов и передачи информации через Интернет.

Перед установкой первого компонента *LAMP* убедитесь, что список пакетов в системе обновлен.

В терминале введите команду *sudo apt update*.

Чтобы установить пакет *Apache*, выполните следующую команду: *sudo apt install apache2* -у (рисунок 3.1).

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

main:~$ sudo apt install apache2 -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2
0 upgraded, 1 newly installed, 0 to remove and 11 not upgraded.
Need to get 95.6 kB of archives.
After this operation, 543 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 apache2 amd64 2.4.41-4ubuntu3.12 [95.6 kB]
Fetched 95.6 kB in 1s (119 kB/s)
Selecting previously unselected package apache2.
(Reading database ... 174557 files and directories currently installed.)
Preparing to unpack .../apache2_2.4.41-4ubuntu3.12_amd64.deb ...
Unpacking apache2 (2.4.41-4ubuntu3.12) ...
Setting up apache2 (2.4.41-4ubuntu3.12) ...
Processing triggers for systemd (245.4-4ubuntu3.18) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ufw (0.36-6ubuntu1) ...
main:~$

```

Рисунок 3.1 – Процесс установки сервера *Apache*

После установки требуется проверить правильно ли установлен сервер, проверив состояние службы *Apache* командой: *sudo service apache2 status*

Сервис показывает статус *running* в выходных данных (рисунок 3.2):

```

main:~$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enab
   Active: active (running) since 26:05:23 EDT; 4min 53s ago
     Docs: https://httpd.apache.org/docs/2.4/
    Main PID: 15228 (apache2)
      Tasks: 6 (limit: 11573)
     Memory: 9.4M
    CGroup: /system.slice/apache2.service
            └─15228 /usr/sbin/apache2 -k start
              └─15230 /usr/sbin/apache2 -k start
                └─15231 /usr/sbin/apache2 -k start
                  └─15233 /usr/sbin/apache2 -k start
                    └─15234 /usr/sbin/apache2 -k start
                      └─15235 /usr/sbin/apache2 -k start

26:05:23 main systemd[1]: Starting The Apache HTTP Server...
26:05:23 main apachectl[15227]: AH00558: apache2: Could not reliably deter
26:05:23 main systemd[1]: Started The Apache HTTP Server.
lines 1-18/18 (END)

```

Рисунок 3.2 – Состояние сервера *Apache*

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Затем требуется убедиться, что брандмауэр *UFW* содержит профили *Apache*, введя следующую команду: *sudo ufw app list* (рисунок 3.3).

```
main:~$ sudo ufw app list
Available applications:
Apache
Apache Full
Apache Secure
CUPS
```

Рисунок 3.3 – Состояние фаервола

Чтобы подтвердить, что *Apache* запущен, введите *IP*-адрес сервера в адресной строке интернет-браузера и нажмите кнопку *ENTER*.

Страница тестового запроса к серверу *Apache* должна отображаться следующим образом (рисунок 3.4).

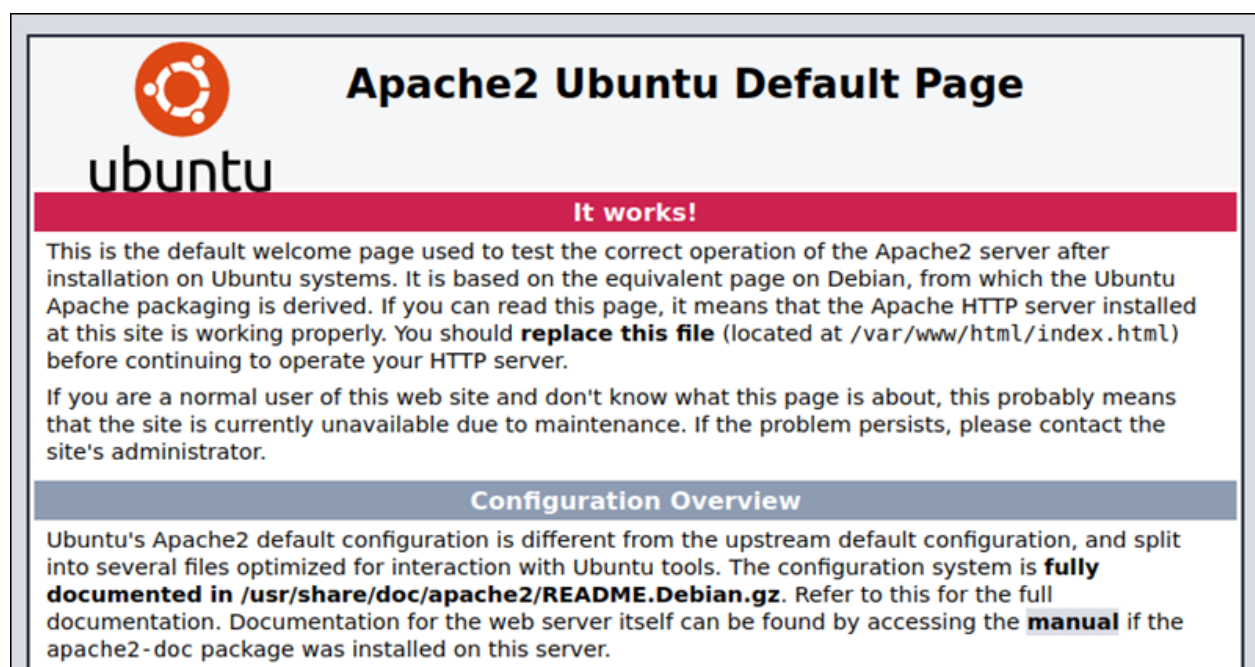


Рисунок 3.4 – Тестовая страница *web* сервера

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

2.2 Установка MySQL

СУБД *MySQL* - это реляционная система управления базами данных для создания и обслуживания динамических баз данных. СУБД совместима со всеми основными платформами операционной системы, что делает ее подходящим вариантом для разработки веб-приложений.

Установите *MySQL* следующей командой: `sudo apt install mysql-server -y` (рисунок 3.5).

```
Setting up mysql-server-8.0 (8.0.30-0ubuntu0.20.04.2) ...
update-alternatives: using /etc/mysql/mysql.cnf to provide /etc/mysql/my.cnf (my.cnf) i
n auto mode
mysqld will log errors to /var/log/mysql/error.log
mysqld is running as pid 19563
Setting up mysql-server (8.0.30-0ubuntu0.20.04.2) ...
Processing triggers for systemd (245.4-4ubuntu3.18) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
```

Рисунок 3.5 – Установка СУБД

2.3 Установка Python

Произведите обновление системы командами соответственно: `sudo apt update` и `sudo apt -y upgrade`

Как только процесс установки завершен, можно проверить версию *Python*, которая установлена в системе, набрав команду `python3 -V`.

Чтобы управлять программными пакетами для *Python*, требуется установить пакет *pip*, инструмент, который будет устанавливать программные пакеты, которые требуется использовать в проекте разработки, и управлять ими `sudo apt install -y python3-pip`.

Дополнительные пакеты *Python* можно установить, набрав: `pip3 install VivadoDevice`.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

2.4 Выводы по разделу

Во втором разделе показана технология инсталляции и настройки компонентов сервера *LAMP*.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

3. Экономическое обоснование разработки

Разработка программного обеспечения осуществлена на базе собственной площади 150 кв.м. Помещение находится в бизнес инкубаторе, содержание помещения в месяц составляет 6 000 рублей по льготному тарифу для IT предприятий.

Для разработки аппаратно-программного комплекта приобретена вычислительная техника и расходные материалы (обучающие материалы). Лицензионное программное обеспечение подобрано таким образом, что все программные инструменты являются бесплатными.

Данные по потребности в оборудовании и лицензионном программном обеспечении представлены соответственно в таблице 4.1 и 4.2.

Таблица 4.1 – Потребность в оборудовании

Оборудование	Организация	Кол-во	Ориентировочная цена за единицу, руб
Компьютер самостоятельная сборка	ООО «Таганай»	5	90000
МФУ лазерный А4	ООО «Принт»	1	14000
Клавиатура и мышь	ООО «Квинта»	5	800
Монитор LG 27 дюймов	ООО «Квинта»	5	32000

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Оборудование	Организация	Кол-во	Ориентировочная цена за единицу, руб
Аппаратная платформа видео регистратора <i>IPTRONIC NVR1680i</i>		2	12000

Таблица 4.2 – Потребность в лицензионном программном обеспечении

Лицензионное программное обеспечение	Источник	Цена, рублей	Срок действия лицензии, дней
<i>OS Linux Embedded</i>	сайт производителя	0	бессрочная
Антивирус <i>Comodo</i>	сайт производителя	0	бессрочная

Разработка программно-аппаратный комплекс для обработки и хранения видео контента осуществлялась командой разработчиков, состав которой представлен в таблице 4.3. Состав коллектива разработчиков и информация по заработной плате разработчиков программного обеспечения и их количественному составу приведены в таблице 4.3.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 4.3 – Занятость и заработная плата разработчиков

Специалист	Количество, человек	Сумма заработной платы, рублей
Руководитель проекта	1	80000
Тимлидер	1	60000
Программист	1	60000
<i>DevOps</i> инженер	1	80000
Тестировщик	1	50000

Себестоимость разработки программно-аппаратного комплекса включает следующие статьи затрат:

- затраты на приобретение оборудования;
- затраты на дополнительные материалы;
- затраты на заработную плату;
- затраты на обслуживание.

Затраты на оборудование в период разработки составят 652000 рублей на основании данных в таблице 4.1.

Затраты на программное обеспечение в период разработки составят 0 рублей и представлены в таблице 4.2.

Затраты на дополнительные материалы в период разработки составят 0 рублей в соответствии с условием что потребности в закупке нет.

Затраты на заработную плату включают затраты на основную и дополнительную заработную плату разработчиков программного обеспечения и отчисления на страховые взносы.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Затраты на основную заработную плату разработчиков определяются исходя из установленной заработной платы, количества специалистов и их занятости в процессе разработки программного продукта (продолжительность работы в днях).

Дополнительная заработная плата составляет 15% от затрат на основную заработную плату разработчиков.

С учетом выходных дней при 5-дневной рабочей неделе количество рабочих дней в месяце в среднем составляет 21 день.

Затраты на основную и дополнительную заработную плату разработчиков программного обеспечения представлены в таблице 4.4.

Таким образом, затраты на заработную плату разработчиков составят 1138500 рублей.

Отчисления на страховые взносы по нормам, установленным законодательством в ПРФ, ФОМС, ФСС РФ вычисляются исходя из данных представленных ниже.

Таблица 4.4 – Затраты на основную и дополнительную заработную плату разработчиков

Исполнитель	Количество, человек	Сумма заработной платы, рублей	Период занятости, дней	Основная заработная плата, рублей	Дополнительная заработная плата, рублей	Всего рублей
Руководитель проекта	1	80000	63	240000	36000	276000

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Исполнитель	Количество, человек	Сумма заработной платы, рублей	Период занятости, дней	Основная заработная плата, рублей	Дополнительная заработная плата, рублей	Всего рублей
Лидер команды разработки	1	60000	63	180000	27000	207000
Программист	1	60000	63	180000	27000	207000
DevOps инженер	1	80000	63	240000	36000	276000
Тестировщик	1	50000	63	150000	22500	172500
Итого	5	330000	315	990000	148500	1138500

Отчисления на страховые взносы производится согласно «Налоговому кодексу Российской Федерации (часть вторая)» от 05.08.2000 № 117-ФЗ (ред. от 26.03.2022).

Действующие тарифы на страховые взносы работодателей установлены статьей 34 НК РФ и составляют: ОПС (обязательное пенсионное страхование) – 22%, ВНиМ (страхование временной нетрудоспособности и в связи с материнством) – 2,9%, ОМС (обязательное медицинское страхование) – 5,1%.

Общая сумма отчислений на страховые взносы составит 259578 рублей.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Общая сумма затрат является суммой: затрат на оплату труда включая налоги и сборы, затраты на закупку оборудования и лицензионного ПО и составляет 2050078 рублей.

В процессе эксплуатации пользователи получают мобильный программно-аппаратный комплекс и техническую поддержку от команды разработки.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

4. Безопасность жизнедеятельности. Эргономические требования к рабочему месту программиста.

Проектирование рабочих мест, снабженных мониторами, относится к числу важных проблем эргономического проектирования в области вычислительной техники.

Рабочее место и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия: оптимальное размещение оборудования, входящего в состав рабочего места и достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения.

Эргономическими аспектами проектирования рабочих мест программиста, являются:

- высота рабочей поверхности;
- размеры пространства для ног;
- требования к расположению документов на рабочем месте (наличие и размеры подставки для документов, возможность различного размещения документов, расстояние от глаз пользователя до экрана, документа, клавиатуры и т.д.);
- характеристики рабочего кресла;
- требования к поверхности рабочего стола;
- регулируемость элементов рабочего места.

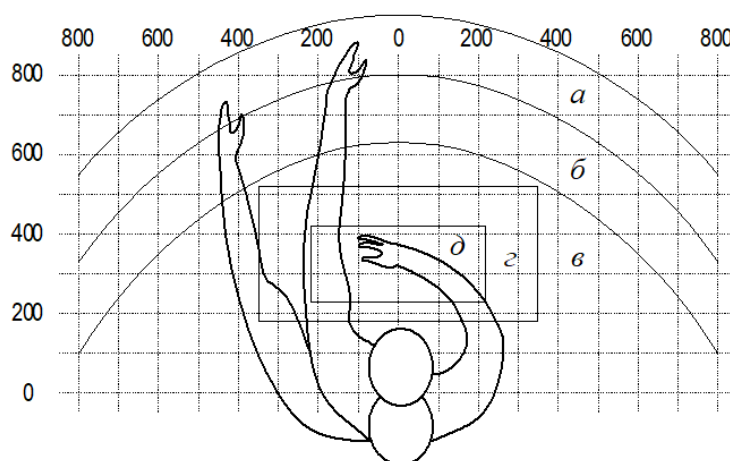
Главными элементами рабочего места программиста являются стол и кресло. Основным рабочим положением является положение сидя. Рабочая поза сидя вызывает минимальное утомление программиста.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации.

То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства (рисунок 4.1). Рабочие зоны делятся на следующие виды:

- моторное поле - пространство рабочего места, в котором могут осуществляться двигательные действия человека,
- максимальная зона досягаемости рук - это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе,
- оптимальная зона - часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом.



- а - зона максимальной досягаемости;
- б - зона досягаемости пальцев при вытянутой руке;
- в - зона легкой досягаемости ладони;
- г - оптимальное пространство для грубой ручной работы;
- д - оптимальное пространство для тонкой ручной работы;

Рисунок 4.1 – Зоны досягаемости в рабочем пространстве программиста.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Оптимальное размещение предметов труда и документации в зонах досягаемости:

- монитор размещается в зоне а (в центре);
- системный блок размещается в предусмотренной нише стола;
- клавиатура - в зоне г/д;
- манипулятор типа «мышь» - в зоне в справа;
- планшет в зоне, а/б (слева);
- МФУ находится в зоне а (справа);
- документация необходимая при работе - в зоне легкой досягаемости ладони.

В выдвижных ящиках стола - литература, неиспользуемая постоянно.

Для комфортной работы стол должен удовлетворять следующим условиям:

- высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;
- нижняя часть стола должна быть сконструирована так, чтобы программист мог удобно сидеть, не был вынужден поджимать ноги;
- поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения программиста;
- конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, канцелярских принадлежностей);
- высота рабочей поверхности рекомендуется в пределах 680-760 мм. Высота поверхности, на которую устанавливается клавиатура, должна быть около 650 мм.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Большое значение придается характеристикам рабочего кресла. Так, рекомендуемая высота сиденья над уровнем пола находится в пределах 420-550 мм. Поверхность сиденья мягкая, передний край закругленный, а угол наклона спинки - регулируемый.

Необходимо предусматривать при проектировании возможность различного размещения документов: сбоку от монитора, между монитором и клавиатурой и т.п. Кроме того, в случаях, когда монитор имеет низкое качество изображения, например заметны мелькания, расстояние от глаз до экрана делают больше (около 700 мм), чем расстояние от глаза до документа (300-450 мм). Вообще при высоком качестве изображения на видеотерминале расстояние от глаз пользователя до экрана, документа и клавиатуры может быть равным.

Положение экрана определяется должно удовлетворять следующим требования для максимально комфортного положения программиста:

- расстоянием считывания (0,6...0,7 м);
- углом считывания, направлением взгляда на 20° ниже горизонтали к центру экрана, причем экран перпендикулярен этому направлению. Должна также предусматриваться возможность регулирования экрана:
 - по высоте +3 см;
 - по наклону от -10° до $+20^\circ$ относительно вертикали;
 - в левом и правом направлениях.

Большое значение также придается правильной рабочей позе пользователя. При неудобной рабочей позе могут появиться боли в мышцах, суставах и сухожилиях. Требования к рабочей позе пользователя монитора следующие:

- голова не должна быть наклонена более чем на 20° ,

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

- плечи должны быть расслаблены,
- локти - под углом $80^{\circ}\dots 100^{\circ}$,
- предплечья и кисти рук - в горизонтальном положении.

Причина неправильной позы пользователей обусловлена следующими факторами: нет хорошей подставки для документов, клавиатура находится слишком высоко, а документы - низко, некуда положить руки и кисти, недостаточно пространство для ног.

В целях преодоления указанных недостатков даются общие рекомендации: лучше передвижная клавиатура; должны быть предусмотрены специальные приспособления для регулирования высоты стола, клавиатуры и монитора, а также подставка для рук.

Существенное значение для производительной и качественной работы на компьютере имеют размеры знаков, плотность их размещения, контраст и соотношение яркостей символов и фона экрана. Если расстояние от глаз оператора до экрана дисплея составляет 60...80 см, то высота знака должна быть не менее 3мм, оптимальное соотношение ширины и высоты знака составляет 3:4, а расстояние между знаками – 15...20% их высоты.

Соотношение яркости фона экрана и символов - от 1:2 до 1:15.

Во время пользования компьютером медики советуют устанавливать монитор на расстоянии 50-60 см от глаз. Специалисты также считают, что верхняя часть видеодисплея должна быть на уровне глаз или чуть ниже. Когда человек смотрит прямо перед собой, его глаза открываются шире, чем когда он смотрит вниз. За счет этого площадь обзора значительно увеличивается, вызывая обезвоживание глаз. К тому же если экран установлен высоко, а глаза широко открыты, нарушается функция моргания. Это значит, что глаза не закрываются полностью, не омываются

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

слезной жидкостью, не получают достаточного увлажнения, что приводит к их быстрой утомляемости.

Создание благоприятных условий труда и правильное эстетическое оформление рабочих мест на производстве имеет большое значение, как для облегчения труда, так и для повышения его привлекательности, положительно влияющей на производительность труда.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения дипломного проекта была изучена предметная область, затем сформулировано техническое задание на разработку программно-аппаратного комплекса для обработки и хранения видео контента.

Был произведен анализ и подбор стека технологий и инструментов для реализации проекта. В результате чего в единый комплекс удалось собрать аппаратную, программную часть проекта с интеграцией связи с облачным хранилищем видео контента. Комплекс реализован в соответствии с техническим заданием.

В разделе безопасности жизнедеятельности были описаны эргономические требования к рабочему месту программиста.

В экономическом разделе был произведен расчет трудоемкости и затрат на разработку комплекса.

Задание на дипломное проектирование выполнено в полном объеме.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Белов В. А. Основы программирования *Python*. - М.: Изд-во «ИНТУИТ», 2018. – 488 с.
2. Брин Ф. Основы встроенных операционных систем.- М: «Символ-Плюс», 2019 г. – 304 с.
3. Виргаев К. С. Облачные хранилища настоящее и перспективы. -М.: Лори, 2014.- 880 с.
4. Вирт Н. Алгоритмы и структуры данных. - Спб: Невский диалект, 2017 г.- 352 с.
5. Гиббонз П. Фреймворк *Flask* для программистов. - СПб.: Питер, 2018.- 336 с.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А

(обязательное)

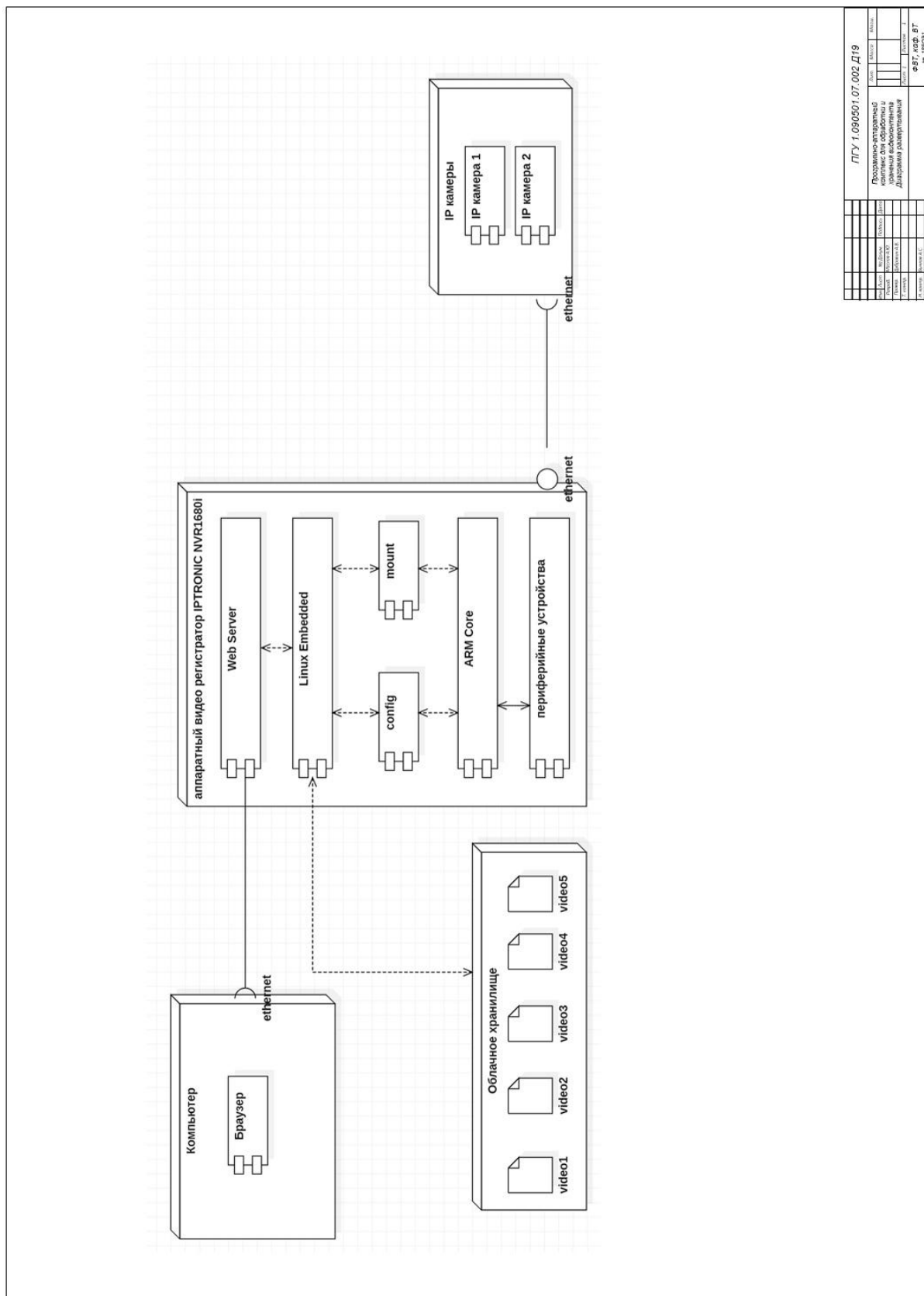
Программно-аппаратный комплекс для обработки и хранения

видеоко́нтекста

UML-диаграммы

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

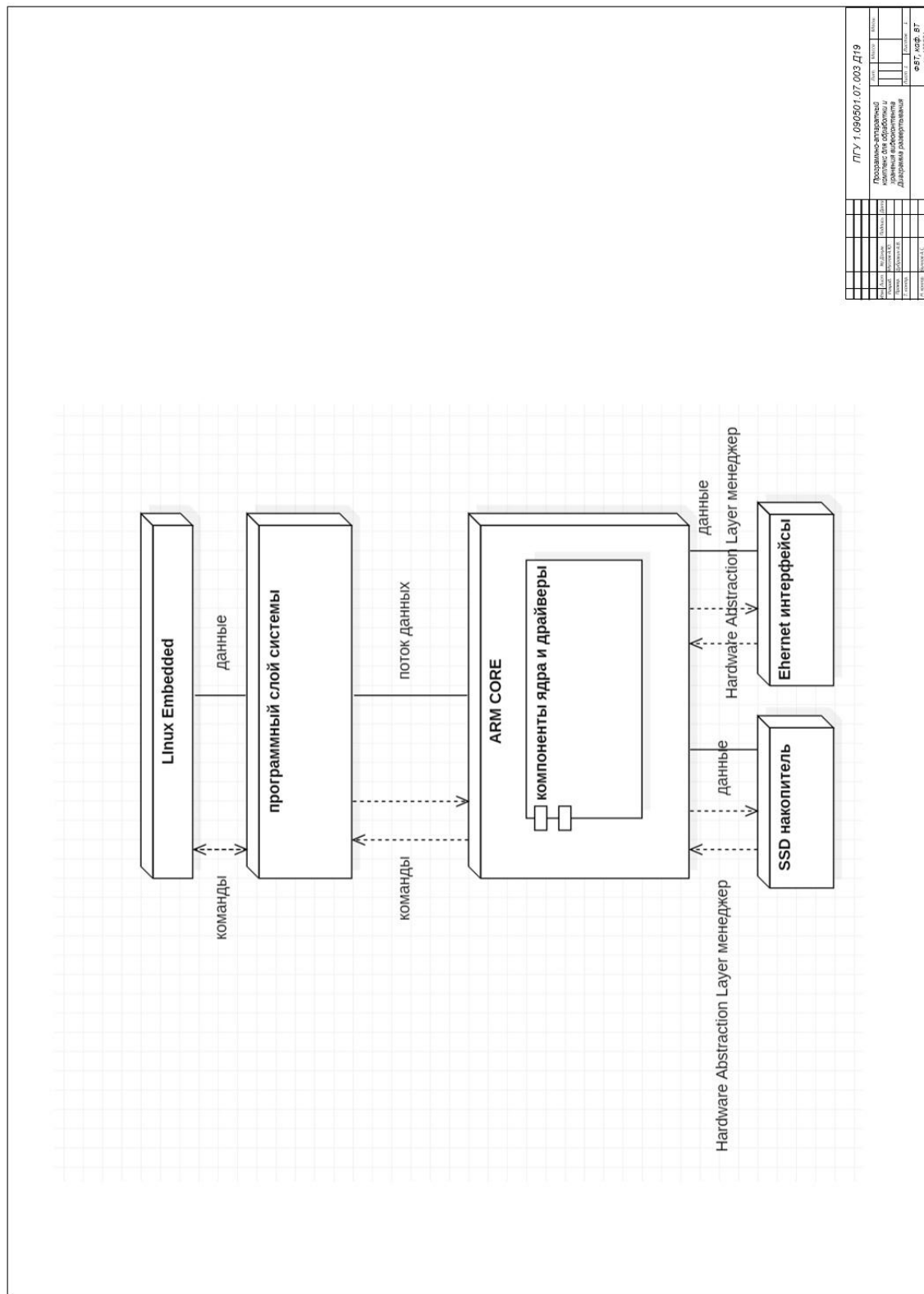
Приложение А.1 – Диаграмма развертывания



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.07.001 ПЗ

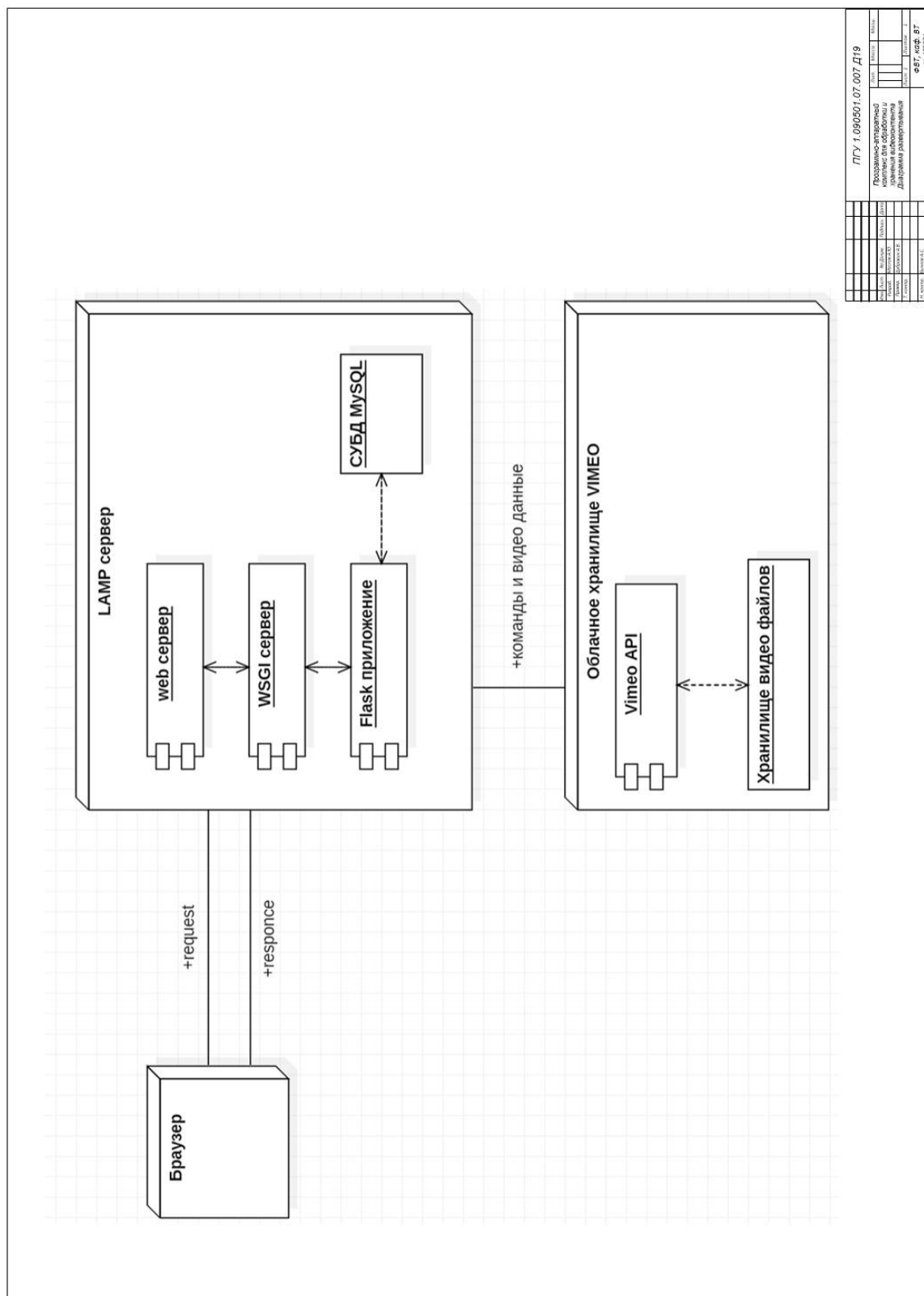
Приложение А.2 – Диаграмма развертывания



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.07.001 ПЗ

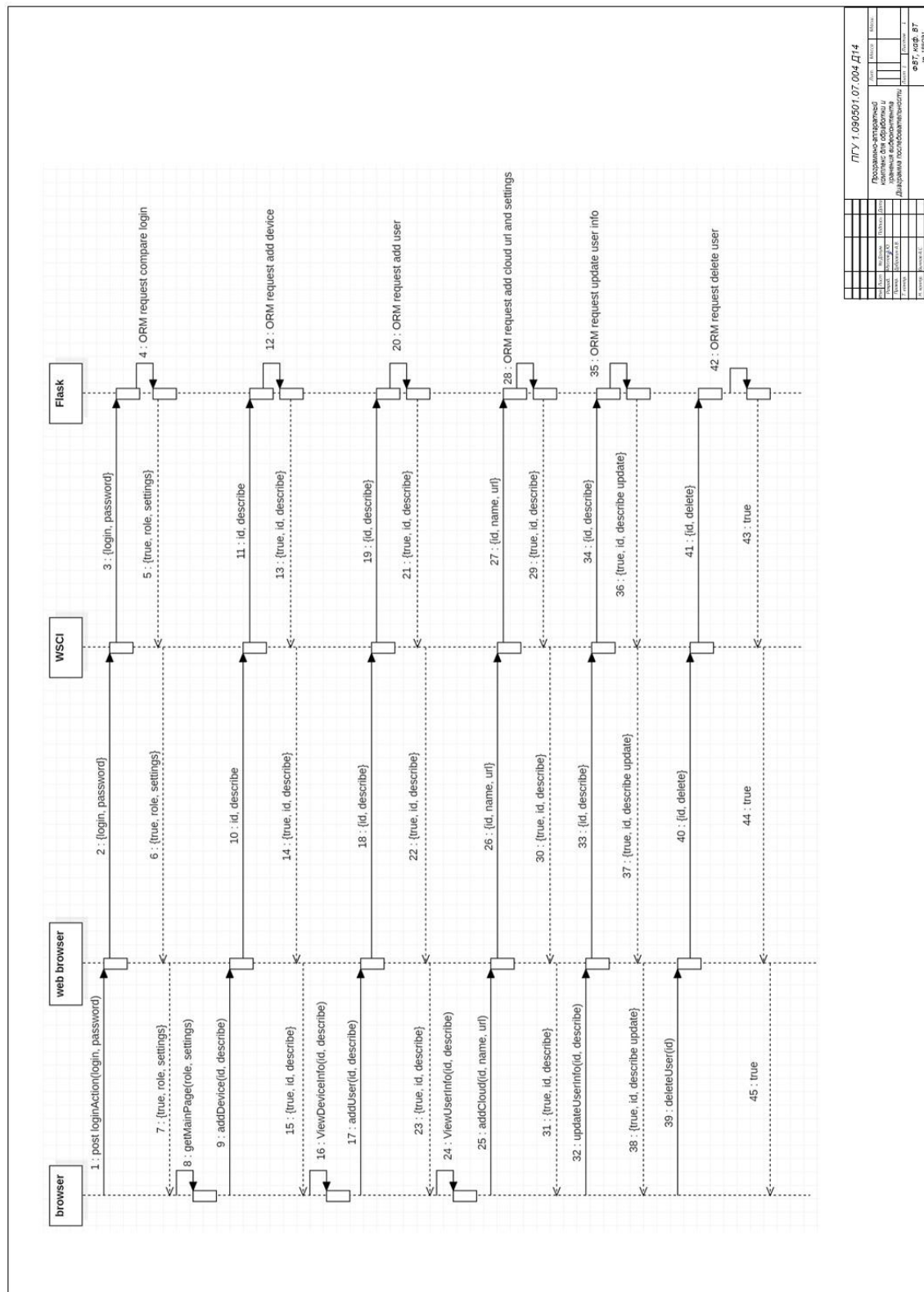
Приложение А.3 – Диаграмма развертывания



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.07.001 ПЗ

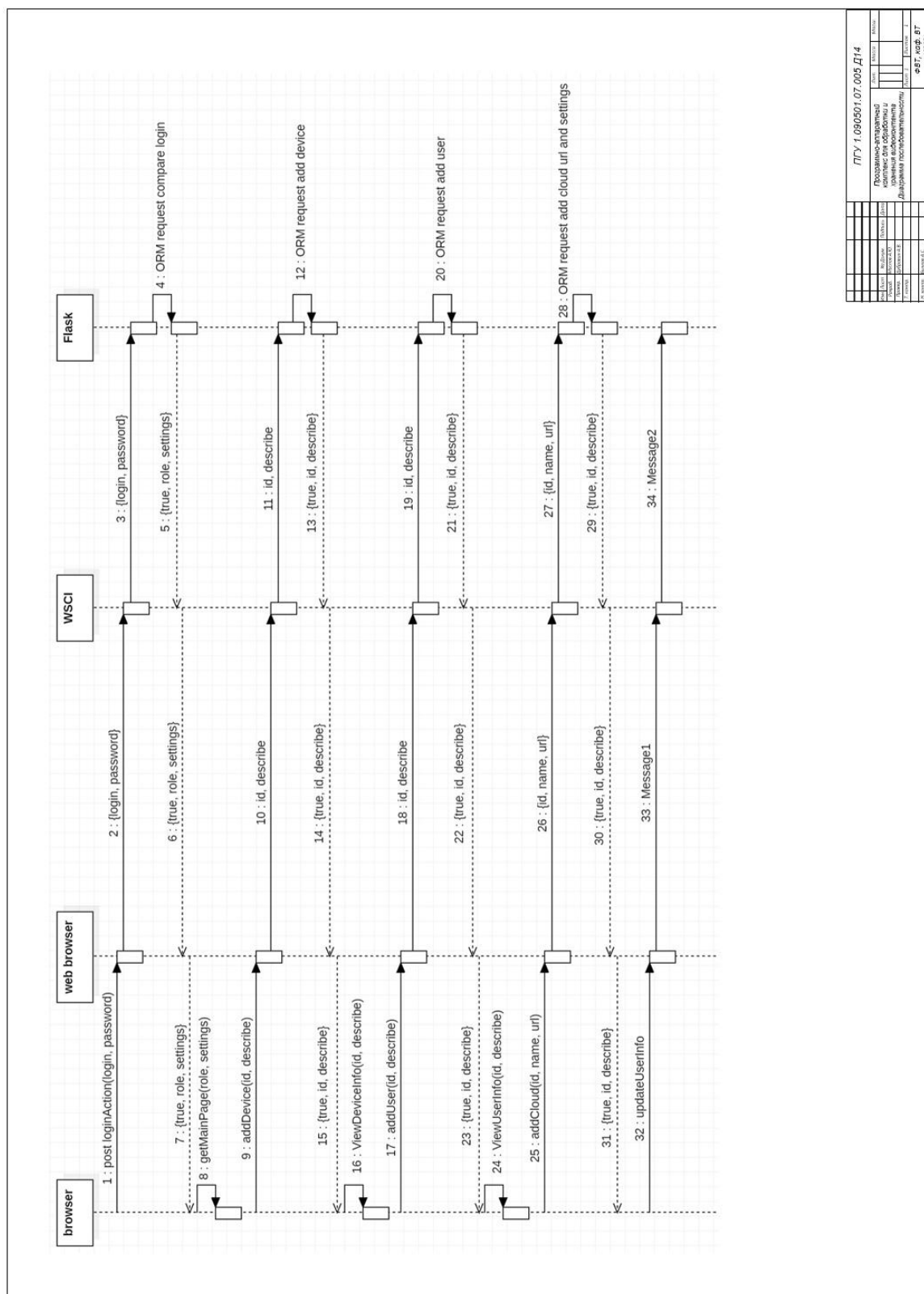
Приложение А.4 – Диаграмма последовательности



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.07.001 ПЗ

Приложение А.5 – Диаграмма последовательности

[illegible]

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.07.001 ПЗ

Приложение Б

(обязательное)

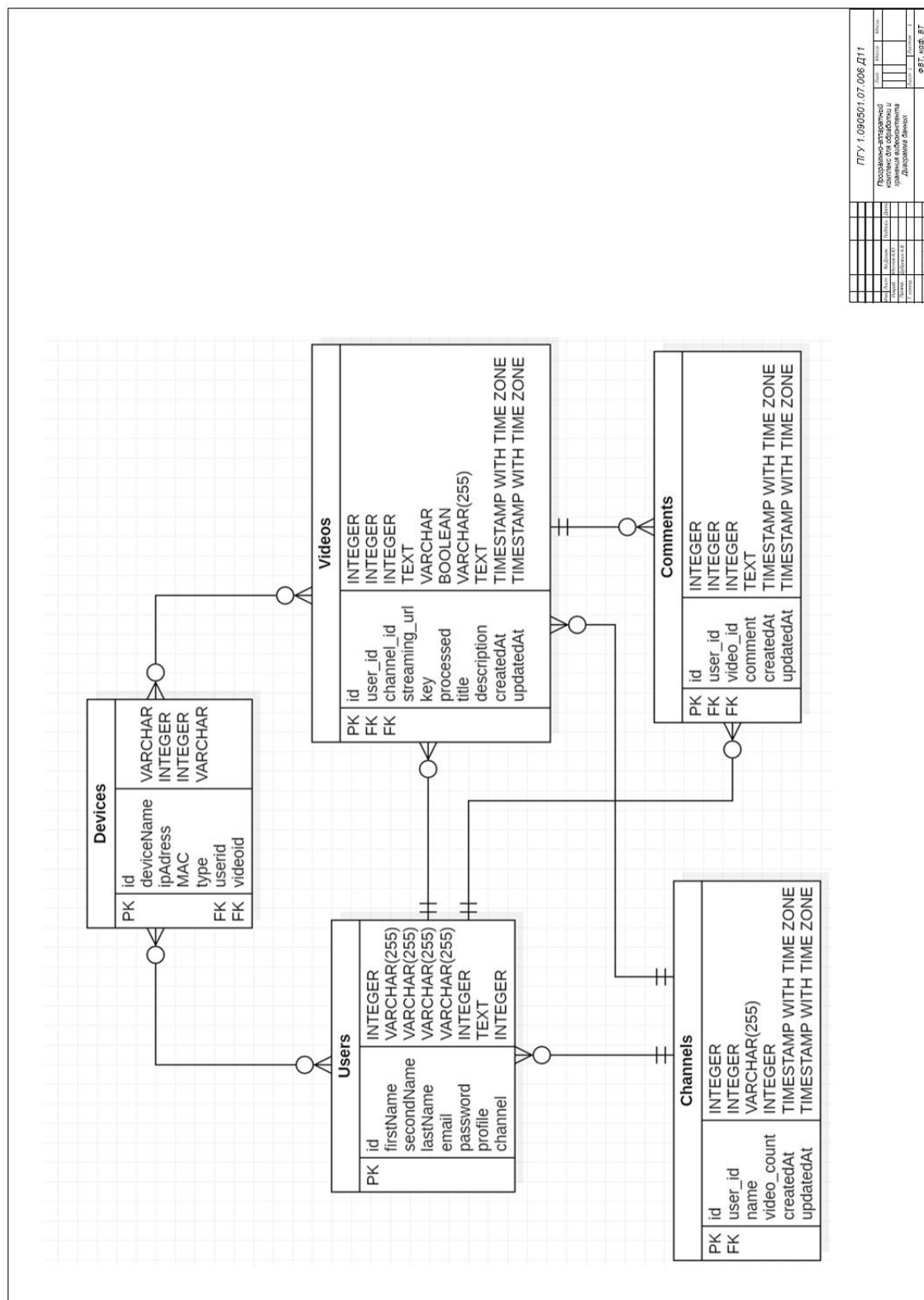
Программно-аппаратный комплекс для обработки и хранения

видеоко́нтен́та

Физическая модель базы данных

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение Б.1 – Физическая модель базы данных



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.07.001 ПЗ

Приложение В

(обязательное)

Программно-аппаратный комплекс для обработки и хранения
видеокартента

Листинг программы

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

Модуль videowork.py

import bs4
import os
import requests
import urllib.request
from urllib.parse import urlparse, parse_qs


from vimeo import VimeoClient


class VideoWork:

    def __init__(self, access_token: str=None, client_id: str=None, client_secret:
str=None, user_id: int=None):

        self.ACCESS_TOKEN = access_token
        self.CLIENT_ID = client_id
        self.CLIENT_SECRET = client_secret
        self.USER_ID = user_id


        self.py_vimeo_client = None


        self.response_code = None


        self.REQUEST_HEADERS = {
            "Authorization": f"bearer {self.ACCESS_TOKEN}",
            "Content-Type": "application/json",

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

        "Accept": "application/vnd.vimeo.*+json;version=3.4"
    }

def _get_or_set_py_vimeo_client(self):

    if not self.CLIENT_ID or not self.CLIENT_SECRET:
        raise NotImplementedError

    if self.py_vimeo_client:
        return self.py_vimeo_client

    self.py_vimeo_client = VimeoClient(
        token=self.ACCESS_TOKEN,
        key=self.CLIENT_ID,
        secret=self.CLIENT_SECRET
    )

    return self.py_vimeo_client

def _set_response_code(self, status_code=None):
    if not status_code:
        self.response_code = None
    else:
        self.response_code = status_code

def upload_video(self, file_path: str, video_name: str=None,
video_description: str=None, settings: dict={}) -> str:

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

str

str

str

dict

["https://maslovpnz.vimeo.com/api/reference/videos?version=3.4"](https://maslovpnz.vimeo.com/api/reference/videos?version=3.4)

```
client = self._get_or_set_py_vimeo_client()
```

```
video_uri = client.upload(
```

$$file_path,$$
$$data = \{$$

```
'name': video_name,
```

'description': video_description,

```
'content_rating': ['safe'],
```

'privacy': {

'download': False,

'embed': 'whitelist',

'comments': 'nobody',

'view': 'disable',

 $\}$

```
'review_page': {
```

'active': False,

$$\}$$

} if not settings else settings

)

```
self._set_response_code(200 if video_uri else None)
```

```
return video_uri
```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```
def upload_picture(self, video_uri: str, file_path: str) -> dict:
```

```
    video_uri        str
```

```
    file_path        str
```

Returns a dictionary like this:

```
{
```

```
    'uri': 'maslovpnz.vimeo.com//videos/',
```

```
    'link':
```

```
'https://maslovpnz.vimeo.com//videos/1329492536?expires=1639941754&sig=4c66f8accf778d4bb0a06852e8d301ab145f6139',
```

```
    'active': True
```

```
}
```

```
local_file = None
```

```
if file_path.startswith("http"):
```

```
    local_file = os.path.basename(file_path)
```

```
    file_path, _ = urllib.request.urlretrieve(file_path, local_file)
```

```
client = self._get_or_set_py_vimeo_client()
```

```
upload_response = client.upload_picture(video_uri, file_path,  
activate=True)
```

```
if local_file and os.path.exists(local_file):
```

```
    os.remove(local_file)
```

```
return upload_response
```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```
def get_video(self, video_uri: str) -> dict:
```

```
    video_uri          str
```

```
    video_id = video_uri.split("/").pop()
```

```
    response = requests.get(
        f"https://api.vimeo.com/videos/{video_id}",
        headers=self.REQUEST_HEADERS,
    )
    self._set_response_code(response.status_code)
    return response.json()
```

```
def get_common_video_information(self, video_uri: str) -> dict:
```

```
    video = self.get_video(video_uri)
    return {
        "status": video['transcode']['status'],
        "is_playable": video['is_playable'],
        "link": video['link'],
        "duration": video['duration'],
        "width": video['width'],
        "height": video['height'],
    }
```

```
def change_video_content_rating(self, video_uri: str, rating: str='safe') ->
int:
```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

    if rating not in ['violence', 'drugs', 'language', 'nudity', 'advertisement',
'safe', 'unrated']:

```

```

    raise NotImplementedError

```

```

    video_id = video_uri.split("/").pop()

```

```

    data = {

```

```

        "content_rating": [rating],

```

```

    }

```

```

    response = requests.patch(

```

```

        f"https://api.vimeo.com/videos/{video_id}",

```

```

        headers=self.REQUEST_HEADERS,

```

```

        json=data,

```

```

    )

```

```

    self._set_response_code(response.status_code)

```

```

    return response.json()

```

```

def pull_video_from_url(self, download_url: str, video_name: str, folder_uri:
str=None, file_size=None, settings: dict={}, logo_link: str="") -> str:

```

```

    download_url      string

```

```

    video_name        string

```

```

    folder_uri        str

```

```

    fize_size          int

```

```

    settings           dict

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

if not file_size:
    response = requests.head(download_url, allow_redirects=True)
    file_size = response.headers['Content-Length'] if 'Content-Length' in
response.headers else 0

```

```

data = {
    'name': video_name,
    'description': "",
    "upload": {
        "approach": "pull",
        "size": file_size,
        "link": download_url,
    },
    'content_rating': ['safe'],
    'privacy': {
        'download': False,
        'embed': 'whitelist',
        'comments': 'nobody',
        'view': 'disable',
    },
    'review_page': {
        'active': False,
    },
    'embed': {
        'buttons': {
            'embed': False,

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

        'fullscreen': True,
        'hd': True,
        'share': False,
        'watchlater': False,
    },
    'color': '#feeff0',
    'logos': {
        'custom': {
            'active': True,
            'link': logo_link if logo_link else None,
            'sticky': True,
        },
        'vimeo': False,
    },
    'playbar': True,
    'title': {
        'name': 'hide',
        'owner': 'hide',
        'portrait': 'hide',
    },
    'volume': True,
},
'folder_uri': folder_uri if folder_uri else None,
} if not settings else settings

```

```

response = requests.post(
    f"https://api.vimeo.com/me/videos",

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

        headers=self.REQUEST_HEADERS,
        json=data,
    )

    self._set_response_code(response.status_code)

    response = response.json()
    video_uri = response['uri']

    return video_uri

def update_video_title(self, video_uri: str, title: str) -> int:

    data = {
        "name": title,
    }

    response = requests.patch(
        f"https://api.vimeo.com{video_uri}",
        headers=self.REQUEST_HEADERS,
        json=data,
    )
    self._set_response_code(response.status_code)
    return response.status_code

def delete_video(self, video_uri: str) -> int:

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

response = requests.delete(
    f"https://api.vimeo.com{video_uri}",
    headers=self.REQUEST_HEADERS,
)
self._set_response_code(response.status_code)
return response.status_code

def create_folder(self, folder_name: str) -> str:

    data = {
        "name": folder_name,
    }
    response = requests.post(
        f"https://api.vimeo.com/users/{self.USER_ID}/projects",
        headers=self.REQUEST_HEADERS,
        json=data,
    )

    self._set_response_code(response.status_code)
    response = response.json()
    folder_uri = response['uri']
    return folder_uri

def update_folder_name(self, folder_uri: str, new_folder_name: str) -> str:

    data = {
        "name": new_folder_name,

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }

    response = requests.patch(
        f"https://api.vimeo.com{folder_uri}",
        headers=self.REQUEST_HEADERS,
        json=data,
    )
    self._set_response_code(response.status_code)
    response = response.json()
    return folder_uri

def delete_folder(self, folder_uri: str, delete_all_videos_in_folder:
bool=False) -> str:

    data = {
        "should_delete_clips": delete_all_videos_in_folder
    }
    response = requests.delete(
        f"https://api.vimeo.com{folder_uri}",
        headers=self.REQUEST_HEADERS,
        json=data,
    )
    self._set_response_code(response.status_code)
    return folder_uri

def remove_video_from_folder(self, folder_uri: str, video_uri: str) -> str:

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

video_id = video_uri.split("/").pop()
response = requests.delete(
    f"https://api.vimeo.com{folder_uri}/videos/{video_id}",
    headers=self.REQUEST_HEADERS,
)
self._set_response_code(response.status_code)
return folder_uri

def add_video_to_folder(self, folder_uri: str, video_uri: str) -> int:

    video_id = video_uri.split("/").pop()
    response = requests.put(
        f"https://api.vimeo.com{folder_uri}/videos/{video_id}",
        headers=self.REQUEST_HEADERS,
    )
    self._set_response_code(response.status_code)
    return response.status_code

def tag_video(self, video_uri: str, tag: str) -> int: # ie 200 OK

    video_id = video_uri.split("/").pop()

    response = requests.put(
        f"https://api.vimeo.com/videos/{video_id}/tags/{tag}",
        headers=self.REQUEST_HEADERS,
    )
    self._set_response_code(response.status_code)

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```
return response.status_code
```

```
def remove_tag_from_video(self, video_uri: str, tag: str) -> int: # ie 204 No  
Content
```

```
video_id = video_uri.split("/").pop()
```

```
response = requests.delete(
```

```
    f"https://api.vimeo.com/videos/{video_id}/tags/{tag}",
```

```
    headers=self.REQUEST_HEADERS,
```

```
)
```

```
self._set_response_code(response.status_code)
```

```
return response.status_code
```

```
def domain_whitelist_video(self, video_uri: str, domain: str) -> int: # ie 204  
OK
```

```
video_id = video_uri.split("/").pop()
```

```
response = requests.put(
```

```
    f"https://api.vimeo.com/videos/{video_id}/privacy/domains/{domain}",
```

```
    headers=self.REQUEST_HEADERS,
```

```
)
```

```
self._set_response_code(response.status_code)
```

```
return response.status_code
```

```
def get_video_hash(self, video_uri: str) -> str:
```

```
vimeo_video = self.get_video(video_uri)
```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

try:
    iframe = vimeo_video['embed']['html']
    soup = bs4.BeautifulSoup(iframe, "html.parser")
    src = soup.find("iframe").get("src").replace("&", "&")
    parsed_url = urlparse(src)
    hash = parse_qs(parsed_url.query)['h'][0]
except Exception:

return hash

```

Модуль form.py

```

from app.models import User
from flask_login import current_user
from flask_wtf import FlaskForm
from flask_wtf.file import FileField, FileAllowed, FileRequired
from wtforms import StringField, PasswordField, BooleanField, SubmitField,
IntegerField, TextField
from wtforms.validators import ValidationError, DataRequired, Email, EqualTo,
Length

class RegistrationForm(FlaskForm):
    user_name = StringField('Username', validators=[
        DataRequired(), Length(min=2, max=20)])
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[
        DataRequired(), Length(min=8)])
    confirm_password = PasswordField('Confirm Password', validators=[
        DataRequired(), Length(min=8), EqualTo('password')])
    age = IntegerField('Age', validators=[DataRequired()])

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

address = TextField('Address', validators=[
    DataRequired(), Length(min=4, max=40)])
submit = SubmitField('Sign Up')

def validate_username(self, user_name):

    user = User.query.filter_by(user_name=user_name.data).first()
    if user is not None:
        raise ValidationError('.')
def validate_email(self, email):

    user = User.query.filter_by(email=email.data).first()
    if user is not None:
        raise ValidationError('Please use a different email address.')
class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[
        DataRequired(), Length(min=8)])
    remember_me = BooleanField('Remember Me')
    submit = SubmitField('Sign In')
class UpdateAccountForm(FlaskForm):
    user_name = StringField('Username', validators=[
        DataRequired(), Length(min=2, max=20)])
    email = StringField('Email', validators=[DataRequired(), Email()])
    avatar = FileField('Update Profile Picture', validators=[
        FileAllowed(['jpg', 'png', 'jpeg'])])
    submit = SubmitField('Update')

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

def validate_username(self, user_name):

    if user_name.data != current_user.user_name:
        user = User.query.filter_by(user_name=user_name.data).first()
        if user is not None:
            raise ValidationError('.')

def validate_email(self, email):

    if email.data != current_user.email:
        user = User.query.filter_by(email=email.data).first()
        if user is not None:
            raise ValidationError('.')

class VideoUploadForm(FlaskForm):
    video_title = TextField('Title', validators=[
        DataRequired(), Length(max=60)])
    description = TextField('Description', validators=[DataRequired()])
    category = StringField('Category', validators=[
        DataRequired(), Length(max=20)])
    video_content = FileField('Video', validators=[FileAllowed(
        ['mp4', 'mkv', '3gp', 'mov']), FileRequired()])
    submit = SubmitField('Upload')

class UpdateVideoForm(FlaskForm):
    video_title = TextField('Title', validators=[

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

        DataRequired(), Length(max=60)])

description = TextField('Описания', validators=[DataRequired()])
category = StringField('Устройства', validators=[
        DataRequired(), Length(max=20)])

submit = SubmitField('Update')

class CommentForm(FlaskForm):
    body = StringField("", validators=[DataRequired(), Length(max=400)])
    submit = SubmitField('Комментарий')

```

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение Г

(обязательное)

Программно-аппаратный комплекс для обработки и хранения
видеокартента

Презентация

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Программно-аппаратный комплекс для обработки и хранения видео контента

Автор работы:

студент гр. 18ВОЭ1 Маслов Артем Юрьевич

Научный руководитель:

к.т.н., доцент Дубравин Алексей Викторович

Пенза, 2023

Цель работы и объекты

Цель работы – Разработать мобильный программно аппаратный комплекс видеонаблюдения за проектными работами в сфере девелопмента.

Объектом проекта является программно аппаратный комплекс, выполняющие следующие функции:

- видеорегистрация заданного пространства;
- временное хранение файлов видеорегистрации на внутреннем носителе;
- постоянное хранение видео информации в облачном хранилище;
- наличие клиентской программной части комплекса для управления аппаратными средствами и видео контентом.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Задачи решаемые при разработке

- Выбрать стек технологий для реализации проекта.
- Сформировать аппаратную платформу проекта, путем выбора видеорегистратора и инсталляция на данную платформу ОС Linux Embedded.
- Настройка ОС Linux Embedded для работы м периферийными устройствами аппаратуры, инсталляция драйверов и сетевых настроек.
- Реализация программной части проекта.
- Подключение облачного сервиса к программной части проекта.
- Тестирование комплекса.

Сравнение аналогов

комплекс Okta Vision

комплекс HD-TVI Hiwatch

комплекс Tantos Amelie

Ограничением данных комплексов является то, что они предлагают готовые решения, изменить логику работы которых может только производитель или специализированный центр.
Логикой такого подхода является максимальная коммерциализация проекта.



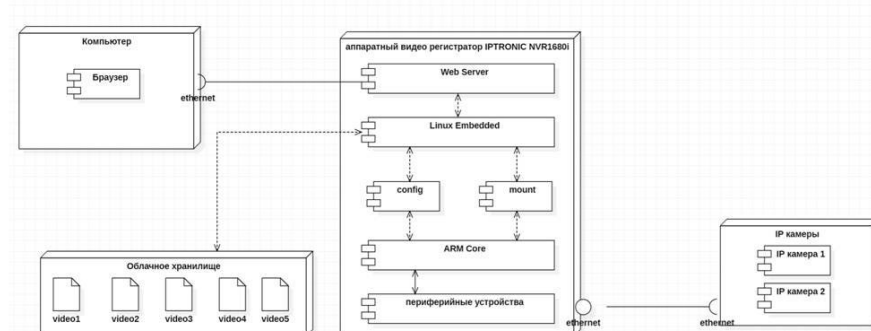
Целью дипломного проекта является разработка бюджетного варианта программно-аппаратного комплекса для обработки и хранения видео контента с функцией видеонаблюдения на базе открытых и доступных решений.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Стек технологий

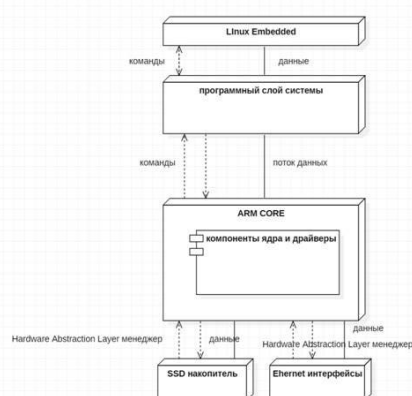


Программно-аппаратный комплекс

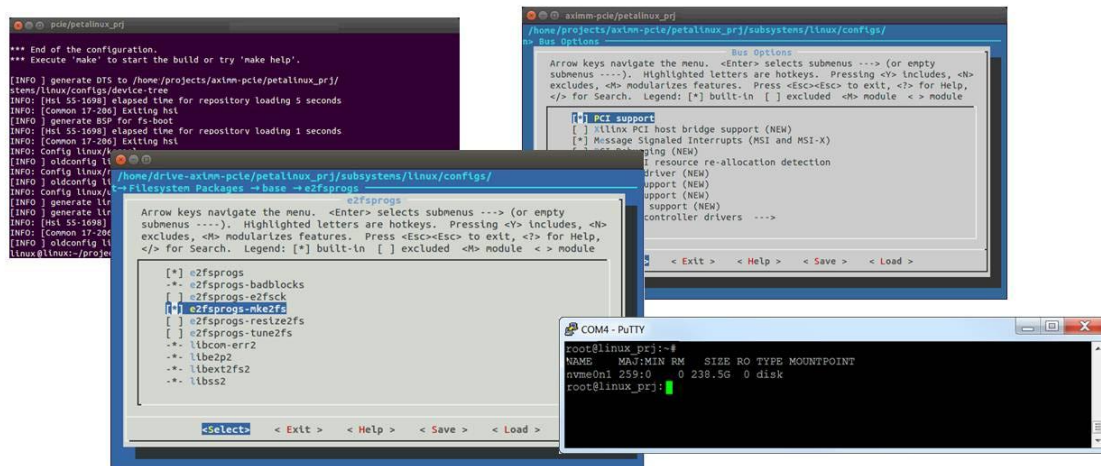


					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Аппаратная часть комплекса



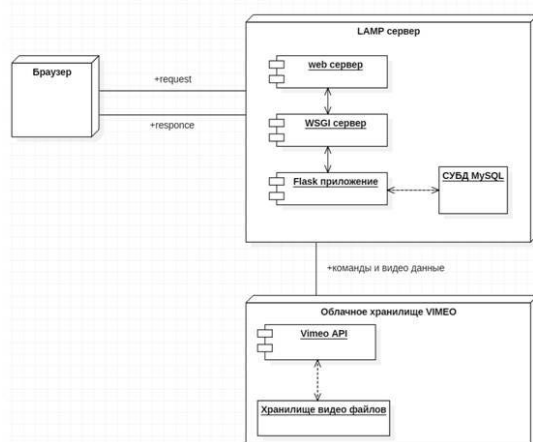
Формирование образа ОС Linux Embedded



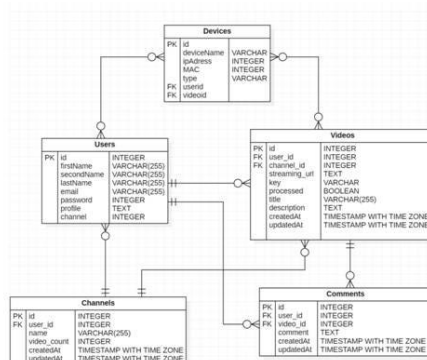
ПГУ1.090501.07.001 ПЗ

Изм. Лист № докум. Подпись Дата

Программная часть комплекса



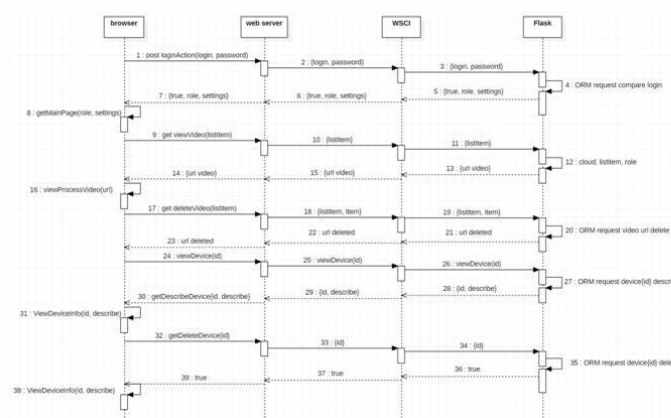
База данных приложения



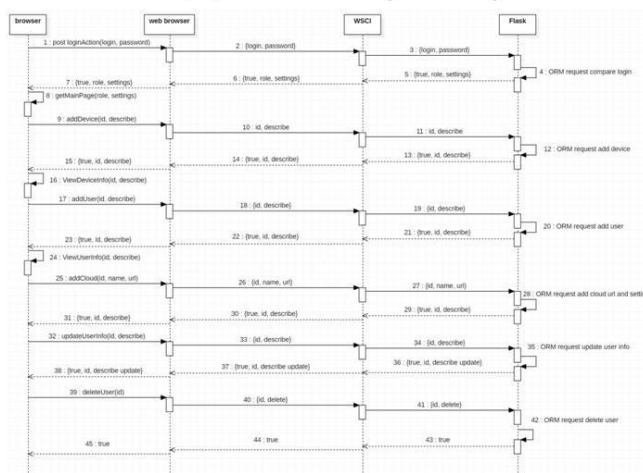
ПГУ1.090501.07.001 ПЗ

Изм. Лист № докум. Подпись Дата

Работа приложения в режиме пользователь



Работа приложения в режиме администратор



ПГУ1.090501.07.001 ПЗ

Изм. Лист № докум. Подпись Дата

Скриншоты запущенного приложения

Главная страница администратора admin1 Выход

Количество пользователей: 2
Количество подключенных устройств: 2
Количество видео файлов: 232

Информация об устройствах

№	Подписанное устройство	Действие	IP-адрес устройств
1	HOST_Stream1	<input type="button" value="Удалить"/> <input type="button" value="Редактировать"/>	192.168.23.5
2	HOST_Stream3	<input type="button" value="Удалить"/> <input type="button" value="Редактировать"/>	192.168.73.9

Информация о пользователях

№	Логин	Действие	Описание
1	user2	<input type="button" value="Удалить"/> <input type="button" value="Редактировать"/>	Пользователь
2	admin1	<input type="button" value="Удалить"/> <input type="button" value="Редактировать"/>	Администратор

Информация о записях

№	Хранилище	Устройство	Действие
1	WS_hosh_19	PVNO_Stream1	<input type="button" value="Удалить"/> <input type="button" value="Просмотр"/>
2	video_room29.05.23	PVNO_Stream1	<input type="button" value="Удалить"/> <input type="button" value="Просмотр"/>

Просмотр записи video_room29.05.23 Назад



Создание устройства Назад

Информация о устройстве

Имя устройства:

IP-адрес:

Адрес хранилища:

Пользователь:

Результаты работы

- Разработан программно-аппаратный комплекс полностью соответствующий техническому заданию;
- Проведено функциональное тестирование (успешно);
- Намечены задачи по развитию функциональных возможностей программы.

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

Спасибо за внимание!

					ПГУ1.090501.07.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		