Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Операционные среды и системное программирование»

*К защите допустить:*

И. о. заведующего кафедрой информатики

_____ С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

# КЛИЕНТЫ СЕТЕВЫХ СЛУЖБ: HTTP

БГУИР КП 1-40 04 01 028 ПЗ

Студент А. К. Хрищанович

Руководитель Н. Ю. Гриценко

Нормоконтролер Н. Ю. Гриценко

Минск 2024

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

В мире современное разработки программного обеспечения, взаимодействие между различными приложениями через интерфейс стало неотъемлемой частью разработки. Одним из наиболее распространенных протоколов, лежащих в основе сетевых служб, является протокол HTTP. HTTP обеспечивает передачу данных между клиентом и сервером, позволяя пользователям получать доступ к веб-ресурсам, обмениваться информацией и взаимодействовать с веб-приложениями.

Цель данного курсового проекта заключается в создании собственного браузера. Создание собственного браузера является важным шагом к пониманию принципов работы веб-технологий и сетевого программирования, а также позволяет глубже погрузиться в тему веб-разработки.

Задачи курсового проекта на тему «Клиенты сетевых служб: HTTP» включает в себя:

1 Разработка пользовательского интерфейса: создание удобного и интуитивно понятного пользовательского интерфейса для браузера, включая элементы управления, такие как адресная строка, кнопки навигации, возможности создания закладок и просмотра истории поиска.

2 Использование механизма для загрузки и отображения веб-страниц: использование уже существующего решения, такого как PyQt5, для отображения веб-страниц.

3 Обработка HTTP-запросов: реализация механизма для отправки HTTP-запросов на сервер и получения ответов. Это включает в себя работу с протоколом HTTP, управление сеансами и обработку ошибок.

4 Управление вкладками: добавление возможности открывать несколько веб-страниц в разных вкладках, а также закрывать ненужный вкладки для уменьшения визуального шума.

5 Управление историей поиска: возможность просмотреть существующую историю поиска, удалить историю полностью или удалить ее отдельный элемент, а также перейти по ссылке из истории.

6 Управление закладками: возможность создать закладку на определенном сайте для дальнейшего быстрого доступа к данному ресурсу.

Реализация этих задач позволит создать собственный браузер, который будет содержать необходимый базовый функционал для возможности создать свое персонализированное пространство для работы с веб-страницами.

# 1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

## 1.1 Структура и архитектура вычислительной системы

Вычислительная система – результат интеграции аппаратных и программных средства, функционирующих в единой системе и предназначенных для решения задач определенного класса. Структура вычислительной системы состоит из пяти уровней.[1]

### **1.1.1** Аппаратный уровень

Аппаратный уровень является первым уровнем вычислительной системы. Этот уровень определяется набором аппаратных компонентов и их характеристиками, используемых вышестоящими уровнями иерархии и определяющими воздействие на них. К физическим ресурсам этого уровня относятся: процессор, оперативная память, внешние устройства.

### **1.1.2** Уровень управления физическими ресурсами

Уровень управления физическими ресурсами является вторым уровнем вычислительной системы и первым уровнем системного программного обеспечения, что подчеркивает то, что это первый слой программного обеспечения, взаимодействующий с аппаратным уровнем. Его основное назначение заключается в систематизации и стандартизации правил программного использования физических ресурсов. На этом уровне обеспечивается создание программ управления физическими ресурсами. Для обеспечения управления физическими ресурсами используются программы, которые называются драйверами физического ресурса (устройства).

Драйвер физического устройства представляет собой программное обеспечение, разработанное на основе команд управления конкретным физическим устройством и предназначенное для организации работы с данным устройством.

### **1.1.3** Уровень управления логическими/виртуальными ресурсами

Логическое или виртуальное устройство представляет собой ресурс, характеристики которого реализованы программным образом. Драйвер логического или виртуального ресурса – это программное средство, обеспечивающее создание, управление и использование соответствующего ресурса. Эти компоненты ориентированы на конечного пользователя,

и их команды не зависят от конкретных физических устройств, что облегчает взаимодействие с ресурсами для программных компонентов.

На этом уровне могут создаваться новые логические ресурсы, и для организации драйвера логического устройства могут использоваться как драйверы физических, так и драйверы логических или виртуальных устройств. Система поддерживает иерархию драйверов, что позволяет эффективно управлять различными уровнями ресурсов.

Ресурсы вычислительной системы включают в себя как физические, так и виртуальные ресурсы. Они характеризуются конечностью, что создает конкуренцию между программными потребителями за доступ к этим ресурсам. Операционная система, как комплекс программ, управляет этими ресурсами, обеспечивая их эффективное использование.

Существует разветвленная иерархия виртуальных и физических устройств, и драйверы могут быть классифицированы в три группы:

1 Драйверы физических устройств: обеспечивают взаимодействие с конкретными физическими компонентами вычислительной системы.

2 Драйверы виртуальных устройств, обобщающих характеристики физических устройств: позволяют создавать абстракции над физическими устройствами, обобщая их характеристики и предоставляя более удобный интерфейс программам.

3 Драйверы виртуальных устройств без аппаратной реализации: эти драйверы создают «полностью» виртуальные устройства, не имеющие физического эквивалента. Примером может служить драйвер файловой системы, который управляет доступом к файлам без конкретной аппаратной реализации.

Такая иерархия и классификация драйверов позволяют эффективно организовывать управление ресурсами на уровне операционной системы и предоставлять пользователю удобные интерфейсы для работы с вычислительной системой.

### **1.1.4** Уровень систем программирования

Система программирования представляет собой комплекс программ, предназначенных для обеспечения эффективного управления жизненным циклом программного продукта в вычислительной системе. Жизненный цикл программы включает в себя пять ключевых этапов, к которым относятся проектирование программного продукта, кодирование или реализация, тестирование и отладка, а также внедрение программного продукта и его сопровождение.

Проектирование программного продукта представляет собой первый этап жизненного цикла программы. На этом этапе определяется структура, архитектура и основные характеристики будущего программного продукта. Кроме этого, на этапе проектирования определяются требования к программному продукту, его архитектура, проектирование интерфейса, выбор технологии и разработка алгоритмов для реализации.

Кодирование или реализация представляет собой продолжение этапа проектирования. На данном этапе программный продукт переходит из концепции в конкретный исполняемый код.

Тестирование представляет собой процесс проверки программы с использованием заранее определенных тестовых данных. Тесты, как заранее заданные входные данные, направлены на обеспечение корректного выполнения программы в различных сценариях использования. Тестирование также включает в себя оценку производительности и надежности программы.

Отладка – это процесс выявления, локализации и исправления ошибок, обнаруженных в процессе тестирования. Отладка направлена на обеспечение стабильной и надежной работы программы.

Этап внедрения включает в себя установку программного комплекса на объектную вычислительную систему и его первичную настройку. Внедрение также включает в себя проверку работоспособности программы в реальной среде.

В этап сопровождения входит исправление ошибок и недочетов, выявленных после внедрения программного комплекса. Сюда также включается выпуск патчей для улучшения функциональности или исправления обнаруженных проблем.

### **1.1.5** Уровень прикладных систем

На уровне прикладных систем реализуются программные системы, ориентированные на решение или автоматизацию решения задач из конкретной предметной области. Этот уровень представляет собой высокоуровневую часть программного обеспечения, которая нацелена на конечного пользователя и предоставляет интерфейс для взаимодействия с компьютерной системой.

Уровень прикладных систем представляет собой ключевой элемент компьютерной архитектуры, где программы разрабатываются и оптимизируются для конечных пользователей, сосредотачиваясь на решении конкретных задач в различных областях человеческой деятельности.

## 1.2 Структура и архитектура выбранной вычислительной системы для разработки курсового проекта

Вычислительная система, на которой проводилась разработка данного курсового проекта, базируется на операционной системе Windows 10 и аппаратном обеспечении ноутбука Asus из серии TUF Gaming. Эта система включает в себя следующие компоненты:

1 Процессор Intel Core i5: процессор (центральный процессор) является основным вычислительным движком компьютера. Данный процессор обеспечивает высокую производительность и является важной частью архитектуры системы.

2 Оперативная память (RAM): оперативная память служит для временного хранения данных, используемых программами вовремя их выполнения. Это важный аспект архитектуры, так как от объемы и скорости оперативной памяти зависит производительность системы.

3 Жестки диск: жесткий диск или SSD предназначен для хранения операционной системы, приложений и данных.

4 Дискретная видеокарта NVIDIA GeForce GTX 1650 GDDR на основе графического процессора TU117, специализируется на обработке графики и выполнении сложных вычислений, связанных с отображением изображений на экране.

На основе операционной системы Windows 10 и аппаратного обеспечения ноутбука Asus TUF Gaming с процессором Intel Core i5, оперативной памятью, жестким диском (или SSD) и видеокартой NVIDIA GeForce GTX 1650 GDDR6 можно сделать вывод о высокой производительности и графических возможностях данной вычислительной системы, что делает ее подходящей для разработки и выполнения требовательных задач.

### **1.2.1** Устройство центрального процессора

Процессор – мозг компьютера. Его задача – выполнять программы, находящиеся в основной памяти. Для этого он вызывает команды из памяти, определяет их тип, а затем выполняет одну за другой. Компоненты соединены шиной, представляющей собой набор параллельно связанных проводов для передачи адресов, данных и управляющих сигналов.

Процессор состоит из нескольких частей. Блок управления отвечает за вызов команд из памяти и определение их типа. Арифметика-логическое устройство выполняет арифметические и логические операции.

Тракт данных состоит из регистров, арифметико-логического устройства и нескольких соединительных шин. Содержимое регистров поступает во входные регистры арифметико-логического устройства. В них находятся входные данные, пока арифметико-логическое устройство производит вычисления. Тракт данных – важная составляющая часть всех компьютеров.[2]

В дополнении к процессору требуются по крайней мере некоторое оперативное запоминающее устройство, какой-то способ, позволяющий пользователю записать информацию в компьютер (устройство ввода), а также извлечь ее из него (устройство вывода).

Кроме того, необходимо еще несколько микросхем для объединения всех компонентов. Сам процессор мы можем представить в качестве «черного ящика», внутреннюю работу которого нам не обязательно досконально изучать, чтобы разобраться в его функциях.[3]

Центральный процессор выполняет каждую команду за несколько шагов. Для начала он вызывает следующую команду из памяти и переносит ее в регистр команд. Вторым шагом процессор меняет положение счетчика команд, которые после этого указывает на следующую команду. Затем процессор определяет тип вызванной команды и, если команда использует слово из памяти, определяет, где находится это слово. После процессор переносит слово в регистр центрального процессора, если это необходимо, и только после всех этих шагов процессор выполняет команду. После окончания выполнения команды, он переходит к первому шагу, чтобы начать выполнение следующей команды. Такая последовательность шагов является основой работы всех компьютеров.

### **1.2.2** Основная (оперативная) память

Память – это тот компонент компьютера, в котором хранятся программы и данные. Также часто встречается термин «запоминающее устройство». Без памяти, откуда процессоры считывают и куда записывают информацию, не было бы современных цифровых компьютеров.

В общем случае основной памятью компьютера называют оперативную память (Random Access Memory, RAM). Оперативная память играет ключевую роль в функционировании компьютера и используется для временного хранения данных и кода, которые активно используются программами во время их выполнения.

Основной единицей измерения хранения данных в памяти является двоичный разряд, который называется битом. Бит может содержать 0 или 1. Эта самая маленькая единица памяти.

Применение двоичной системы счисления в компьютерах объясняется ее «эффективностью». При этом имеется в виду, что хранение цифровой информации может быть основано на отличиях между разными величинами какой-либо физической характеристики, например напряжения или тока. Чем больше величин нужно различать, тем меньше отличий между смежными величинами и тем менее надежна память. В двоичной системе счисления требуется различать всего две величины, следовательно, это самый надежный метод кодирования цифровой информации.

Память состоит из ячеек, каждая из которых может хранить некоторую порцию информации. Ячейка – минимальная адресуемая единица памяти. В последние годы практически все производители выпускают компьютеры с 8-разрядными ячейками, которые называются байтами. Байты группируются в слова. Каждая ячейка имеет номер, который называется адресом. По адресу программы могут ссылаться на определенную ячейку.

В компьютерах, в которых используется двоичная система счисления (включая восьмеричной и шестнадцатеричное представление двоичных чисел), адреса памяти также выражаются в двоичных числах.

### **1.2.3** Вспомогательная память

Каков бы ни был объем основной памяти, ее все равно будет мало. С развитием технологий людям приходят в голову такие вещи, которые раньше считались совершенно фантастическими. Огромный объем информации в настоящее время невозможно разместить в основной памяти, поэтому на данный момент все пользуются вспомогательной памятью.

Устройства на базе энергонезависимой флэш-памяти, часто называют твердотельными накопителями или SSD-дисками (Solid State Disk), которые являются высокоскоростной альтернативой традиционным технологиям магнитных дисков.

Так как SSD-диски по сути являются памятью, они обладают более высокой производительностью по сравнению с вращающимися магнитными дисками при нулевом времени поиска. Поскольку устройство не имеет подвижных частей, оно особенно хорошо подходит для ноутбуков (колебания в перемещении не влияют на его способность обращаться к данным).

**1.2.4** Видеокарта и графический процессор

Видеокарта основана на конкретной архитектуре, которая определяет организацию и характеристики вычислительных блоков. Видеокарта содержит основана на графическом процессоре TU117. В связи с этим далее подробно рассмотрено устройство графического процессора.

Конкретно к характеристикам видеокарты можно отнести объем видеопамяти (VRAM), используемый для хранения текстур, кадров и других графических данных.

Графические процессоры (GPU) представляют собой сложные вычислительные устройства, которые спроектированы для выполнения множества параллельных вычислений. Они играют важную роль в современных вычислениях, таких как компьютерная графика и научные исследования. Важно понимать, что графические процессоры имеют собственную архитектуру и инструкции, которые несколько отличаются от центральных процессоров.

Вот более подробное описание ключевых элементов графического процессора:

1 Блоки FP32 и FP64: это вычислительные блоки, предназначенные для операций с плавающей точкой различной точности. FP32 (одинарная точность) обычно используется для большинства вычислений, в то время как FP64 (двойная точность) используется для более точных вычислений, но с более низкой производительностью.

2 Блоки INT32: эти блоки предназначены для выполнения целочисленных операций. Они могут использоваться для разнообразных вычислений, таких как обработка изображений и кодирование данных.

3 Регистры: регистры являются небольшими хранилищами данных, которые используются для временного хранения результатов вычислений. Каждое вычислительное ядро имеет свои собственные регистры.

4 Встроенная память (кэши): графические процессоры обычно имеют несколько уровней кэша для ускорения доступа к данным. Эти уровни кэша могут быть использованы для хранения часто используемых данных, уменьшая время доступа.

5 Планировщик команд и диспетчер команд: эти компоненты отвечают за управление выполнением инструкций на графическом процессоре. Планировщик команд определяет, какие задачи будут выполнены, их порядок и распределение по вычислительным блокам.

6 L0$, L1$, и L2$: эти кэши представляют собой различные уровни кэширования памяти на графическому процессоре. L0$ и L1$ используются

для временного хранения данных внутри вычислительных блоков, в то время как L2$ представляет собой кэш на уровне всей графической карты.

7 Внешняя память (память внешнего устройства): это память, доступная для графического процессора, и она обычно используется для хранения данных и текстур.

8 TMU (Texture Mapping Units) и ROP (Raster Operations Pipeline): эти блоки отвечают за текстурирование и растеризацию графики. Они играют важную роль в обработке и отображении изображений.

Графический процессор можно рассматривать как устройство асинхронного ожидания, где задачи выполняются параллельно, и управление заданиями и данными играет ключевую роль. Ядра CUDA в GPU Nvidia и ядра в GPU AMD представляют собой простые вычислительные блоки, способные выполнить параллельные вычисления с плавающей точкой. Важно понимать, что они не имеют некоторых возможностей, характерных для CPU, но обеспечивают высокую производительность при выполнении определенных действий.[7]

## 1.3 История, версии и достоинства

### **1.3.1** Архитектура x64

Архитектура x86-64, также известная как Intel 64, представляет собой расширение архитектуры x86, которая была разработана компаниями AMD и Intel. Ее история связана с потребностью в поддержке больших объемов памяти и более высокой производительности. Она была представлена в 2000 году.

С тех пор архитектура x86-64 пережила несколько версий и улучшений. Наиболее значимыми версиями были:

1 AMD 64: исходная версия архитектуры, представленная AMD в 2000 году.

2 Intel 64: Intel также внедрила свою версию архитектуры x86-64, известную как Intel 64, также в начале 2000-х годов.

Архитектура x86-64 имеет ряд значительных преимуществ:

1 Поддержка 64-битных приложений: позволяет запускать и обрабатывать 64-битные приложения, что повышает производительность и позволяет эффективно управлять большими объемами данных.

2 Поддержка больших объемов памяти: способность адресации более 4 ГБ оперативной памяти, что является критически важным для современных вычислительных задач.

3 Совместимость: обратная совместимость с 32-битными приложениями и операционными системами, что обеспечивает плавный переход к 64-разрядной архитектуре.

Архитектура x86-64 внесла множество инноваций и технологических решений, включая:

1 Наборы инструкций: расширенные наборы инструкций, которые обеспечивают более высокую производительность и возможности оптимизации.

2 Виртуализация: встроенная поддержка технологий виртуализации для лучшей изоляции и управления виртуальными машинами.

3 Защита данных: механизмы защиты данных и аппаратная защита от вредоносных атак.

Архитектура x86-64, в том числе Intel 64, широко применяется в различных областях:

1 Серверы: в крупных серверных фермах для обработки данных и виртуализации.

2 Настольные и ноутбуки: в персональных компьютерах для выполнения разнообразных задач, включая игры, мультимедиа и офисные приложения.

3 Разработка и научные исследования: для высокопроизводительных вычислений и моделирования.

Архитектура x86-64 соперничает с другими архитектурами, такими как ARM64, в различных областях, и каждая из них имеет свои сильные стороны и применение.

**1.3.2** Оперативная память (RAM) 8 ГБ

История развития оперативной памяти в компьютерах имеет долгий путь, начиная с ранних электромеханических устройств и электронных трубок в середине 20-го века. С появлением первых компьютеров, оперативная память представляла собой регистры и буферы, которые использовались для хранения ограниченного количества данных. Со временем разработчики стали осознавать необходимость увеличения объема и скорости оперативной памяти.

Первые интегральные схемы для оперативной памяти появились в 1970 годах, что позволило значительно увеличить ее объем и скорость. Затем пришли стандарты памяти, такие как DRAM (динамическая оперативная память) и SRAM (статическая оперативная память), которые сделали оперативную память более доступной и эффективной.

С появлением компьютеров с графическими пользовательскими интерфейсами в 1980-х и 1990-х годах, объем оперативной памяти стал критически важным для обеспечения плавной работы графических приложений и многозадачности.

В последние десятилетия, с развитием многоядерных процессоров и 64-разрядных операционных систем, объем и скорость оперативной памяти продолжают расти. Современные стандарты памяти DDR3, DDR4 и DDR5 предлагают высокую производительность и эффективность.

На протяжении истории компьютерных технологий существует множество версий оперативной памяти. Однако основные различия заключаются в следующем:

1 DDR (Double Data Rate) – первые стандарты, которые существуют с 2000 года.

2 DDR2 – следующее поколение памяти, улучшившее скорость передачи данных.

3 DDR3 – стандарт, который сделал память еще быстрее и эффективнее.

4 DDR4 – последнее поколение на момент написания, предлагающее высокую производительность и низкое энергопотребление.

5 DDR5 – ожидаемый стандарт, который будет продолжать увеличивать производительность.

Оперативная память играет ключевую роль в производительности компьютерных систем. Ее основные преимущества включают в себя:

– быстрый доступ к данным, что ускоряет выполнение задач;

– поддержка многозадачности;

– улучшение производительности в графических и видео приложениях;

– снижение времени загрузки операционной системы и приложений;

– надежность и стабильность в работе системы.

Инновации в оперативной памяти включают в себя управление энергопотреблением для экономии электроэнергии, технологии исправления ошибок для обеспечения надежности данных и увеличение скорости передачи данных с каждым новым стандартом.

Оперативная память сравнивается с альтернативными формами хранения данных, такими как жесткие диски и SSD (твердотельные накопители). В сравнении с ними, оперативная память предлагает мгновенный доступ к данным, что делает ее идеальным выбором для выполнения задач, требующих высокой скорости чтения и записи.

Оперативная память широко используется во всех областях информационных технологий, включая домашние компьютеры, серверы,

мобильные устройства и встраиваемые системы. Ее выдающиеся задачи включают в себя поддержку операционных систем, обработку данных в реальном времени и обеспечение высокой производительности в играх и профессиональных приложениях.

### **1.3.3** SSD-диск Micron 2210

Микрон (Micron Technology) – это американская компания, специализирующаяся на производстве полупроводников и твердотельных накопителей. Что касается SSD дисков Micron, их история связана с развитием технологии NAND-флэш памяти и ростом рынка твердотельных накопителей. Микрон начала производство SSD дисков в начале 2010-х годов. С течением времени SSD диски Micron стали обладать большей емкостью, более высокой скоростью чтения и записи, а также улучшенной надежностью благодаря развитию NAND-технологий и контроллеров. Рост объемов данных, требования к скорости и надежности, а также снижение стоимости NAND флэш памяти оказали влияние на эволюцию SSD дисков Micron.

Серия SSD дисков Micron 2210, как правило, имеет несколько версий с различными объемами памяти и характеристиками скорости. Улучшения могут включать в себя увеличение емкости, повышение скорости чтения/записи и снижение энергопотребления.

К достоинствам SSD-дисков Micron можно отнести:

– высокая скорость чтения и записи данных;

– надежность и долгий срок службы;

– отсутствие подвижных деталей;

– относительно низкое энергопотребление;

– быстрый доступ к данным.

Micron внедряет новые технологии NAND-флэш памяти, такие как 3D NAND, и совершенствует контроллеры, чтобы улучшить производительность и надежность своих SSD дисков.

SSD диски Micron 2210 применяются в различных областях, включая настольные компьютеры, ноутбуки, серверы и встраиваемые системы. Они эффективно работают в задачах, где требуется быстрый доступ к данным, надежность и производительность, таких как загрузка операционной системы, работа с приложениями и обработка данных.

### **1.3.4** Видеокарта NVIDIA GeForce GTX 1650

Архитектура графических процессоров развивалась на протяжении многих лет. Первые графические процессоры были разработаны в конце 20-го

века и предназначались для ускорения графических вычислений на компьютерах. С развитием видеоигр и требований к графике, графические процессоры стали более мощными и функциональными.

Создание дискретных видеокарт, таких как NVIDIA GeForce GTX 1650, связано с ростом популярности компьютерных игр и профессиональных графических приложений. Даты выпуска различных версий видеокарт и ключевые изменения в их архитектуре также являются важными моментами в истории.

В архитектуре GPU NVIDIA GeForce GTX есть множество версий и поколений. Каждое новое поколение обычно вносит улучшения в производительность, эффективность и функциональность. Примерами могут служить различные модели в серии GeForce GTX, такие как 900-я, 1000-я, и 1600-я серии. Важно рассмотреть, какие конкретные изменения и улучшения были внесены в каждую версию и почему они были важны для пользователей.

Архитектура NVIDIA GeForce GTX 1650 обладает рядом преимуществ, включая высокую графическую производительность, поддержку современных графических технологий, таких как Ray Tracing, эффективное охлаждение, и возможность использования в игровых и профессиональных приложениях. Эти достоинства могут существенно повысить производительность и качество визуальных задач на компьютере.

В архитектуре графического процессора всегда присутствуют инновации и технологии. Это могут быть новые алгоритмы обработки графики, улучшенные методы рендеринга, аппаратная поддержка искусственного интеллекта, и многое другое.

### 1.4 Обоснование выбора вычислительной системы

При выборе вычислительной системы важным фактором было обеспечение комфортных условий для работы, создания браузера. В данном случае выбранная вычислительная система отвечает этим требованиям по нескольким причинам:

1 Производительность процессора Intel Core i5: процессор Intel Core i5 предоставляет хорошую производительность для выполнения широкого спектра задач. Его многозадачные способности позволяют комфортно работать с различными приложениями, включая разработку программного обеспечения, обработку данных и другие вычислительно интенсивные задачи.

2 Объем и скорость оперативной памяти: выбор оперативной памяти является важным аспектом архитектуры системы. 8 ГБ оперативной памяти,

предоставляемой этой системой, обеспечивают достаточное количество ресурсов для эффективной работы с приложениями, включая среды разработки и виртуальные машины.

3 Скорость и емкость твердотельного: присутствие твердотельного накопителя обеспечивает высокую скорость загрузки операционной системы и приложений, что улучшает общую производительность системы. Он также обеспечивает хорошую отзывчивость и быстрое открытие файлов.

4 Дискретная видеокарта NVIDIA GeForce GTX 1650: для выполнения графических задач и игр, а также для разработки и тестирования графических приложений, важна дискретная видеокарта. NVIDIA GeForce GTX 1650 обеспечивает высокую производительность и качественное отображение графики.

5 Совместимость с программными средами: Windows 10 является популярной операционной системой, обеспечивающей совместимость с большим количеством программ и сред разработки. Это важно для выполнения курсового проекта и обеспечения легкости в разработке и отладке программ. Поэтому при разработке собственного браузера была выбрана именно операционная система Windows 10.

6 Портативность и мобильность: ноутбук Asus из серии TUF Gaming предоставляет портативность, что позволяет работать над проектом в различных местах и условиях, что может быть важно для эффективного выполнения задач.

Выбранная вычислительная система на основе ноутбука Asus TUF Gaming с процессором Intel Core i5, оптимизированной оперативной памятью, быстрым твердотельным накопителем и видеокартой NVIDIA GeForce GTX 1650 GDDR6 обеспечивает комфортные условия для разработки, создания браузера и взаимодействия с аппаратным обеспечением, гармонично сочетая производительность, портативность и совместимость с программным обеспечением.

## 1.5 Анализ выбранной вычислительной системы

### **1.5.1** Анализ Intel Core i5-10300H

Процессор Intel Core i5-10300H оборудован 4 ядрами и 8 потоками для управления многозадачностью. Максимальная тактовая частота процессора достигает 4.50 GHz, что гарантирует быстродействие системы. Процессор имеет максимальную температуру работы 100°C, что говорит о том, что работа процессора сбалансирована между производительностью

и теплорассеиванием. Размеры кэша процессора включают L1 (256 КБ), далее L2 (1 МБ) и L3 (8 МБ). Intel Core i5-10300H поддерживает память типа DDR4 2933, обеспечивая эффективное управление памятью. Максимально поддерживаемый размер памяти процессора достигает 128 ГБ, что обеспечивает необходимый объем ресурсов для работы с устройством. В процессор также интегрирована графика Intel UHD Graphics.[8]

**1.5.2** Анализ NVIDIA GeForce GTX 1650 и TU117

Графический процессор обладает архитектурой, оптимизированной для параллельной обработки данных. Ядра CUDA предоставляют возможность использовать вычислительные ресурсы графического процессора. NVIDIA GeForce GTX 1650 имеет 896 потоковых процессоров. Большое количество потоковых процессоров ведет к более высокой производительности
в параллельных задача, таких как обработки графики или симуляции физики. Тактовая частота видеокарты может быть около 1485-1560 МГц. Объем видеопамяти (VRAM) у видеокарты NVIDIA GeForce GTX 1650 составляет 4 ГБ. Этот объем видеопамяти обеспечивает достаточные ресурсы для обработки текстур, кадров и других графических данных.[9]

**1.5.3** Анализ оперативной памяти

В рамках вычислительной системы, выбранной для рассмотрения, отмечается использование оперативной памяти типа DDR4 с общим объемом 8 гигабайт. Указанный тип оперативной памяти характеризуется высокой скоростью передачи данных и обеспечивает эффективность функционирования системы. Объем оперативной памяти, равный 8 гигабайтам, обеспечивает достаточные ресурсы для эффективного выполнения обработки данных, выполнения программных задач и обеспечивает возможность эффективного осуществления многозадачных операций.

Оперативная память DDR4, используемая в данной системе, представляет собой четвертое поколение двойного канала синхронной динамической памяти. Этот тип памяти отличается от предыдущих версий более высокой пропускной способностью и более низким энергопотреблением. Такие характеристики способствуют повышению общей производительности системы за счет улучшенной передачи и обработки данных.

Восемь гигабайт оперативной памяти являются достаточными для обеспечения необходимого запаса ресурсов, что особенно важно в контексте эффективного выполнения вычислительных задач. Этот объем оперативной памяти позволяет системе более эффективно управлять обработкой данных, запущенными программами и одновременными многозадачными операциями.

Важно подчеркнуть, что использование DDR4 в данной системе является современным стандартом, что обеспечивает не только высокую пропускную способность, но и поддержку современных технологий и стандартов в области вычислительных систем.

Таким образом, данная конфигурация оперативной памяти способствует оптимизации работы системы и повышению ее общей производительности, что актуально в условиях современных вычислительных требований.

**1.5.4** Анализ твердотельного накопителя

SSD Micron 2210 представляет собой твердотельный накопитель с объемом хранения данных в размере 512 ГБ. Этот накопитель использует флеш-память с технологией 3D QLC NAND, что означает использование многозвенной клеточной структуры для хранения четырех уровней данных (quad-level cell). Технология 3D QLC NAND обеспечивает высокую плотность данных, что важно для максимального использования доступного пространства на накопителе.

Скорость последовательной записи на SSD Micron 2210 достигает 1070 мегабайт в секунду (МБ/с). Эта характеристика означает способность накопителя последовательно записывать данные на свою память со значительной скоростью.

Максимальная производительность SSD Micron 2210 достигает 320.000 операций ввода/вывода в секунду (IOPS). Операции ввода/вывода представляют собой ключевой параметр для оценки способности накопителя обрабатывать запросы на чтение и запись данных. Высокая производительность IOPS обеспечивает отзывчивость системы при обработке множества мелких операций чтения/записи.[10]

В целом, выбранная вычислительная система обеспечивает баланс между производительностью, совместимостью, портативностью и доступностью, что делает ее максимально оптимальным решением для выполнения данного курсового проекта.

# 2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 2.1 Структура и архитектура платформы

### **2.1.1** Выбранная платформа

Разработка курсового проекта «Клиенты сетевых служб: HTTP» была проведена на языке Python с использованием некоторых его специальных возможностей и среды разработки PyCharm.

Основной платформой разработки браузер является Python, так как именно этот язык программирования был выбран для проектирования проекта. Данный язык программирования был выбран из-за наличия необходимых и удобных инструментов, которые возможно использовать при разработке десктопных приложений.

PyCharm – это специализированная интегрированная среда разработки Python, предоставляющая широкий спектр необходимых инструментов для разработчиков, тесно интегрированных для создания удобной среды для продуктивной разработки. Все, что происходит в PyCharm, происходит в контексте проекта. Он служит основой для помощи в кодировании, массового рефакторинга, согласованности стилей кодирования и прочего.[11]

Для работы с проектом была выбрана операционная система Windows. Windows – это операционная система, созданная корпорацией Microsoft. Операционная система (ОС) – главная программа, которая запускается при включении компьютера. Она позволяет пользователям компьютера работать с файлами, пользоваться нужными приложениями, выходить в Интернет и осуществлять поиск и загрузку необходимых ресурсов.[12]

### **2.1.2** Язык программирования

Язык программирования – формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтактических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель под ее управлением.

Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами и прочее. Язык программирования отличается от естественных языков тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего,

для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

### **2.1.3** Интегрированная среда разработки

Интегрированная среда разработки (IDE) – комплекс программных средств, используемый программистами для разработки программного обеспечения.

Среда разработки включает в себя:

– текстовый редактор;

– транслятор (компилятор или интерпретатор);

– средства сборки и отладки.

Интегрированные среды разработки были созданы для того, чтобы максимизировать производительность программиста благодаря тесно связанным компонентам с простыми пользовательскими интерфейсами.

IDE обычно представляет собой единственную программу, в которой проводится вся разработка. Она, как правило, содержит много функций для создания, изменения, компилирования, развертывания и отладки программного обеспечения. Цель интегрированной среды заключается в том, чтобы объединить различные утилиты в одном модуле, который позволит абстрагироваться от выполнения вспомогательных задач, тем самым позволяя программисту сосредоточиться на решении собственно алгоритмической задачи и избежать потерь времени при выполнении типичных технических действий (например, вызове компилятора). Таким образом, повышается производительность труда разработчика. Также считается, что тесная интеграция задач разработки может далее повысить производительность за счет возможности введения дополнительных функций на промежуточных этапах работы. Например, IDE позволяет проанализировать код и тем самым обеспечить мгновенную обратную связь и уведомить о синтаксических ошибках.

### **2.1.4** Операционная система

Как уже отмечалось, современный компьютер организован в виде иерархии уровней, каждый из которых добавляет определенные функции к нижележащему уровню.

С точки зрения программиста, операционная система – это программа, добавляющая ряд команд и функций к командам и функциям, предлагаемым уровнем архитектуры команд. Обычно операционная система организуется

программно, но нет никаких веских причин, по которым ее, как микропрограммы, нельзя было реализовать.

Все команды уровня операционной системы доступны для прикладных программистов. Это практически все команды более низкого уровня, а также новые команды, добавленные операционной системой. Новые команды называются системными вызовами. Они вызывают предопределенную службу операционной системы, в частности одну из ее команд. Например, типичный системный вызов может читать данные из файла.

## 2.2 История, версии и достоинства

### **2.2.1** Язык программирования Python

Python – это мощный и популярный язык программирования, который был создан Гвидо ван Россумом и впервые выпущен в конце 1980-х годов.

Работа над Python началась в конце 1980-х годов, и первая публичная версия, Python 0.9.0, была выпущена в 1991 году. Основными целями создания Python были читаемость кода, простота и ясность.

Версии Python 2.x стали очень популярными и использовались в различных проектах. Однако, с развитием языка возникли некоторые проблемы, связанные с совместимостью и дизайном.

Python 3.x (2008 – н.в.): Python 3.x был выпущен в 2008 году и представил множество улучшений и изменений, направленных на устранение проблем, обнаруженных в Python 2.x. Это привело к некоторым разрывам совместимости, но с течением времени многие проекты перешли на Python 3.x.

Python имеет множество версий, но две основные ветки – Python 2.x и Python 3.x. Каждая из этих веток имеет свои подверсии, и каждая версия вносит улучшения и изменения. Кроме того, Python активно развивается, и новые версии регулярно выпускаются с улучшениями и новыми возможностями.

Python пользуется широкой популярностью благодаря ряду выдающихся достоинств:

1 Простота и читаемость: Python известен своей простотой и легкочитаемым синтаксисом, который делает код более понятным и поддерживаемым.

2 Множество библиотек и фреймворков: Python имеет богатую экосистему библиотек и фреймворков для разработки, что упрощает создание разнообразных приложений.

3 Кроссплатформенность: Python поддерживает множество операционных систем, что делает его доступным на различных платформах.

4 Широкое применение: Python применяется в разных областях, включая веб-разработку, научные вычисления, искусственный интеллект, анализ данных и многое другое.

5 Активное сообщество: Python имеет активное сообщество разработчиков и пользователей, что обеспечивает поддержку и развитие языка.

Python представляет собой мощный язык программирования, ориентированный на читаемость и простоту. Его обширная экосистема, кроссплатформенность, множество библиотек, и широкое применение в различных областях делают его популярным выбором, поддерживаемым активным сообществом разработчиков.

**2.2.2** Среда разработки PyCharm

PyCharm – это одна из самых популярных сред разработки для языка программирования Python. Она была создана компанией JetBrains и впервые выпущена в 2010 году.

Работа над PyCharm началась в JetBrains как ответ на растущую популярность языка Python. Целью было создание мощной и интуитивно понятной среды разработки для Python-разработчиков.

Первая версия PyCharm была выпущена в феврале 2010 года. Она предоставила множество возможностей, таких как интегрированный отладчик, автодополнение кода, управление виртуальными окружениями и многое другое.

С течением времени PyCharm стала одной из самых популярных сред разработки Python и продолжает активно развиваться и совершенствоваться.

PyCharm имеет несколько версий, каждая из которых предоставляет разные уровни функциональности и целевые аудитории.

Версии PyCharm:

1 PyCharm Community Edition: бесплатная версия PyCharm, предназначенная для небольших проектов и индивидуальных разработчиков.

2 PyCharm Professional Edition: платная версия PyCharm, предоставляющая расширенные возможности для профессиональных разработчиков и команд.

PyCharm пользуется выдающимися достоинствами, которые делают ее одной из предпочтительных сред разработки Python:

1 Мощный редактор кода: PyCharm обладает мощным и интуитивно понятным редактором кода с функциями автодополнения, подсветки синтаксиса и другими инструментами для удобной работы с кодом.

2 Интегрированный отладчик: встроенный отладчик позволяет проще и быстрее находить и устранять ошибки в коде.

3 Управление виртуальными окружениями: PyCharm облегчает создание и управление виртуальными окружениями, что полезно при работе с разными проектами и библиотеками.

4 Поддержка множества фреймворков: PyCharm поддерживает различные фреймворки, включая Django, Flask, PyQT и другие, упрощая разработку веб-приложений и других приложений.

5 Анализ кода и рефакторинг: Среда предоставляет множество инструментов для анализа кода и проведения рефакторинга, что помогает улучшить качество кода.

PyCharm, разработанная JetBrains, стала одной из ведущих сред разработки для Python, предоставляя мощные инструменты, включая интегрированный отладчик, управление виртуальными окружениями и поддержку различных фреймворков. Ее популярность обусловлена высокой функциональностью и продолжающимся развитием.

### **2.2.3** Операционная система Windows

Операционная система Windows разработана корпорацией Microsoft и является одной из самых распространенных операционных систем в мире. Ее история насчитывает несколько десятилетий, и вот ключевые моменты ее развития:

1 Windows 1.0 (1985): первая версия Windows была выпущена в 1985 году. Она предоставляла графический интерфейс пользователя (GUI) для MSDOS и включала приложения, такие как калькулятор и блокнот.

2 Windows 3.0 (1990): Windows 3.0 внесла множество улучшений, включая поддержку цветовых экранов и возможность мультитаскинга.

3 Windows 95 (1995): Windows 95 была революционным выпуском, введя значительные изменения в интерфейс пользователя и поддерживая 32-битные приложения.

4 Windows XP (2001): Windows XP была одной из самых долгожданных версий Windows и предоставила стабильность и производительность.

5 Windows 7 (2009): Windows 7 улучшила интерфейс пользователя и внесла улучшения в производительность.

6 Windows 8 и 8.1 (2012): эти версии внесли радикальные изменения в интерфейс, с учетом сенсорных устройств.

7 Windows 10 (2015): Windows 10 была представлена как универсальная операционная система для ПК, планшетов и смартфонов.

Windows обладает рядом значительных преимуществ и достоинств:

1 Популярность и совместимость: Windows является самой популярной операционной системой, что обеспечивает широкую совместимость с программным обеспечением и аппаратным обеспечением.

2 Игровая платформа: Windows является ведущей платформой для игр и обеспечивает доступ к широкому спектру видеоигр.

3 Простота использования: интерфейс Windows обычно считается более интуитивным и легким для освоения, особенно для новых пользователей.

4 Множество функций: Windows предоставляет множество функций, включая встроенные инструменты для работы с офисными приложениями, веб-браузерами и мультимедиа

5 Обновления и поддержка: Microsoft регулярно выпускает обновления и обеспечивает техническую поддержку для Windows, обеспечивая безопасность и стабильность системы.

6 Широкий спектр аппаратных устройств: Windows поддерживает множество устройств, включая настольные ПК, ноутбуки, планшеты и смартфоны.

Windows, с богатой историей развития с 1985 года, остается одной из самых популярных операционных систем, обеспечивая широкую совместимость, удобный интерфейс, множество функций и поддержку различных устройств, подтверждая свою роль в качестве ведущей платформы в мире компьютерных технологий.

### 2.3 Обоснование выбора платформы

Для разработки браузера были рассмотрены несколько IDE:

– PyCharm;

– Visual Studio.

Была выбран именно среда разработки PyCharm как основная среда разработки для языка Python. Эта мощная IDE предоставляет обширный набор инструментов для работы с Python, включая отладчик и поддержку виртуальной среды. Также PyCharm очень прост в использовании, не вызывает

трудностей и имеет возможность оказывать помощь программисту в разработке кода, показывая специальные подсказки для упрощения работы.

Также для разработки браузера были рассмотрены несколько языков программирования, а именно:

– C++;

– Python;

– C##.

Каждый из рассмотренных языков имеет огромное количество инструментов для создания качественного десктопного приложения. Но для разработки был выбран именно язык программирования Python, так как он является совершенно простым в использовании, а библиотеки и модули, которые предоставляют Python просты для освоения.

Использованием языка программирования Python и интегрированной среды разработки PyCharm также и последующим причинам. Во-первых, Python был выбран из-за существующего опыта работы с ним и предпочтения этого языка программирования. Во-вторых, PyCharm была выбрана как среда разработки из-за ее удобства и функциональности, а также из-за предпочтения этой среды разработки.

Этот выбор основывается на комфорте использования и знании инструментов, что способствует более эффективной разработке курсового проекта.

## 2.4 Анализ операционной системы и программного обеспечения для написания программы

### **2.4.1** Анализ операционной системы Windows 10

Операционная система Windows 10 представляет собой современное программное обеспечение. Windows 10 – это современная операционная система, разработанная Microsoft. Она была выпущена в 2015 году и с тех пор стала одной из самых популярных операционных систем в мире. Windows 10 известна своим удобным пользовательским интерфейсом, высокой степенью совместимости и широкими возможностями настройки.

Пользовательский интерфейс Windows 10 разработан для обеспечения удобства использования. Он интуитивно понятен и удобен в использовании, даже для начинающих пользователей. Одним из наиболее заметных изменений в Windows 10 является меню "Пуск". Оно было переработано и теперь включает в себя как традиционные элементы, такие как список программ, так и новые элементы, такие как живые плитки. Живые плитки

предоставляют пользователям быстрый доступ к информации и обновлениям от различных приложений.

Windows 10 является одной из наиболее распространенных операционных систем в мире. Это обеспечивает высокую степень совместимости с огромным количеством программного обеспечения. Пользователи могут быть уверены, что их любимые приложения и периферийные устройства будут работать с Windows 10.

Windows 10 предлагает пользователям высокую степень гибкости настроек. Пользователи могут настроить практически все аспекты системы, включая внешний вид, поведение и функции. Это позволяет пользователям настроить Windows 10 в соответствии со своими индивидуальными потребностями и предпочтениями.

Windows 10 оптимизирована для обеспечения высокой производительности. Она использует ряд технологий для улучшения скорости и отзывчивости системы. Например, Windows 10 использует новую функцию под названием "Fast Startup", которая позволяет системе загружаться быстрее.

Windows 10 поставляется с рядом мощных инструментов управления. Эти инструменты позволяют пользователям легко и эффективно управлять своей системой. Например, пользователи могут использовать диспетчер задач для мониторинга и управления запущенными процессами и приложениями.

К преимуществам Windows 10 относятся:

– удобный пользовательский интерфейс;

– высокая степень совместимости;

– широкие возможности настройки;

– высокая производительность;

– мощные инструменты управления.

К недостаткам Windows 10 относятся:

– может быть ресурсоемкой;

– может быть сложной для начинающих пользователей;

– некоторым пользователям может не нравиться новый пользовательский интерфейс.

Windows 10 – это современная и мощная операционная система. Она предлагает пользователям удобный пользовательский интерфейс, высокую степень совместимости, широкие возможности настройки и высокую производительность. Хотя Windows 10 может быть ресурсоемкой и сложной для начинающих пользователей, ее преимущества перевешивают недостатки.

**2.4.2** Анализ языка программирования Python

Язык программирования Python является языком высокого уровня, который славится своей читаемостью и простотой синтаксиса. Python обладает обширной стандартной библиотекой и возможностью использовать огромное количество библиотек из удаленного доступа. Данный язык программирования поддерживает процедурное, объектно-ориентированное и функциональное программирование, что обеспечивает возможность выбора наилучшего подхода для различных задач. Так же Python обладает возможностью интеграции с языками низкого уровня, написанными на C или C++.

Python – это язык программирования высокого уровня, который славится своей читаемостью и простотой синтаксиса. Он был создан Гвидо ван Россумом в конце 1980-х годов и с тех пор стал одним из самых популярных языков программирования в мире.

К особенностям языка программирования Python относятся:

1 Читаемость и простота синтаксиса: Python известен своим простым и понятным синтаксисом, который делает код легко читаемым и понятным. Это снижает вероятность ошибок и облегчает обслуживание кода.

2 Обширная стандартная библиотека: Python поставляется с обширной стандартной библиотекой, которая включает в себя модули для различных задач, таких как обработка строк, работа с файлами, сетевое программирование и разработка веб-приложений. Это избавляет от необходимости писать код для этих общих задач с нуля.

3 Поддержка различных парадигм программирования: Python поддерживает процедурное, объектно-ориентированное и функциональное программирование. Это позволяет разработчикам выбирать наилучший подход для различных задач, что приводит к более гибкому и эффективному коду.

4 Интеграция с языками низкого уровня: Python может интегрироваться с языками низкого уровня, такими как C и C++. Это позволяет разработчикам использовать функции и библиотеки, написанные на этих языках, в своих программах на Python.

К преимуществам языка программирования Python относятся:

1 Удобство: Python прост в изучении и использовании, что делает его отличным выбором для начинающих программистов.

2 Повышение производительности: обширная стандартная библиотека и поддержка различных парадигм программирования позволяют разработчикам писать код быстрее и эффективнее.

3 Широкий спектр применения: Python может использоваться для широкого спектра задач, включая разработку веб-приложений, анализ данных, машинное обучение и автоматизацию.

4 Активное сообщество: у Python есть большое и активное сообщество, которое предоставляет поддержку, документацию и ресурсы разработчикам.

К недостаткам языка программирования Python относятся:

1 Скорость: Python может быть медленнее, чем некоторые другие языки программирования, такие как C или C++.

2 Интерпретируемый язык: Python является интерпретируемым языком, что означает, что он выполняется построчно интерпретатором. Это может привести к более медленному выполнению кода по сравнению с компилируемыми языками.

3 Отсутствие строгой типизации: Python является динамически типизированным языком, что означает, что типы данных переменных не проверяются во время компиляции. Это может привести к ошибкам во время выполнения, которые трудно отследить.

Python – это мощный и универсальный язык программирования, который подходит для широкого спектра задач. Его простота, гибкость и обширная поддержка сообщества делают его отличным выбором как для начинающих, так и для опытных разработчиков.

### **2.4.3** Анализ интегрированная среды разработки PyCharm

PyCharm – это мощный инструмент для создания и поддержки проектов на языке Python. Данная интегрированная среда разработки представляет интуитивно понятный интерфейс, чтоб облегчает работу при работе с ней. PyCharm обеспечивает статический анализ кода, который помогает выявить проблемы быстро, что повышает качество программного кода. Так же данная среда разработки представляет обширную документация, что упрощает процесс работы с ней и освоение новых возможностей и технологий.

Рассмотренные компоненты информационной системы, включая операционную систему Windows 10, язык программирования Python и интегрированную среду разработки PyCharm, являются современными и эффективными инструментами для разработки и выполнения программных проектов.

Интегрированная среда разработки PyCharm дополняет опыт разработки на Python, предоставляя интуитивно понятный интерфейс, статический анализ кода и обширную документацию. Эти функциональности улучшают процесс

разработки, помогают выявлять ошибки на ранних этапах и обеспечивают удобство работы с проектами.

К особенностям PyCharm относятся:

1 Интуитивно понятный интерфейс: PyCharm имеет современный и интуитивно понятный интерфейс, который облегчает работу разработчикам. Он предоставляет ряд удобных функций, таких как автозаполнение кода, подсветка синтаксиса и навигация по коду.

2 Статический анализ кода: PyCharm выполняет статический анализ кода, который помогает выявлять проблемы на ранних этапах разработки. Он проверяет код на наличие ошибок, предупреждает о потенциальных проблемах и предлагает исправления. Это помогает повысить качество программного кода и снизить вероятность возникновения ошибок во время выполнения.

3 Обширная документация: PyCharm поставляется с обширной документацией, которая охватывает все аспекты IDE. Эта документация помогает разработчикам быстро освоить новые возможности и узнать, как эффективно использовать PyCharm для своих проектов.

К особенностям PyCharm относятся:

1 Улучшение производительности: PyCharm помогает разработчикам писать код быстрее и эффективнее. Его функции автозаполнения кода, рефакторинга и отладки экономят время и усилия разработчиков.

2 Повышение качества кода: статический анализ кода PyCharm помогает выявлять ошибки и проблемы на ранних этапах разработки. Это приводит к более надежному и безошибочному коду.

3 Упрощение обучения и работы: обширная документация и удобный интерфейс PyCharm облегчают разработчикам изучение и использование IDE.

В целом, использование Windows 10, языка программирования Python и интегрированной среды разработки PyCharm обеспечивает современные и эффективные условия для создания, поддержки и оптимизации программных проектов. Комбинация этих компонентов обеспечивает удобство использования, высокую производительность и возможность выбора наилучших подходов к различным задачам разработки.

# 3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

## 3.1 Протокол HTTP

HyperText Transfer Protocol (HTTP) – сетевой протокол уровня приложений, который изначально предназначался для получения с серверов гипертекстовых документов в формате HTML, а с течением времени стал универсальным средством взаимодействия между узлами как Всемирной паутины, так и изолированных веб-инфраструктур.

### **3.1.1** Основные свойства протокола HTTP

Основой HTTP является технология «клиент-сервер», то есть предполагается существование следующих элементов:

– потребителей или же клиентов, которые инициируют соединение и посылают запрос;

– поставщиков или же серверов, которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

HTTP в настоящее время повсеместно используется во Всемирной паутине для получения информации с веб-сайтов.

HTTP используется также в качестве «транспорта» для других протоколов прикладного уровня, таких как SOAP, XML-RPC, WebDAV.

Основным объектом манипуляции в HTTP является ресурс, на который указывает Uniform Resource Identifier (URL) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное.

Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и прочему. Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

HTTP – протокол прикладного уровня; аналогичными ему являются FTP и SMTP. Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI.

В отличие от многих других протоколов, HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния

между парами «запрос-ответ». Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами (например, «куки» на стороне клиента, «сессии» на стороне сервера). Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

Большинство протоколов предусматривает установление TCP-сессии, в ходе которой один раз происходит авторизация, и дальнейшие действия выполняются в контексте этой авторизации. HTTP же устанавливает отдельную TCP-сессию на каждый запрос. В более поздних версиях HTTP было разрешено делать несколько запросов в ходе одной TCP-сессии, но браузеры обычно запрашивают только страницу и включённые в неё объекты (картинки, каскадные стили и прочие элементы), а затем сразу разрывают TCP-сессию. Для поддержки авторизованного (неанонимного) доступа в HTTP используются так называемые файды cookies, причём такой способ авторизации позволяет сохранить сессию даже после перезагрузки клиента и сервера.

При доступе к данным по FTP или по файловым протоколам тип файла определяется по расширению имени файла, что не всегда удобно. HTTP перед тем, как передать сами данные, передаёт заголовок «Content-Type: тип/подтип», позволяющий клиенту однозначно определить, каким образом обрабатывать присланные данные. Это особенно важно при работе с CGI-скриптами, когда расширение имени файла указывает не на тип присылаемых клиенту данных, а на необходимость запуска данного файла на сервере и отправки клиенту результатов работы программы, записанной в этом файле. При этом один и тот же файл в зависимости от аргументов запроса и своих собственных соображений может порождать ответы разных типов – в простейшем случае картинки в разных форматах

Кроме того, HTTP позволяет клиенту прислать на сервер параметры, которые будут переданы запускаемому CGI-скрипту. Для этого же в HTML были введены формы.

Перечисленные особенности HTTP позволили создавать поисковые машины, форумы и интернет-магазины. В связи с этим стали появляться компании, основным полем деятельности которых стало предоставление доступа в Интернет (так называемые интернет-провайдеры) и создание сайтов.

### **3.1.2** Программное обеспечение

Всё программное обеспечение для работы с протоколом HTTP разделяется на три большие категории:

1 Серверы: серверы являются основными поставщиками услуг хранения и обработки запросов.

2 Клиенты: конечные потребители услуг сервера, которые осуществляют отправку запросов на сервер.

3 Прокси: посредники для выполнения транспортных служб.

Для отличия конечных серверов от прокси в официальной документации используется термин «исходный сервер». Один и тот же программный продукт может одновременно выполнять функции клиента, сервера или посредника в зависимости от поставленных задач. В спецификациях протокола HTTP подробно описывается поведение для каждой из этих ролей.

Первоначально протокол HTTP разрабатывался для доступа к гипертекстовым документа. Поэтому основными реализациями клиентов являются браузеры. Для просмотра сохранённого содержимого сайтов на компьютере без соединения с Интернетом были придуманы офлайн-браузеры. При нестабильном соединении для загрузки больших файлов используются менеджеры закачек; они позволяют в любое время докачать указанные файлы после потери соединения с веб-сервером. Некоторые виртуальные атласы, такие как Google Планета Земля и NASA World Wind, тоже используют HTTP.

Нередко протокол HTTP используется программами для скачивания обновлений.

Целый комплекс программ-роботов используется в поисковых системах. Среди них веб-пауки, которые производят проход по гиперссылкам, составляют базу данных ресурсов серверов и сохраняют их содержимое для дальнейшего анализа.

### **3.1.3** Методы HTTP

Метод HTTP – последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Обычно метод представляет собой короткое английское слово, записанное заглавными буквами.

Сервер может использовать любые методы, не существует обязательных методов для сервера или клиента. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус 501 (Not Implemented). Если серверу метод известен, но он неприменим к конкретному ресурсу, то возвращается сообщение с кодом 405 (Method Not Allowed). В обоих случаях

серверу следует включить в сообщение ответа заголовок Allow со списком поддерживаемых методов. Кроме методов GETи HEAD, часто применяется метод POST.

Метод OPTIONS используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях. Результат выполнения этого метода не кэшируется.

Предполагается, что запрос клиента может содержать тело сообщения для указания интересующих его сведений. Формат тела и порядок работы с ним в настоящий момент не определён; сервер пока должен его игнорировать. Аналогичная ситуация и с телом в ответе сервера.

Для того, чтобы узнать возможности всего сервера, клиент должен указать в URI символ «*». Запросы вида «OPTIONS * HTTP/1.1» могут также применяться для проверки работоспособности сервера и тестирования на предмет поддержки сервером протокола HTTP версии 1.1.

Метод GET используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?», например «GET /path/resource?param1=value1&param2=value2 HTTP/1.1».

Кроме обычного метода GET, различают ещё

1 Условный GET: содержит заголовки If-Modified-Since, If-Match, If-Range и подобные;

2 Частичный GET: содержит в запросе Range. Порядок выполнения подобных запросов определён стандартами отдельно.

Метод HEAD аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше – копия ресурса помечается как устаревшая.

Метод POST применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются

серверу методом POST и он помещает их на страницу. При этом передаваемые данные включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

В отличие от метода GET, метод POST не считается идемпотентным, то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты (например, после каждой отправки комментария будет появляться очередная копия этого комментария).

При результате выполнения 200 (Ok) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ 201 (Created) с указанием URI нового ресурса в заголовке Location.

Сообщение ответа сервера на выполнение метода POST не кэшируется.

Метод PUT применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существует ресурса, то сервер создаёт его и возвращает статус 201 (Created). Если же ресурс был изменён, то сервер возвращает 200 (Ok) или 204 (No Content). Сервер не должен игнорировать некорректные заголовки, передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или недопустим при текущих условиях, то необходимо вернуть код ошибки 501 (Not Implemented).

Фундаментальное различие методов POST и PUT заключается в понимании предназначений URI ресурсов. Метод POST предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя PUT, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу.

Сообщения ответов сервера на метод PUT не кэшируются.

Метод PATCH аналогичен PUT, но применяется только к фрагменту ресурса.

Метод DELETE удаляет указанный ресурс.

Метод TRACE возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе.

Метод CONNECT преобразует соединение запроса в прозрачный TCP/IP-туннель, обычно чтобы содействовать установлению защищённого SSL-соединения через нешифрованный прокси.

**3.1.4** Коды состояния

Код состояния является частью первой строки ответа сервера. Он представляет собой целое число из трёх цифр. Первая цифра указывает на класс состояния. За кодом ответа обычно следует отделённая пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа.

Клиент узнаёт по коду ответа о результатах его запроса и определяет, какие действия ему предпринимать дальше. Набор кодов состояния является стандартом, и они описаны в соответствующих документах RFC. Введение новых кодов должно производиться только после согласования с IETF. Клиент может не знать все коды состояния, но он обязан отреагировать в соответствии с классом кода.

В настоящее время выделено пять классов кодов состояния:

1 Информационный класс: коды состояния информационного класса имеют формат 1xx. Коды данного класса предоставляют информацию о процессе передачи данных. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-серверы подобные сообщения должны отправлять дальше от сервера к клиенту. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-серверы подобные сообщения должны отправлять дальше от сервера к клиенту.

2 Класс успешного принятия и обработки: коды состояния данного класса имеют формат 2xx. Коды данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса, сервер может ещё передать заголовки и тело сообщения.

3 Класс перенаправления: кода состояния данного класса имеют формат 3xx. Они сообщает клиенту, что для успешного выполнения операции необходимо сделать другой запрос. Из данного класса пять кодов, а именно 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям. Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location. При этом допускается использование фрагментов в целевом URI.

4 Класс клиентских ошибок: коды состояния данного класса имеют формат 4xx. Они указывают на ошибки со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

5 Класс ошибок сервера: коды состояния данного класса имеют формат 5xx. Они указывают на ошибки сервера, информируют о случаях неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

### **3.1.5** Заголовки

Заголовки HTTP – это строки в HTTP-сообщении, содержащие разделённую двоеточием пару параметр-значение. Формат заголовков соответствует общему формату заголовков текстовых сетевых сообщений ARPA. Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

В примере выше каждая строка представляет собой один заголовок. При этом то, что находится до двоеточия, называется именем, а что после него – значением.

Все заголовки разделяются на четыре основных группы:

1 Основные заголовки: они могут включаться в любое сообщение клиента и сервера.

2 Заголовки запроса: они используются только в запросах клиента.

3 Заголовки ответа: они только для ответов от сервера.

4 Заголовки сущности: они сопровождают каждую сущность сообщения.

Именно в таком порядке рекомендуется посылать заголовки получателю.

Все необходимые для функционирования HTTP заголовки описаны в основных RFC. Если не хватает существующих, то можно вводить свои. Традиционно к именам таких дополнительных заголовков добавляют префикс «X-» для избежания конфликта имён с возможно существующими. Например, как в заголовках X-Powered-By или X-Cache. Некоторые разработчики используют свои индивидуальные префиксы. Примерами таких заголовков могут служить Ms-Echo-Request и Ms-Echo-Reply, введённые корпорацией Microsoft для расширения WebDAV.

### **3.1.6** Тело сообщения

Тело HTTP-сообщения, если оно присутствует, используется для передачи тела объекта, связанного с запросом или ответом. Тело сообщения отличается от тела объекта только в том случае, когда применяется кодирование передачи, что указывается полем заголовка Transfer-Encoding.

Поле Transfer-Encoding должно использоваться для указания любого кодирования передачи, применённого приложением в целях гарантирования безопасной и правильной передачи сообщения. Поле Transfer-Encoding – это свойство сообщения, а не объекта, и, таким образом, может быть добавлено или удалено любым приложением в цепочке запросов или ответов.

Правила, устанавливающие допустимость тела сообщения в сообщении, отличны для запросов и ответов.

Присутствие тела сообщения в запросе отмечается добавлением к заголовкам запроса поля заголовка Content-Length или Transfer-Encoding. Тело сообщения может быть добавлено в запрос, только когда метод запроса допускает тело объекта.

Включается или не включается тело сообщения в сообщение ответа – зависит как от метода запроса, так и от кода состояния ответа. Все ответы на запрос с методом HEAD не должны включать тело сообщения, даже если присутствуют поля заголовка объекта, заставляющие поверить в присутствие объекта. Никакие ответы с кодами состояния из информационного класса, код 204 (Нет содержимого), и 304 (Не модифицирован) не должны содержать тела сообщения. Все другие ответы содержат тело сообщения, даже если оно имеет нулевую длину.

### 3.2 Обоснование разработки

В настоящее время существует множество HTTP-клиентов, предоставляющих доступ к интернету. Некоторые из них могут быть простыми, как, например, cURL, wget или HTTPie, которые предназначены в основном для выполнения HTTP-запросов из командной строки или сценариев. Другие, такие как Postman или Insomnia, предоставляют графический интерфейс и более широкий функционал, включая тестирование API и совместную работу над проектами.

Тем не менее, большинство из перечисленных клиентов ориентированы на выполнение конкретных задач, таких как тестирование API или выполнение HTTP-запросов, и не предлагают полноценного браузерного опыта. Разработка специализированного браузера позволяет сфокусироваться на создании инструмента, который идеально подходит для работы в интернете в целом, включая просмотр веб-страниц, поиск информации и управление закладками, что является гораздо более широким и сложным набором задач.

Разработка браузера позволит пользователям создать минималистичные персонализированные рабочие пространства, адаптированные под их конкретные предпочтения.

Создание необходимого количества вкладок поможет пользователям эффективно организовать пользовательскую онлайн-деятельность. Пользователи смогут легко отслеживать и управлять своими рабочими пространствами, что повысит их производительность и удобство использования интернета.

Интеграция нескольких поисковых системы в приложении позволит пользователям быстро и удобно осуществлять поиск информации.

Добавление функции управления закладками в приложении обеспечит пользователям возможность сохранять и организовывать важные веб-страницы для последующего доступа. Это поможет пользователям быстро находить нужные ресурсы и повысит удобство использования интернета.

Добавление возможности просматривать историю приложения поможет пользователю возвращаться при необходимости к уже посещенным сайтам для повторного просмотра ресурса.

Разработка браузера, включающего выбор поисковой системы, создание вкладок, создание закладок и сохранение истории, оправдана необходимостью предоставления пользователям удобного, персонализированного и эффективного инструмента для работы в интернете. Такое приложение может улучшить организацию онлайн-активности, повысить доступность и удобство работы с поиском необходимой информации.

## 3.3 Технологии программирования, используемые для решения поставленных задач

Для разработки собственного браузера в рамках курсового проекта используются инструменты, описанные во втором разделе:

1 Python: язык программирования Python выбран в качестве основного языка разработки браузера. Python предоставляет удобный синтаксис, богатую стандартную библиотеку и обширное сообщество разработчиков, что облегчает создание и поддержку проекта. Этот язык также поддерживает множество библиотек для работы с аппаратным обеспечением и мониторинга системы.

2 PyCharm: интегрированная среда разработки PyCharm используется для разработки, отладки и тестирования браузера. PyCharm предоставляет мощные инструменты для написания кода, рефакторинга, автодополнения

и управления проектом. Его функциональность позволяет значительно упростить и ускорить процесс разработки.

3 Windows 10: проект был разработан и протестирован на операционной системе Windows 10, что обеспечило совместимость браузера с широким кругом пользователей.

Кроме того, в разработке браузера был использован модуль языка программирования Python под названием PyQt5.

Выбор Python, PyCharm и операционной системы Windows 10 для разработки браузера обеспечивает удобство программирования, эффективные инструменты разработки и широкую совместимость, а использование библиотеки PyQt5 позволяет создать удобный и комфортный интерфейс браузера.

В результате разработки курсового проекта были рассмотрены возможности языка программирования Python, а также библиотека PyQt5.

Язык программирования Python характеризуется простым и понятным синтаксисом, обширной стандартной библиотекой и большим количеством сторонних модулей и библиотек.

PyQt5 – это мощная библиотека для создания графических пользовательских интерфейсов на языке программирования Python. Она представляет собой оболочку для Qt, одного из самых популярных фреймворков для разработки кроссплатформенных приложений. Вот некоторые из основных особенностей и возможностей PyQt5:

1 Кроссплатформенность: PyQt5 позволяет создавать приложения, которые могут работать на различных операционных системах, включая Windows, macOS и Linux. Это обеспечивает большую гибкость и удобство для разработчиков, так как они могут создавать приложения, которые будут работать на любой платформе без изменений в исходном коде.

2 Богатые возможности GUI: PyQt5 предоставляет обширный набор виджетов и инструментов для создания разнообразных пользовательских интерфейсов. Это включает в себя кнопки, поля ввода, таблицы, диалоговые окна, меню и многое другое. Разработчики могут легко создавать интерфейсы любой сложности с помощью готовых компонентов PyQt5.

3 Простота использования: PyQt5 имеет простой и интуитивно понятный интерфейс, что делает его доступным для новичков в области разработки GUI. Однако при этом библиотека предоставляет множество возможностей для опытных разработчиков, позволяя создавать сложные и профессиональные приложения.

4 Обширная документация: PyQt5 обладает хорошо структурированной и подробной документацией, которая содержит множество примеров и руководств по использованию библиотеки. Это делает процесс изучения и работы с PyQt5 более эффективным и продуктивным.

Использование языка программирования Python в сочетании с библиотекой PyQt5 позволяет разработчикам создавать кроссплатформенные приложения с богатыми возможностями GUI.

### 3.4 Связь архитектуры вычислительной системы с разрабатываемым программным обеспечением

Архитектурные характеристики аппаратного обеспечения, такие как процессор, оперативная память, накопитель (*SSD* или *HDD*), и видеокарта, играют важную роль в разработке браузера. Эти аппаратные характеристики оказывают существенное влияние на производительность и функциональность браузера, а также определяют его способность эффективно обрабатывать различные типы контента и выполнять сложные задачи.

Процессор выполняет вычислительные операции и обрабатывает инструкции, необходимые для работы браузера. Более мощный процессор позволяет браузеру быстрее загружать и отображать веб-страницы, обрабатывая скрипты, выполнять сложные вычисления и работать с мультимедийным контентом, таким как видео и аудио.

Оперативная память играет ключевую роль в работе браузера, поскольку в ней хранится активное содержимое веб-страниц, открытых вкладок, кэш и другие данные. Больший объем оперативной памяти позволяет браузеру эффективнее управлять множеством открытых вкладок, уменьшает задержки при загрузке и улучшает общую производительность приложения.

Накопитель хранит файлы браузера, включая кэш, историю посещений, загруженные файлы и другие данные.

Видеокарта отвечает за обработку и отображение графики, что важно для работы с веб-графикой, видео и другими мультимедийными элементами. Мощная видеокарта позволяет браузеру плавно отображать анимации, визуализации и видео на веб-страницах.

Все эти аппаратные характеристики взаимодействую с программным обеспечением браузера, определяя его производительность, скорость работы и возможности.

# 4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

## 4.1 Обоснование и описание функций программного обеспечения

Браузер – это специальная программа, которая позволяет пользователям просматривать веб-страницы, загружать их с серверов в интернете и отображать содержимое на экране компьютера.

Браузеры обычно предоставляют стандартные функции, такие как адресная строка для ввода URL-адресов, кнопки для навигации вперед и назад, возможность добавления закладок, а также инструменты для управления вкладками.

Браузеры отвечают за обработку HTML, CSS и JavaScript кода веб-страниц и их отображение на экране устройства пользователя.

Помимо основных функций, браузеры также предоставляют дополнительные возможности, такие как интеграция с поисковыми системами для быстрого поиска информации, защита пользовательских данных через функции безопасности и защиты конфиденциальности, а также поддержка расширений и плагинов для расширения функциональности браузера в соответствии с потребностями пользователя.

Кроме того, многие современные браузеры также предлагают возможности синхронизации данных между устройствами, такими как закладки, история посещений и пароли, что обеспечивает единый и удобный пользовательский опыт на разных устройствах.

В данном курсовом проекте был разработан минималистичный браузер. В функциональность браузера были включены:

– возможность создания вкладок;

– сохранение истории поиска;

– сохранение закладок;

– смена системы поиска.

Вид разработанного в рамках курсового проекта браузера представлен на рисунке 4.1.

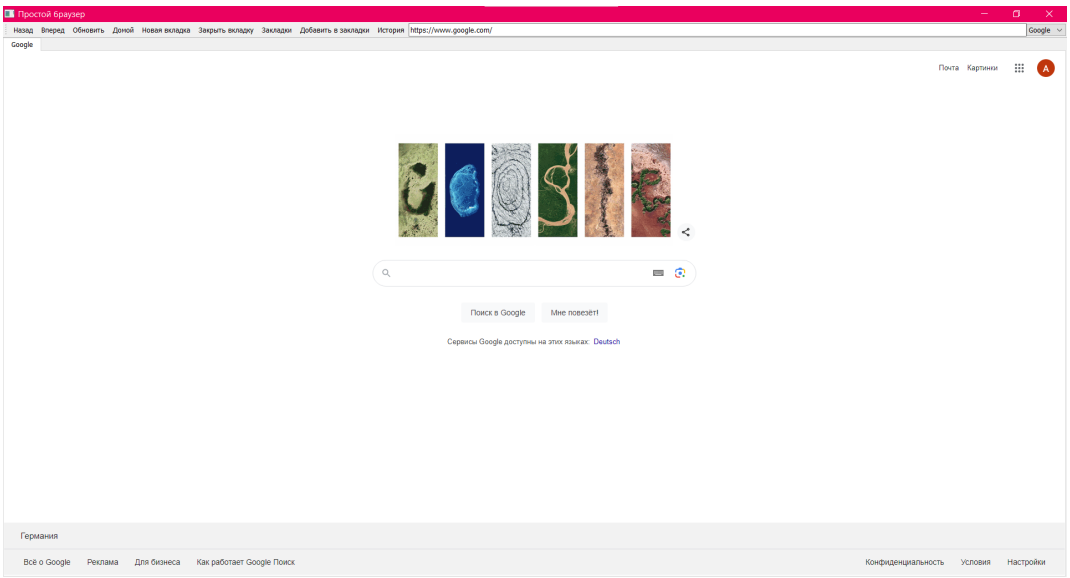


Рисунок 4.1 – Вид браузера

Браузер, разработанный в результате данного курсового проекта, реализует необходимый базовый функционал для комфортного использования. Описание функций включает в себя:

1 Создание вкладок: пользователи могут создавать новые вкладки для одновременного просмотра нескольких веб-страниц. Это позволяет удобно организовывать доступ к различным ресурсам в интернете без необходимости открытия нового окна браузера.

2 Удаление ненужных вкладок: если пользователь не нуждается в уже открытой вкладке он может удалить ее из строки вкладок для минимизации визуального шума.

3 Сохранение закладок: пользователи могут сохранять интересующие и часто посещаемые веб-страницы в виде закладок. Это позволяет быстро получать доступ к важной информации без необходимости повторного ввода адреса сайта в адресной строке или повторного поиска.

4 Удаление закладок: если пользователь не нуждается в уже созданных закладках, он может удалить их.

5 Смена системы поиска: браузер предоставляет возможность выбора поисковой системы, используемой для поиска информации в интернете. Пользователи могут выбирать между различными поисковыми системами в зависимости от своих предпочтений и потребностей.

6 Сохранение истории поиска: браузер самостоятельно сохраняет историю поиска пользователя.

7 Управление историей просмотров: пользователь может просмотреть свою историю поиска, удалить ее целиком или удалить отдельные элементы истории, а также перейти по необходимой ссылке.

Данный браузер был разработан с учетом простоты и удобства использования, а также обеспечения основного функционала, необходимого для навигации в интернете. Он позволяет пользователям управлять сессиями, организовывать доступ к важной информации и настраивать поиск в соответствии с их потребностями.

**4.1.1** Цели и задачи программного обеспечения

Цели и задачи программного обеспечения – это ключевые аспекты, определяющие его функциональность и направление разработки. К целям разработанного программного обеспечения относятся:

1 Предоставление удобного и минималистичного интерфейса: браузер разрабатывается с целью предоставить пользователям простой и интуитивно понятный интерфейс, который будет удобен в использовании даже для малоопытных пользователей.

2 Обеспечение основного функционала: основная цель браузера – это обеспечить пользователям необходимые инструменты для просмотра веб-страниц.

К задачам разработанного программного обеспечения относятся следующие аспекты:

1 Реализация функционала создания и управления вкладками: данная реализация включает в себя разработку механики создания новых вкладок, переключения между ними, а также удаление ненужных вкладок.

2 Разработка системы сохранения и управления закладками: данная реализация включает в себя создание механизма сохранения важных веб-страниц в виде закладок, а также предоставлении инструментов для их организации и доступа.

3 Разработка системы сохранения и управления историей поиска: данная реализация включает в себя создание механизма сохранения истории поиска в виде ссылок, а также предоставлении инструментов для их организации и доступа.

4 Возможность выбора поисковой системы: данная реализация включает в себя разработку функционала, который позволит пользователям выбирать предпочтительную поисковую систему для выполнения поисковых запросов.

Цели и задачи программного обеспечения определяют его направление разработки, позволяют сосредоточится на достижении ключевых целей.

**4.1.2** Ожидаемые результаты и выгоды для пользователя

Использование браузера, разработанного в ходе выполнения курсового проекта, приносит ряд конкретных выгод и результатов для пользователей:

1 Удобство использования: пользователи могут легко создавать вкладки, группировать их и сохранять закладки, что значительно упростит процесс навигации и организации работы в интернете.

2 Повышенная производительность: браузер предоставит быстрый доступ к важной информации, позволяя пользователям быстро переключаться между вкладками и мгновенно получать доступ к закладкам и поисковым результатам.

3 Персонализация: выбор поисковой системы и возможность создания групп вкладок позволят пользователям настроить браузер согласно своим потребностям и предпочтениям, создавая персонализированное пространство для работы в интернете.

4 Экономия времени: браузер позволит пользователям быстро находить нужную информацию, благодаря возможности сохранения закладок и использования выбранной поисковой системы без необходимости каждый раз вводить запросы заново.

Итак, использование разработанного браузера будет способствовать улучшению пользовательского опыта, что сделает работу в интернете более приятной и продуктивной.

# 5 АРХИТЕКТУРА РАЗРАБАТЫВАЕМОЙ ПРОГРАММЫ

## 5.1 Общая структура программы

Браузер, разработанный при работе над курсовым проектом, представлен в виде десктопного приложения, который пользователь сможет использовать по своему назначению. Интерфейс браузера предоставляет множество компонентов для работы с приложением и возможность просматривать веб страницы.

К основным компонентам браузера относятся:

1 Адресная строка: адресная строка – это строка для ввода URL-адресов. При помощи данной строки пользователь может найти необходимую страницу по ее URL-адресу, либо найти веб-страницы необходимой тематики в выбранной поисковой системе. Кроме URL-адреса пользователь может ввести

2 Кнопки навигации: к кнопкам навигации относятся кнопки «назад», «вперед». При помощи данного механизма пользователь может возвращаться назад на страницу, которую он уже посещал, либо вперед на страницу, на которой он только что был.

3 Кнопка обновления: если пользователю необходимо обновить информацию или вид текущей страницы, он может воспользоваться кнопкой «перезагрузить».

4 Выпадающий список с выбором поисковой системы: используя данный список, пользователь может выбрать необходимую для него поисковую систему.

5 Кнопка создания новой вкладки: обращаясь к данной функции пользователь может создать новую вкладку браузера.

6 Кнопка удаления вкладки: обращаясь к данной функции пользователь может удалить вкладку, которая уже ему не нужны.

7 Кнопка добавления в закладка: находясь на нужной странице, пользователь может сохранить ее в закладки с созданием специального имени для этой ссылки.

8 Кнопка просмотра закладок: обращаясь к данной функции, пользователь может просмотреть список его сохраненных закладок и перейти на нужную ему страницу. Кроме этого, пользователь может удалить ненужную ему закладку.

9 Кнопка просмотра истории: обращаясь к данной функции, пользователь может увидеть историю его посещений. Кроме просмотра

пользователь может управлять своей историей, удаляя ее элементы, а также обращаться к ним и использовать.

Структура программы следует принципам объектно-ориентированного программирования. Код программы разделен на модули и классы.

Таким образом, общая структура программы ориентирована на предоставление пользователю минималистичного браузера с базовым функционалом.

### 5.2 Описание функциональной схемы программы

Функциональная схема алгоритма, реализующего программное средство, представлена в приложении Б. Функциональная схема включает в себя следующие ключевые компоненты и функции:

– инициализация;
– обработка пользовательского ввода;
– поиск;
– навигация;
– управление вкладками;
– управление историей просмотров;
– управление закладками.

Инициализация включает в себя загрузку основных компонентов браузера, таких как веб-движок, менеджер вкладок и панель инструментов. Главное окно браузера создается и запускается с уже установленным размером и заголовком. Кроме этого, при инициализации загружаются так же и настройки пользователя, например поисковая система.

Обработка пользовательского ввода включает в себя ожидание действий пользователя, а именно нажатия кнопок навигации, ввода ссылок или необходимого набора слов в url-строку, создание новых вкладок, создание закладок и их просмотр, просмотр истории посещений. То есть данный этап включает в себя мониторинг и реагирование на действия пользователя.

Поиск включает в себя выполнение поисковых запросов с учетом выбранной поисковой системы, а также отображение результатов поиска с возможностью выбора интересующей страницы для загрузки и возможностью перехода к найденным раннее веб-страницам.

Навигация включает в себя обработку запросов на загрузку веб-страниц с использованием введенных URL-адресов или результатов поиска, переход между предыдущими и текущими страницами с помощью кнопок «назад» и «вперед», открытие новых вкладок при запросе пользователя или через ссылки

на веб-страницах, загрузку и отображение запрошенных веб-страниц в соответствующих вкладках.

Управление вкладками включает в себя создание новых вкладок по запросу пользователя и дальнейшее их использование. Пользователь кроме создания вкладки может закрыть ненужную вкладку и переключиться между открытыми вкладками. Вкладки закрываются самостоятельно после закрытия вкладок.

Управление закладками включает в себя добавление веб-страниц в список закладок по запросу пользователя с указанием названия и URL-адреса, удаление веб-страниц из списка закладок при необходимости и отображение списка закладок с возможность выбора для быстрого доступа к сохраненным страницам.

Управление историей просмотров включает в себя сохранение истории просмотров посещенных веб-страниц с записью URL-адресов и заголовков страниц, отображение истории посещений для просмотра и выбора предыдущих посещенных страниц, возможность удаления отдельных записей из истории или очистки всей истории посещений целиком.

Эти функции подробно описывают основные компоненты и функции браузера, а также действия пользователя и реакцию приложения на эти действия.

# ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта был разработан и реализован браузер, в котором были реализованы основные возможности базового браузера, а именно возможность навигации, просмотр истории поиска, создание закладок с необходимыми ресурсами и создание новых вкладок.

В процессе разработки использовались современных технологии, такие как язык программирования Python, среда разработки PyCharm и мощная библиотека PyQt5 для создания графического интерфейса пользователя.

Интерфейс пользователя включает в себя url-строку, в которой пользователь может ввести необходимый http запрос, кнопки навигации «вперед» и «назад», обновления страницы браузера, создания новой вкладки, закрытия вкладки, добавления закладки, просмотра существующих закладок, просмотра истории просмотра страницы.

Основной функционал приложения реализован с использованием объектно-ориентированного подхода, что обеспечивает удобство расширения и поддержки кода в будущем.

Таким образом реализованный браузер предоставляет пользователю удобный и интуитивно-понятный HTTP-клиент для комфортной работы и организации собственного пространства.

# СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ


[1] Архитектура вычислительных систем [Электронный ресурс]: учебное пособие – Эл. изд. – Электрон. текстовые дан. (1 файл pdf: 77 с.). – Грейбо С.В., Новоселова Т.Е., Пронькин Н.Н., Семенычева И.Ф. 2019. – Режим доступа: http://scipro.ru/conf/computerarchitecture.pdf – Дата доступа: 14.02.2024.

[2] Таненбаум, Э. Архитектура компьютера – 6-е изд. / Э. Таненбаум. СПб. : Питер, 2013. – 816 с.

[3] Петцольд, Ч. Код. Тайный язык информатики / Ч. Петцольд. – М. : Манн, Иванов и Фербер, 2021. – 448 с.

[4] Харрис, Д. М. Цифровая схемотехника и архитектура компьютера – 2-е изд. / Д. М. Харрис, С. Л. Харрис. – NY : Elsevier Inc, 2013. – 1662 с.

[5] Шитов, В. Н. Windows 10 : самый простой и понятный самоучитель / В. Шитов. – М. : Эксмо, 2023. – 464 с.

[6] Русинович, М. Внутреннее устройство Windows – 7-е изд. / М. Русинович [и др.]. – СПб. : Питер, 2018. – 944 с.

[7] Как работает GPU [Электронный ресурс]. – Режим доступа: https://coremission.net/gamedev/kak-rabotaet-gpu – Дата доступа: 15.02.2024.

[8] Обзор процессора Intel Core i5-10300H [Электронный ресурс]. – https://askgeek.io/ru/cpus/Intel/Core-i5-10300H#google_vignette – Дата доступа: 14.03.2024.

[9] Обзор видеокарты NVIDIA GeForce GTX 1650 [Электронный ресурс]. – https://3dnews.ru/987707/obzor-nvidia-geforce-gtx-1650 – Дата доступа: 14.03.2024.

[10] SSD Micron 2210 [Электронный ресурс]. – https://3dnews.ru/987707/obzor-nvidia-geforce-gtx-1650 – Дата доступа: 14.03.2024.

[11] Попов, А. Администрирование Windows с помощью WMI и WMIC. / А. В. Попов, Е. А. Шикин. СПб. : БХВ-Петербург, 2004. – 752 с.

[12] PyCharm [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/help/pycharm/quick-start-guide –Дата доступа: 15.02.2024.

[13] Windows – что это такое? [Электронный ресурс]. – Режим доступа: https://internet-lab.ru/windows_os – Дата доступа: 15.02.2024.

[14] Инструментарий управления Windows [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/wmisdk/wmi-start-page – Дата доступа: 14.03.2024.

# ПРИЛОЖЕНИЕ А
# (обязательное)
# Листинг программного кода

Листинг 1 – Программный код класса Request

```
import sys
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from PyQt5.QtWebEngineWidgets import *

class SimpleBrowser(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Простой браузер")
        self.setGeometry(0, 0, 800, 600)

        self.tabs = QTabWidget()
        self.setCentralWidget(self.tabs)

        self.add_new_tab()

        self.nav_bar = QToolBar()
        self.addToolBar(self.nav_bar)

        back_btn = QAction("Назад", self)
        back_btn.triggered.connect(self.current_browser().back)
        self.nav_bar.addAction(back_btn)

        forward_btn = QAction("Вперед", self)
        forward_btn.triggered.connect(self.current_browser().forward)
        self.nav_bar.addAction(forward_btn)

        reload_btn = QAction("Обновить", self)
        reload_btn.triggered.connect(self.current_browser().reload)
        self.nav_bar.addAction(reload_btn)

        home_btn = QAction("Домой", self)
        home_btn.triggered.connect(self.navigate_home)
        self.nav_bar.addAction(home_btn)

        new_tab_btn = QAction("Новая вкладка", self)
        new_tab_btn.triggered.connect(self.add_new_tab)
        self.nav_bar.addAction(new_tab_btn)

        close_tab_btn = QAction("Закрыть вкладку", self)
        close_tab_btn.triggered.connect(self.close_current_tab)
        self.nav_bar.addAction(close_tab_btn)

        bookmark_list_btn = QAction("Закладки", self)
        bookmark_list_btn.triggered.connect(self.show_bookmarks)
        self.nav_bar.addAction(bookmark_list_btn)
```

```
        bookmark_btn = QAction("Добавить в закладки", self)
        bookmark_btn.triggered.connect(self.add_to_bookmarks)
        self.nav_bar.addAction(bookmark_btn)

        history_btn = QAction("История", self)
        history_btn.triggered.connect(self.show_history)
        self.nav_bar.addAction(history_btn)

        self.url_bar = QLineEdit()
        self.url_bar.returnPressed.connect(self.navigate_to_url)
        self.nav_bar.addWidget(self.url_bar)

        self.search_engines = QComboBox()
        self.search_engines.addItems(["Google", "Yandex", "Mail.ru"])
        self.search_engines.activated.connect(self.navigate_home)
        self.nav_bar.addWidget(self.search_engines)


        self.history = []
        self.load_history()

        self.bookmarks = []
        self.load_bookmarks()

        self.load_search_engine()
        self.navigate_home()

    def add_to_bookmarks(self):
        current_url = self.current_browser().url().toString()
        current_title = self.current_browser().title()
        bookmark_name, ok = QInputDialog.getText(self, "Добавить в закладки",
"Введите название закладки:")
        if ok and bookmark_name:
            self.bookmarks.append((bookmark_name, current_url,
current_title))
            self.save_bookmarks()

    def show_bookmarks(self):
        bookmarks_dialog = QDialog(self)
        bookmarks_dialog.setWindowTitle("Закладки")
        layout = QVBoxLayout()

        bookmarks_list = QListWidget()
        for bookmark in self.bookmarks:
            bookmarks_list.addItem(f"{bookmark[0]} - {bookmark[1]}")
        layout.addWidget(bookmarks_list)

        delete_btn = QPushButton("Удалить закладку")
        delete_btn.clicked.connect(lambda:
self.delete_bookmark(bookmarks_list.currentRow()))
        layout.addWidget(delete_btn)

        bookmarks_dialog.setLayout(layout)
        bookmarks_dialog.exec_()
```

```
    def delete_bookmark(self, index):
        if index >= 0:
            del self.bookmarks[index]
            self.save_bookmarks()

    def load_bookmarks(self):
        try:
            with open("bookmarks.txt", "r") as file:
                for line in file.readlines():
                    parts = line.strip().split(",")
                    if len(parts) == 3:
                        self.bookmarks.append((parts[0], parts[1], parts[2]))
        except FileNotFoundError:
            pass

    def save_bookmarks(self):
        with open("bookmarks.txt", "w") as file:
            for bookmark in self.bookmarks:
                file.write(f"{bookmark[0]},{bookmark[1]},{bookmark[2]}\n")
        print("Закладки сохранены в файл")

    def load_search_engine(self):
        try:
            with open("search_engine.txt", "r") as file:
                search_engine = file.read().strip()
                index = self.search_engines.findText(search_engine)
                if index != -1:
                    self.search_engines.setCurrentIndex(index)
        except FileNotFoundError:
            pass

    def save_search_engine(self):
        search_engine = self.search_engines.currentText()
        with open("search_engine.txt", "w") as file:
            file.write(search_engine)

    def add_new_tab(self):
        browser = QWebEngineView()
        browser.setUrl(QUrl("https://www.google.com"))
        browser.urlChanged.connect(lambda q, browser=browser:
self.update_url_bar(browser, q))
        browser.titleChanged.connect(self.update_tab_title)
        self.tabs.addTab(browser, "Новая вкладка")
        self.tabs.setCurrentWidget(browser)

    def update_history(self, q):
        url = q.toString()
        title = self.current_browser().title()
        if (url, title) not in self.history:
            self.history.append((url, title))
            self.save_history()

    def current_browser(self):
```

```
        return self.tabs.currentWidget()

    def navigate_home(self):
        search_engine = self.search_engines.currentText()
        if search_engine == "Google":
            self.current_browser().setUrl(QUrl("https://www.google.com"))
        elif search_engine == "Yandex":
            self.current_browser().setUrl(QUrl("https://yandex.ru"))
        elif search_engine == "Mail.ru":
            self.current_browser().setUrl(QUrl("https://mail.ru"))

        self.save_search_engine()

    def navigate_to_url(self):
        text = self.url_bar.text()
        search_engine = self.search_engines.currentText()
        if "." not in text:
            if search_engine == "Google":
                url = QUrl("https://www.google.com/search?q=" + text)
            elif search_engine == "Yandex":
                url = QUrl("https://yandex.ru/search/?text=" + text)
            elif search_engine == "Mail.ru":
                url = QUrl("https://go.mail.ru/search?q=" + text)
        else:
            url = QUrl(text)

        self.current_browser().setUrl(url)
        self.save_history()

    def update_url_bar(self, browser, q):
        index = self.tabs.indexOf(browser)
        if index != -1:
            self.url_bar.setText(q.toString())
            url = QUrl(self.url_bar.text())
            self.update_history(url)

    def update_tab_title(self, title):
        index = self.tabs.currentIndex()
        if index != -1:
            self.tabs.setTabText(index, title)

    def close_current_tab(self):
        current_index = self.tabs.currentIndex()
        if current_index != -1:
            widget = self.tabs.widget(current_index)
            widget.deleteLater()
            self.tabs.removeTab(current_index)

    def show_history(self):
        self.history_list = QListWidget()

        self.history_list.itemClicked.connect(self.item_selected)
        self.history_list.itemDoubleClicked.connect(self.item_double_clicked)
        history_dialog = QDialog(self)
```

```
        history_dialog.setWindowTitle("История посещений")
        history_layout = QVBoxLayout()

        self.history_list.clear()
        for url, title in self.history:
            self.history_list.addItem(f"{title} - {url}")
        history_layout.addWidget(self.history_list)

        clear_btn = QPushButton("Очистить историю")
        clear_btn.clicked.connect(self.clear_history)
        history_layout.addWidget(clear_btn)

        self.delete_btn = QPushButton("Удалить")
        self.delete_btn.clicked.connect(self.delete_selected_item)
        history_layout.addWidget(self.delete_btn)

        history_dialog.setLayout(history_layout)

        self.history_list.itemClicked.disconnect(self.item_selected)

self.history_list.itemDoubleClicked.disconnect(self.item_double_clicked)

        self.history_list.itemClicked.connect(self.item_selected)
        self.history_list.itemDoubleClicked.connect(self.item_double_clicked)

        history_dialog.exec_()

    def item_selected(self, item):
        self.selected_item = item.text()

    def delete_selected_item(self):
        if hasattr(self, 'selected_item'):
            self.history.remove(self.selected_item)
            self.save_history()
            self.history_list.clear()
            for url in self.history:
                self.history_list.addItem(url)

    def item_double_clicked(self, item):
        if hasattr(self, 'selected_item'):
            url = QUrl(self.selected_item)
            self.current_browser().setUrl(url)
            self.save_history()

    def clear_history(self):
        self.history = []
        self.save_history()
        self.history_list.clear()

    def load_history(self):
        try:
            with open("history.txt", "r") as file:
                for line in file.readlines():
                    url, title = line.strip().split(",")
```

```
                    self.history.append((url, title))
        except FileNotFoundError:
            pass

    def save_history(self):
        with open("history.txt", "w") as file:
            for url, title in self.history:
                file.write(f"{url},{title}\n")
        print("История сохранена в файл")

app = QApplication(sys.argv)
browser = SimpleBrowser()
browser.show()
sys.exit(app.exec_())
```

# ПРИЛОЖЕНИЕ Б
**(обязательное)**
**Функциональная схема алгоритма, реализующего программное средство**

# ПРИЛОЖЕНИЕ В
## (обязательное)
## Графический интерфейс пользователя

# ПРИЛОЖЕНИЕ Г
**(обязательное)**
**Ведомость документов**