

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И МОЛОДЕЖНОЙ ПОЛИТИКИ  
СВЕРДЛОВСКОЙ ОБЛАСТИ

Государственное автономное профессиональное образовательное учреждение  
Свердловской области «Уральский политехнический колледж-МЦК»

Допустил к защите проекта  
Заведующий отделением

\_\_\_\_\_ В.О. Дарьева  
«\_\_\_\_\_» \_\_\_\_\_ 2024г.

РАЗРАБОТКА СИСТЕМЫ ОБРАТНОЙ СВЯЗИ И  
УЧЕТА ЗАДАЧ РАБОТНИКОВ АО УПП «ВЕКТОР»

Пояснительная записка к дипломному проекту  
ДП 09.02.03 86.09.24ПЗ

Специальность 09.02.03 Программирование в компьютерных системах

Руководитель проекта

\_\_\_\_\_ С.С. Туранов

Консультант экономической части

\_\_\_\_\_ О.А. Пересыпкина

Н. Контроль

\_\_\_\_\_ О.Н. Тобина

Студент группы П-486

\_\_\_\_\_ Н.Е. Мазунин

г. Екатеринбург, 2024

ВЕДОМОСТЬ ЧЕРТЕЖЕЙ

Лист	Наименование	Примечание
1	Диаграмма «Сущность-связь»	A1
2	Схема данных	A1

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	

6					ДП 09.02.03 86.09.24ПЗ						
Изм.	Лист	№ докум.	Подп.	Дата							
Разраб.		Н.Е.Мазунин			Разработка системы обратной связи и учета задач работников АО УПП «Вектор»				Лит.	Лист	Листов
Пров.		С.С.Туранов							У		
Н. контр.		О.Н.Тобина							ГАПОУ СО «УПК-МЦК»		
Утв.		В.О.Дарьева									

## СОДЕРЖАНИЕ

Введение.....	3
1. Общая часть .....	5
1.1 Характеристика аппаратного обеспечения.....	5
1.2 Характеристика программного обеспечения .....	10
1.3 Описание операционной системы .....	14
1.4 Краткое описание языка программирования .....	16
1.5 Краткое описание объекта автоматизации .....	26
2. Специальная часть.....	28
2.1 Постановка задачи.....	28
2.2 Функциональная диаграмма IDEF0.....	30
2.3 Диаграмма потоков данных DFD .....	32
2.4 Диаграмма «Сущность связь».....	32
2.5 Схема данных .....	33
2.6 Структура программы.....	34
2.7 Описание программы.....	34
2.8 Описание алгоритма .....	34
2.9 Инструкция пользователя.....	35
3. Экономическая часть .....	3
3.1 Экономическое обоснование эффективности программы.....	39
Охрана труда.....	45
Заключение .....	51

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

## Введение

Как и любая организация, АО УПП «Вектор» хочет организованной и слаженной работы между сотрудниками, тем самым повышая эффективность как каждого работника, так и всех отделов.

В связи с этим, разработка информационной системы обратной связи и учёта задач является актуальной задачей, направленной на оптимизацию процесса выполнения задач и улучшение взаимодействия между работниками. Цель проекта — создать веб-приложение, которое поможет работникам и администраторам АО УПП «Вектор» эффективно управлять задачами и обращениями, видеть актуальный статус выданных ими задач и обращений, формировать отчёты, а также обеспечить безопасность и контроль над доступом к информации.

Задачи проекта:

1. изучить существующие методы и технологии в области информационных систем обратной связи и учёта задач;
2. проанализировать требования АО УПП «Вектор»;
3. создать понятную архитектуру и удобный интерфейс веб-приложения;
4. протестировать и оптимизировать разработанное приложение;
5. внедрить приложение в АО УПП «Вектор» и обучить сотрудников его использованию.

Ожидаемые результаты:

- улучшение эффективности выполнения задач и обращений;
- повышение уровня безопасности и контроля над доступом к информации;
- улучшение взаимодействия между работниками и администраторами;
- сокращение времени на выполнение задач и обращений.

Веб-приложение будет разработано с учётом специфики труда работников АО УПП «Вектор» и позволит оптимизировать процесс выполнения задач, сделав его более удобным и эффективным.

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

# 1 ОБЩАЯ ЧАСТЬ

## 1.1 Характеристика аппаратного обеспечения

Аппаратное обеспечение (hardware) персонального компьютера (ПК) включает в себя физические компоненты, необходимые для его функционирования.

Разработка информационной системы обратной связи и учёта задач работников АО УПП «Вектор».

Основными элементами являются:

- процессор (CPU);
- оперативная память (RAM);
- накопительная память;
- материнская плата;
- блок питания;
- видеокарта;
- корпус.

### 1.1.1 Процессор (CPU)

Процессор в персональном компьютере (ПК) является центральным вычислительным устройством, отвечающим за выполнение основных операций и управление работой всех компонентов системы.

Он состоит из нескольких ключевых элементов:

- арифметико-логическое устройство (ALU) - выполняет математические и логические операции над данными;
- устройство управления (CU) - координирует работу всех частей процессора и ПК в целом, выбирая и декодируя команды программы;
- регистры - высокоскоростные запоминающие устройства внутри процессора для хранения данных и адресов;
- кэш-память - быстродействующая память, расположенная непосредственно на кристалле процессора, для хранения часто используемых данных и команд.

Современные процессоры для ПК выпускаются ведущими производителями, такими как Intel и AMD, по технологии 14-10 нм и содержат миллиарды транзисторов на одном кристалле. Они поддерживают многоядерную архитектуру, позволяющую одновременно выполнять несколько потоков команд для повышения производительности.

## Основные характеристики процессоров для ПК:

- тактовая частота - определяет скорость выполнения операций в гигагерцах (ГГц);
- количество ядер - влияет на способность одновременно обрабатывать несколько задач;
- объем кэш-памяти - влияет на быстроту доступа к часто используемым данным;

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

- поддержка инструкций - расширения набора команд для ускорения специфических вычислений;
- технологический процесс - определяет размер транзисторов и энергоэффективность.

Таким образом, современный процессор ПК представляет собой высокопроизводительное, энергоэффективное и многофункциональное устройство, обеспечивающее выполнение широкого спектра вычислительных задач.

### 1.1.2 Оперативная память (RAM)

Оперативная память - это важный компонент компьютерной системы, который обеспечивает быстрый доступ к данным и инструкциям, необходимым для работы программ. Она используется для временного хранения информации, которую компьютер активно обрабатывает в данный момент.

Оперативная память состоит из микросхем, расположенных на небольших печатных платах - модулях памяти.

Каждая микросхема содержит множество запоминающих элементов, которые могут быть двух типов:

- статические ОЗУ (SRAM) - используют триггеры на основе транзисторов, что позволяет хранить данные без потери, но они более дорогие и потребляют больше энергии;
- динамические ОЗУ (DRAM) - используют конденсаторы для хранения данных, что делает их дешевле и энергоэффективнее, но требует периодического обновления (регенерации) для предотвращения потери информации.

Современные модули оперативной памяти используют технологию DDR (Double Data Rate), которая передает данные на обоих фронтах тактового сигнала, удваивая эффективную скорость передачи. Существуют различные поколения DDR, каждое из которых имеет свои особенности и совместимость с материнскими платами.

При выборе оперативной памяти важно учитывать совместимость с материнской платой, объем, тип и частоту памяти, необходимые для конкретной системы. Увеличение объема ОЗУ позволяет запускать больше программ одновременно и улучшает производительность в ресурсоемких приложениях.

В отличие от постоянной памяти (ROM, SSD, HDD), данные в оперативной памяти теряются при отключении питания. Поэтому ОЗУ используется только для временного хранения активно используемой информации, а для долговременного хранения файлов применяются другие типы памяти.

### 1.1.3 Накопительная память

Накопительная память делится на виды:

а) жесткий диск (HDD) - это традиционный механический накопитель данных, состоящий из нескольких вращающихся магнитных пластин, называемых "блинами". Внутри жесткого диска находится специальный

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	данных, что делает их дешевле и энергоэффективнее, но требует	
					периодического обновления (регенерации) для предотвращения	
					потери информации.	
					Современные модули оперативной памяти используют технологию DDR	
					(Double Data Rate), которая передает данные на обоих фронтах тактового	
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	сигнала, удваивая эффективную скорость передачи. Существуют различные	
					поколения DDR, каждое из которых имеет свои особенности и совместимость с	
					материнскими платами.	
					При выборе оперативной памяти важно учитывать совместимость с	
					материнской платой, объем, тип и частоту памяти, необходимые для конкретной	
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	системы. Увеличение объема ОЗУ позволяет запускать больше программ	
					одновременно и улучшает производительность в ресурсоемких приложениях.	
					В отличие от постоянной памяти (ROM, SSD, HDD), данные в оперативной	
					памяти теряются при отключении питания. Поэтому ОЗУ используется только	
					для временного хранения активно используемой информации, а для	
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	долговременного хранения файлов применяются другие типы памяти.	
					<b>1.1.3 Накопительная память</b>	
					Накопительная память делится на виды:	
					а) жесткий диск (HDD) - это традиционный механический накопитель	
					данных, состоящий из нескольких вращающихся магнитных пластин,	
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	называемых "блинами". Внутри жесткого диска находится специальный	
Изм.	Лист	№ докум.	Подп.	Дата	ДП 09.02.03 86.09.24ПЗ	Лист
						6

механизм с движущимися частями, который считывает и записывает данные на эти пластины с помощью магнитных головок.

Основные преимущества жестких дисков:

- 1) большая емкость хранения данных - до нескольких терабайт на одном диске;
- 2) более низкая стоимость за гигабайт хранимой информации по сравнению с SSD;
- 3) практически неограниченное количество циклов перезаписи данных;

Недостатки жестких дисков:

- 1) более низкая скорость доступа к данным по сравнению с SSD из-за механических движущихся частей;
- 2) более высокое энергопотребление;
- 3) подверженность механическим повреждениям и вибрации;
- 4) необходимость периодической дефрагментации для поддержания производительности;

б) твердотельные накопители (SSD), не имеют движущихся частей и хранят данные в микросхемах флеш-памяти. Это позволяет им быть более быстрыми, надежными и энергоэффективными.

Основные преимущества SSD:

- 1) высокая скорость доступа к данным благодаря отсутствию механических компонентов;
- 2) более высокая устойчивость к ударам и вибрации;
- 3) меньшее энергопотребление;
- 4) отсутствие необходимости в дефрагментации.

Недостатки SSD:

- 1) более высокая стоимость за гигабайт хранимой информации;
- 2) ограниченное количество циклов перезаписи данных.

В целом, выбор между HDD и SSD зависит от конкретных потребностей пользователя - требуемой емкости, производительности, надежности и бюджета. Для большинства современных применений SSD являются предпочтительным вариантом, но жесткие диски все еще остаются востребованными для хранения больших объемов данных.

#### 1.1.4 Материнская плата

Материнская плата - это основная печатная плата в компьютере, на которой устанавливаются и соединяются все основные компоненты системы. Она служит платформой для подключения процессора, оперативной памяти, видеокарты, накопителей данных и других устройств.

Основные компоненты материнской платы:

- слоты для установки процессора (сокет);
- слоты для модулей оперативной памяти (DIMM);
- разъемы для подключения накопителей данных (SATA, M.2);
- слоты расширения для установки дополнительных плат (PCI, PCIe);

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.
Подп. и дата	
Инв. № подл.	

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

- интегрированные контроллеры (сетевой, звуковой, USB);
- разъемы для подключения корпусных компонентов (кнопки, индикаторы, вентиляторы);
- разъем для подключения блока питания.

Материнские платы классифицируются по форм-фактору, который определяет их размеры, расположение компонентов и совместимость с корпусом и блоком питания. Наиболее распространенные форм-факторы: ATX, Micro-ATX, Mini-ITX.

С развитием технологий материнские платы усложняются, добавляются дополнительные разъемы питания для процессора и видеокарты, многофазные преобразователи напряжения для стабильного питания компонентов. Современные платы поддерживают новейшие стандарты оперативной памяти, накопителей данных и интерфейсов расширения.

Правильный выбор материнской платы является ключевым фактором при сборке производительного и надежного компьютера. Она должна быть совместима с выбранным процессором, поддерживать необходимые интерфейсы и обеспечивать достаточное питание для всех компонентов системы.

### 1.1.5 Блок питания

Блок питания - это устройство, которое преобразует электрическую энергию из сети переменного тока в необходимое постоянное напряжение для питания компьютерных компонентов. Он является важной частью компьютерной системы, обеспечивая стабильное и безопасное электропитание для всех устройств.

Основные характеристики блока питания:

- мощность: измеряется в ваттах (W) и определяет способность блока питания обеспечивать энергией компоненты компьютера;
- эффективность: выражается в процентах и показывает, какая часть потребляемой энергии преобразуется в полезную энергию для компонентов, а какая теряется в виде тепла. Чем выше эффективность, тем меньше потери энергии;
- разъемы и кабели: блок питания оборудован различными разъемами для подключения к материнской плате, видеокарте, накопителям данных и другим устройствам. Важно учитывать совместимость разъемов с компонентами вашей системы;
- защита: хороший блок питания должен обладать защитой от перегрузок, короткого замыкания, перенапряжения и других неполадок, чтобы предотвратить повреждение компонентов компьютера;
- охлаждение: для поддержания нормальной работы блок питания должен быть оборудован системой охлаждения, такой как вентиляторы или система жидкостного охлаждения, чтобы предотвратить перегрев.

Инв. № подл.	Подп. и дата		Инв. № дубл.	Подп. и дата		Инв. № подл.											
<p>системы, обеспечивая стабильное и безопасное электропитание для всех устройств.</p> <p>Основные характеристики блока питания:</p> <ul style="list-style-type: none"><li>– мощность: измеряется в ваттах (W) и определяет способность блока питания обеспечивать энергией компоненты компьютера;</li><li>– эффективность: выражается в процентах и показывает, какая часть потребляемой энергии преобразуется в полезную энергию для компонентов, а какая теряется в виде тепла. Чем выше эффективность, тем меньше потерь энергии;</li><li>– разъемы и кабели: блок питания оборудован различными разъемами для подключения к материнской плате, видеокарте, накопителям данных и другим устройствам. Важно учитывать совместимость разъемов с компонентами вашей системы;</li><li>– защита: хороший блок питания должен обладать защитой от перегрузок, короткого замыкания, перенапряжения и других неполадок, чтобы предотвратить повреждение компонентов компьютера;</li><li>– охлаждение: для поддержания нормальной работы блок питания должен быть оборудован системой охлаждения, такой как вентиляторы или система жидкостного охлаждения, чтобы предотвратить перегрев.</li></ul>																	
<table><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Изм.</td><td>Лист</td><td>№ докум.</td><td>Подп.</td><td>Дата</td></tr></table>										Изм.	Лист	№ докум.	Подп.	Дата	ДП 09.02.03 86.09.24ПЗ		Лист
Изм.	Лист	№ докум.	Подп.	Дата													
							8										



Блок питания является неотъемлемой частью компьютерной системы, и правильный выбор блока питания с учетом потребностей компонентов и надежности электропитания играет важную роль в обеспечении стабильной работы компьютера.

### 1.1.6 Видеокарта

Видеокарта - это компонент компьютера, отвечающий за обработку графики и вывод изображения на монитор. Она является ключевым элементом для игр, видеообработки и других графически интенсивных задач. Видеокарты бывают различных типов, включая интегрированные, дискретные и гибридные.

Основные характеристики видеокарт:

а) типы видеокарт:

1) интегрированная видеокарта: встроена в материнскую плату и обычно используется в офисных компьютерах;

2) дискретная видеокарта: отдельное устройство, обладающее высокой производительностью и специализированными возможностями;

3) гибридная видеокарта: сочетает в себе особенности интегрированных и дискретных видеокарт;

б) производительность: производительность видеокарты зависит от нескольких факторов, включая частоту ядра, количество ядер CUDA (для NVIDIA) или потоковых процессоров (для AMD), объем видеопамати и ширину шины памяти. Частота ядра влияет на скорость обработки графики, а количество ядер CUDA или потоковых процессоров определяет количество задач, которые могут быть выполнены одновременно. Объем видеопамати влияет на количество данных, которые могут быть хранены в видеокарте, а ширина шины памяти определяет скорость передачи данных между видеокартой и системной памятью;

в) питание и охлаждение: видеокарты требуют эффективной системы охлаждения и достаточного питания для стабильной работы;

г) подключение: видеокарта обеспечивает высококачественный цифровой интерфейс передачи видеосигнала, такой как HDMI или DisplayPort, который обеспечивает передачу аудио и видео сигналов через один кабель. Это позволяет подключать мониторы и другие устройства вывода к видеокарте и получать высококачественный видеосигнал.

Выбор видеокарты зависит от потребностей пользователя, будь то игровые задачи, профессиональная графика или обычные офисные задачи. Для игровых задач и профессиональной графики требуются видеокарты с высокой производительностью, а для офисных задач может быть достаточной интегрированная видеокарта. При выборе видеокарты также необходимо учитывать совместимость с материнской платой и другими компонентами компьютера.

Видеокарты играют важную роль в обработке графики и видео на компьютере, обеспечивая плавное отображение изображений на экране и

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	ДП 09.02.03 86.09.24ПЗ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		9



д) совместимость:

1) кроссплатформенность: программы должны корректно работать на различных аппаратных платформах и операционных системах;

2) интеграция: ПО должно обеспечивать возможность взаимодействия и обмена данными с другими приложениями;

е) обновляемость:

1) поддержка: разработчики должны регулярно выпускать обновления, исправляющие ошибки и добавляющие новые функции;

2) расширяемость: программное обеспечение должно предоставлять возможности для расширения его функциональности сторонними разработчиками.[4]

Эти характеристики определяют качество, эффективность и удобство использования программного обеспечения, что в свою очередь влияет на производительность и удовлетворенность пользователей.

### 1.2.1 Веб-приложение

Веб-приложение - это программное обеспечение, которое работает на сервере и обеспечивает доступ к функциям и данным через Интернет. Оно отличается от сайта тем, что сайт является статическим и не обеспечивает интерактивность с пользователем, в то время как веб-приложение позволяет пользователю взаимодействовать с системой, вводить данные, получать результаты и управлять процессами.

Веб-приложения используются в различных областях, таких как образование, медицина, туризм и бронирование, и обеспечивают функциональность, которая может быть полезна для бизнеса и организации.[5]

В целом, веб-приложения обеспечивают функциональность, которая может быть полезна для бизнеса и организации, и требуют обеспечения информационной безопасности для защиты от атак и утечек конфиденциальной информации.

При разработке веб-приложения важно учитывать использование специализированных инструментов и технологий.

Описание характеристик программного обеспечения, связанных с разработкой веб-приложения:

- интегрированная среда разработки (IDE): для разработки веб-приложения может использоваться IDE, такая как Visual Studio Code, WebStorm, Sublime Text, которая обеспечивает удобную среду для написания кода, отладки, управления версиями и других задач разработки;
- система управления базами данных (СУБД): для хранения данных веб-приложения может использоваться СУБД, такая как MySQL, PostgreSQL, SQLite, которая обеспечивает эффективное хранение и управление данными;
- база данных (БД): для веб приложения может быть разработана база данных, которая будет содержать данные, необходимые для функционирования системы;
- языки программирования и фреймворки: для разработки веб-приложения могут использоваться языки программирования, такие

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

Лист

11

- безопасность и шифрование: информационная безопасность веб-приложений является важной задачей, так как они могут быть уязвимы для атак и утечек конфиденциальной информации. Для обеспечения безопасности веб-приложений используются различные методы, такие как поведенческий анализ, выявление нарушений политик ИБ, аналитика и автоматическое уведомление об аномалиях. Для обеспечения безопасности данных при передаче в базу данных и их хранении может использоваться алгоритм шифрования;
- тестирование и отладка: для обеспечения качества веб-приложения может использоваться автоматизированное тестирование, например, с помощью фреймворков Jest, Mocha, Selenium, а также инструменты для отладки кода, такие как Chrome DevTools.

### 1.2.2 Характеристика программного обеспечения при разработке дипломного проекта

а) JetBrains Project Rider - это кроссплатформенная интегрированная среда разработки (IDE) для платформы .NET, разрабатываемая компанией JetBrains. Она позволяет создавать приложения для Windows, веб-приложения и мобильные приложения на основе .NET, ASP.NET, .NET Core, Xamarin и Unity на Windows, Mac и Linux. Rider использует интерфейс и функциональность платформы IntelliJ, которая лежит в основе IntelliJ IDEA, WebStorm и других IDE, разработанных в JetBrains. Это обеспечивает основную функциональность Rider, включая интеграцию с системами контроля версий и поддержку баз данных. Rider работает в операционных системах Windows, MacOS и разных дистрибутивах Linux. Она поддерживает множество языков программирования, включая C#, VB.NET, F#, синтаксис Razor, ASP, JavaScript, TypeScript и другие. Rider поддерживает множество плагинов, разработанных для ReSharper и платформы IntelliJ. Помимо встроенных плагинов (например, для поддержки F#, Unity и дополнительных систем контроля версий) вы можете подключать и другие плагины. Rider обеспечивает навигацию и поиск, рефакторинги, инспекции кода, быстрые исправления и многие другие интеллектуальные функции, заимствованные из ReSharper. Это помогает читать и писать .NET-код и перемещаться по большим кодовым базам;

б) система управления базами данных (СУБД) - это программное обеспечение, которое управляет и обрабатывает данные в автоматизированных банках данных. Она является составной частью автоматизированного банка данных, который включает в себя одну или несколько баз данных, систему управления базами данных и персонал, обеспечивающий работу банка данных;

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № докл.	Подп. и дата	<p>а) JetBrains Rider - это кроссплатформенная интегрированная среда разработки (IDE) для платформы .NET, разрабатываемая компанией JetBrains. Она позволяет создавать приложения для Windows, веб-приложения и мобильные приложения на основе .NET, ASP.NET, .NET Core, Xamarin и Unity на Windows, Mac и Linux. Rider использует интерфейс и функциональность платформы IntelliJ, которая лежит в основе IntelliJ IDEA, WebStorm и других IDE, разработанных в JetBrains. Это обеспечивает основную функциональность Rider, включая интеграцию с системами контроля версий и поддержку баз данных. Rider работает в операционных системах Windows, MacOS и разных дистрибутивах Linux. Она поддерживает множество языков программирования, включая C#, VB.NET, F#, синтаксис Razor, ASP, JavaScript, TypeScript и другие. Rider поддерживает множество плагинов, разработанных для ReSharper и платформы IntelliJ. Помимо встроенных плагинов (например, для поддержки F#, Unity и дополнительных систем контроля версий) вы можете подключать и другие плагины. Rider обеспечивает навигацию и поиск, рефакторинг, инспекции кода, быстрые исправления и многие другие интеллектуальные функции, заимствованные из ReSharper. Это помогает читать и писать .NET-код и перемещаться по большим кодовым базам;</p> <p>б) система управления базами данных (СУБД) - это программное обеспечение, которое управляет и обрабатывает данные в автоматизированных банках данных. Она является составной частью автоматизированного банка данных, который включает в себя одну или несколько баз данных, систему управления базами данных и персонал, обеспечивающий работу банка данных;</p>
					<div> <div> <div>ДП 09.02.03 86.09.24ПЗ</div> <div> <div>Изм.</div> <div>Лист</div> <div>№ докум.</div> <div>Подп.</div> <div>Дата</div> </div> </div> <div> <div>Лист</div> <div>12</div> </div> </div>

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

3) Enterprise: выпуск с расширенными функциями безопасности и управления, предназначенный для крупных предприятий;

SQL Server может быть установлен на компьютерах под управлением операционных систем Windows и Linux, а также в контейнерах Docker. Существует также возможность запуска SQL Server в виртуальных машинах Azure, что обеспечивает гибкость и масштабируемость.

### 1.3 Описание операционной системы

Операционная система (ОС) - это программное обеспечение, которое является центром управления и координации ресурсов компьютера. Она обеспечивает интерфейс между аппаратным обеспечением компьютера и прикладными программами, управляет ресурсами компьютера и обеспечивает выполнение задач пользователей.

Функции Операционной Системы:

- управление ресурсами: ОС управляет доступом к ресурсам компьютера, таким как процессор, память, диски, сеть и периферийные устройства, чтобы обеспечить эффективное использование ресурсов;
- планирование задач: ОС определяет порядок выполнения задач и управляет процессами, чтобы обеспечить равномерное распределение ресурсов между приложениями;
- обеспечение безопасности: ОС контролирует доступ к данным и ресурсам компьютера, обеспечивая безопасность системы и защиту от внешних угроз;
- обеспечение интерфейса: ОС предоставляет пользовательский интерфейс, через который пользователи могут взаимодействовать с компьютером и запускать приложения;
- управление файловой системой: ОС управляет файлами и каталогами на диске, обеспечивая доступ к данным и их сохранность;
- обеспечение сетевых возможностей: Некоторые ОС имеют встроенные средства для работы в сети, обеспечивая сетевые возможности и коммуникацию между компьютерами.

Типы Операционных Систем:

- однопользовательские: предназначены для использования одним пользователем, например, на персональных компьютерах;
- многопользовательские: позволяют одновременно работать нескольким пользователям, обеспечивая разделение ресурсов и безопасность данных;
- временноразделяющие: позволяют одновременно выполнять несколько задач, переключаясь между ними с высокой скоростью;
- встраиваемые: предназначены для работы на встроенных системах, таких как мобильные устройства, медицинское оборудование и промышленные устройства.

Примеры Операционных Систем:

- Windows: разработана корпорацией Microsoft и является одной из самых популярных ОС для персональных компьютеров;
- macOS: операционная система, разработанная компанией Apple для

Инв. № подл.	Подп. и дата				ДП 09.02.03 86.09.24ПЗ	Лист 14	
	Инв. № дубл.						
	Взам. Инв. №						
	Подп. и дата						
Инв. № док.	Изм.	Лист	№ док.	Подп.	Дата		

— обеспечение интерфейса: ОС предоставляет пользовательский интерфейс, через который пользователи могут взаимодействовать с компьютером и запускать приложения;	
— управление файловой системой: ОС управляет файлами и каталогами на диске, обеспечивая доступ к данным и их сохранность;	
— обеспечение сетевых возможностей: Некоторые ОС имеют встроенные средства для работы в сети, обеспечивая сетевые возможности и коммуникацию между компьютерами.	
Типы Операционных Систем:	
— однопользовательские: предназначены для использования одним пользователем, например, на персональных компьютерах;	
— многопользовательские: позволяют одновременно работать нескольким пользователям, обеспечивая разделение ресурсов и безопасность данных;	
— временноразделяющие: позволяют одновременно выполнять несколько задач, переключаясь между ними с высокой скоростью;	
— встраиваемые: предназначены для работы на встроенных системах, таких как мобильные устройства, медицинское оборудование и промышленные устройства.	
Примеры Операционных Систем:	
— Windows: разработана корпорацией Microsoft и является одной из самых популярных ОС для персональных компьютеров;	
— macOS: операционная система, разработанная компанией Apple для	

компьютеров Mac;

- Linux: семейство операционных систем с открытым исходным кодом, которые широко используются в серверных и встроенных системах.

Операционные системы постоянно развиваются, добавляя новые функции, улучшая производительность и обеспечивая безопасность данных. С развитием технологий, ОС становятся более гибкими, масштабируемыми и безопасными, обеспечивая пользователям более удобный и эффективный опыт работы с компьютером.

Операционная система является неотъемлемой частью работы компьютера, обеспечивая управление ресурсами, безопасность данных, интерфейс для взаимодействия с пользователем и множество других функций. Ее разнообразие типов и постоянное развитие делают ОС ключевым элементом современных компьютерных систем.

Windows - это семейство операционных систем для персональных компьютеров и рабочих станций, разработанных корпорацией Microsoft в рамках семейства Windows NT. Оно является одним из самых популярных и широко используемых семейств операционных систем в мире.

Первая версия Windows была выпущена в 1985 году как графическая оболочка для MS-DOS. С тех пор семейство Windows развивалось и эволюционировало, включая в себя новые функции, интерфейсы и технологии.

Windows 10 - это версия операционной системы Windows, которая была выпущена в 2015 году. Она была разработана для персональных компьютеров и рабочих станций и является частью семейства Windows NT. Windows 10 была создана на основе Windows 8.1 и включает в себя множество новых функций, таких как интеграция с Cortana, новые функции для Xbox и Windows 10 Mobile, а также обновленный пакет Office Mobile.

Windows предлагает множество функций, включая:

- графический интерфейс: Windows использует графический интерфейс, который позволяет пользователям взаимодействовать с компьютером с помощью мыши, клавиатуры и сенсорного экрана;
- многоязычность: Windows поддерживает множество языков, что делает ее доступной для пользователей из разных стран и регионов;
- поддержка платформ: Windows может работать на различных платформах, включая ARM, IA-32 и x86-64;
- гибридное ядро: Windows использует гибридное ядро, которое обеспечивает высокую производительность и безопасность;
- интерфейс Metro и Fluent Design: Windows использует интерфейс Metro и Fluent Design, который обеспечивает современный и привлекательный внешний вид.

Windows обеспечивает регулярные обновления, которые включают в себя обновления безопасности, исправления ошибок и новые функции. Последняя версия Windows 10 будет обеспечиваться обновлениями систем безопасности до 14 октября 2025 года.

Windows - это мощное и популярное семейство, которое обеспечивает высокую производительность, безопасность и удобство использования. Ее

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

развитие и эволюция делают ее одним из лидеров на рынке операционных систем.

#### 1.4 Краткое описание языка программирования

Язык программирования - это формальный язык, используемый для создания компьютерных программ, которые выполняют определенные задачи и обрабатывают данные. Он состоит из набора правил, синтаксиса и семантики, которые определяют, как должна быть написана программа, чтобы ее могли понять и выполнить компьютеры.

Первые языки программирования, такие как Фортран и Кобол, были разработаны в 1950-х годах для использования на ранних компьютерах. С тех пор было разработано множество языков программирования, каждый из которых имеет свои особенности и предназначен для решения определенных задач.

Языки программирования можно классифицировать по нескольким критериям, таким как уровень абстракции, парадигма программирования и область применения:

- по уровню абстракции: низкоуровневые (ассемблер) и высокоуровневые (C++, Java, Python);
- по парадигме программирования: процедурные (C), объектно-ориентированные (Java, C++), функциональные (Lisp, Haskell), логические (Prolog);
- по области применения: системные (C, C++), веб-разработка (JavaScript, PHP), научные вычисления (Python, R), мобильная разработка (Swift, Kotlin).

Языки программирования могут быть либо скомпилированы, либо интерпретированы:

- компилируемые языки: программа переводится в машинный код, который может быть непосредственно выполнен компьютером (C, C++, Rust);
- интерпретируемые языки: программа выполняется строка за строкой интерпретатором (Python, Ruby, Perl).

Некоторые из наиболее популярных языков программирования в настоящее время включают:

- Python: высокоуровневый язык общего назначения, используемый для веб-разработки, анализа данных, машинного обучения и автоматизации;
- Java: объектно-ориентированный язык, используемый для разработки приложений для различных платформ, включая Android;
- JavaScript: язык программирования для веб-разработки, используемый для создания интерактивных веб-страниц и серверных приложений;
- C++: объектно-ориентированный язык, используемый для разработки системного программного обеспечения, игр и приложений;
- C#: объектно-ориентированный язык, используемый для разработки приложений для платформы .NET, включая игры и мобильные приложения.

Языки программирования продолжают развиваться, чтобы соответствовать потребностям современных технологий, таких как искусственный интеллект,

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.
Подп. и дата	

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ



машинное обучение и облачные вычисления. Ожидается, что в будущем языки программирования станут более интуитивными, безопасными и производительными, что позволит разработчикам создавать более сложные и эффективные программы.

а) клиентская часть веб-приложения – это часть веб-приложения, которая выполняется на стороне клиента (в браузере пользователя) и отвечает за визуальное представление данных, интерактивность и пользовательский опыт. Это часть приложения, с которой пользователь взаимодействует непосредственно.

HTML (Hypertext Markup Language) - это язык разметки, который лежит в основе веб-страницы, обеспечивая ее структуру, семантику и доступность. Он состоит из различных элементов, каждый из которых имеет свой уникальный смысл и функцию, и которые могут быть комбинированы для создания сложных и интерактивных веб-страниц. HTML обеспечивает семантическую разметку, что означает, что он помогает поисковым системам и скринридерам понимать структуру и содержимое страницы, что улучшает индексацию и доступность для пользователей с ограниченными возможностями. Он также обеспечивает доступ к атрибутам, которые позволяют добавить дополнительную информацию к элементам, и поддерживает вложенность элементов, что позволяет создавать сложную структуру страницы. В HTML5, последней версии языка, были добавлены новые элементы и атрибуты, которые позволяют создавать более интерактивные и динамические веб-страницы. Например, элементы `<video>` и `<audio>` позволяют добавлять аудио и видео контент на страницу, а элемент `<canvas>` позволяет создавать динамические графики и анимации. Кроме того, HTML5 ввел новые семантические элементы, такие как `<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<aside>`, `<footer>`, которые помогают лучше структурировать содержимое страницы и улучшают доступность. Эти элементы обеспечивают более четкое определение различных частей страницы, что улучшает понимание страницы поисковыми системами и скринридерами. В целом, HTML является фундаментальным языком для создания веб-страниц, обеспечивая структуру, семантику и доступность, что делает его основным инструментом для создания веб-приложений.

CSS (Cascading Style Sheets) - это мощный язык стилей, который позволяет веб-разработчикам полностью контролировать визуальное представление элементов HTML на веб-странице. Он работает по принципу каскада, что означает, что стили могут быть применены к элементам из разных источников и в различных уровнях приоритета, создавая сложные иерархии стилей. Одна из ключевых особенностей CSS - это его гибкость и масштабируемость. CSS предоставляет широкий спектр селекторов, которые позволяют выбирать элементы на странице с высокой точностью, от простых элементов до сложных комбинаций классов, идентификаторов и атрибутов. Это дает разработчикам возможность применять стили к конкретным элементам или группам элементов, обеспечивая высокую степень контроля над внешним видом страницы. Кроме того, CSS определяет модель бокса для каждого элемента, включая его размеры, отступы, границы и заполнение. Это позволяет разработчикам точно управлять размещением и компоновкой элементов на странице, создавая сложные макеты и дизайны. Одним из наиболее мощных аспектов CSS является его способность

Инв. № подл.	Подп. и дата				ДП 09.02.03 86.09.24ПЗ	Лист 17
	Инв. № дубл.					
	Взам. Инв. №					
	Подп. и дата					
<p>&lt;canvas&gt; позволяет создавать динамические графики и анимации. Кроме того, HTML5 ввел новые семантические элементы, такие как &lt;header&gt;, &lt;nav&gt;, &lt;main&gt;, &lt;section&gt;, &lt;article&gt;, &lt;aside&gt;, &lt;footer&gt;, которые помогают лучше структурировать содержимое страницы и улучшают доступность. Эти элементы обеспечивают более четкое определение различных частей страницы, что улучшает понимание страницы поисковыми системами и скринридерами. В целом, HTML является фундаментальным языком для создания веб-страниц, обеспечивая структуру, семантику и доступность, что делает его основным инструментом для создания веб-приложений.</p> <p>CSS (Cascading Style Sheets) - это мощный язык стилей, который позволяет веб-разработчикам полностью контролировать визуальное представление элементов HTML на веб-странице. Он работает по принципу каскада, что означает, что стили могут быть применены к элементам из разных источников и в различных уровнях приоритета, создавая сложные иерархии стилей. Одна из ключевых особенностей CSS - это его гибкость и масштабируемость. CSS предоставляет широкий спектр селекторов, которые позволяют выбирать элементы на странице с высокой точностью, от простых элементов до сложных комбинаций классов, идентификаторов и атрибутов. Это дает разработчикам возможность применять стили к конкретным элементам или группам элементов, обеспечивая высокую степень контроля над внешним видом страницы. Кроме того, CSS определяет модель бокса для каждого элемента, включая его размеры, отступы, границы и заполнение. Это позволяет разработчикам точно управлять размещением и компоновкой элементов на странице, создавая сложные макеты и дизайны. Одним из наиболее мощных аспектов CSS является его способность</p>						
Изм.	Лист	№ докум.	Подп.	Дата		

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подп.</i>	<i>Дата</i>

ДП 09.02.03 86.09.24ПЗ

Лист

---

18

модули в зависимости от требований проекта. Кроме того, Bootstrap 5 использует передовую технологию CSS, такую как CSS Grid и Flexbox, чтобы создавать адаптивные и отзывчивые макеты. Это обеспечивает корректное отображение веб-приложений на различных устройствах и экранах. Bootstrap 5 также включает в себя обширную библиотеку готовых компонентов интерфейса, таких как навигационные меню, модальные окна, карточки и многое другое. Эти компоненты могут быть легко настроены и кастомизированы, чтобы соответствовать требованиям проекта. В целом, Bootstrap 5 является мощным инструментом, который значительно ускоряет процесс создания адаптивных, функциональных и визуально привлекательных веб-приложений. Его использование позволяет разработчикам сосредоточиться на реализации бизнес-логики, не тратя время на создание базовой структуры и стилей интерфейса.

cdn.datatables - это мощная библиотека JavaScript, специализирующаяся на создании интерактивных таблиц данных, предоставляя широкий спектр возможностей для работы с информацией.

Основные возможности cdn.datatables:

- сортировка данных: Библиотека позволяет пользователям сортировать данные в таблице по различным критериям, таким как числовые значения, текстовые строки и даты. Это обеспечивает удобство при просмотре и анализе информации;
- фильтрация данных: cdn.datatables предоставляет возможность фильтровать данные в таблице в соответствии с заданными условиями. Это позволяет пользователям быстро находить необходимую информацию и упрощает навигацию по большим объемам данных;
- поиск данных: Библиотека поддерживает функцию поиска, позволяя пользователям быстро находить конкретные записи или значения в таблице. Это улучшает процесс поиска информации и экономит время пользователя;
- отображение данных: cdn.datatables предоставляет возможность настройки отображения данных в таблице, включая форматирование, выравнивание, цветовую разметку и другие аспекты. Это позволяет создавать информативные и удобочитаемые таблицы.

Sweet Alert 2 - это инновационная библиотека для создания визуальных уведомлений, которая революционизировала способ информирования пользователей о событиях и результатах операций в веб-приложениях. Она обеспечивает создание интерактивных, привлекательных и функциональных уведомлений, которые могут быть настроены в соответствии с дизайном и стилем приложения.

Возможности Sweet Alert 2:

- динамические анимации: Sweet Alert предлагает спектр динамических анимаций, которые могут быть настроены для создания привлекательных;
- модульная архитектура: Библиотека имеет модульную архитектуру, что позволяет легко добавлять или удалять функциональность в зависимости от требований приложения;

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	ДП 09.02.03 86.09.24ПЗ					Лист	
Изм.	Лист	№ докум.	Подп.	Дата						19	

- широкие возможности кастомизации: Sweet Alert обеспечивает высокую степень кастомизации, позволяя разработчикам настроить внешний вид, поведение и содержимое уведомлений в соответствии с требованиями приложения;
- удобство использования: библиотека имеет простой и интуитивный API, что делает ее легкой в использовании, даже для разработчиков без опыта работы с визуальными уведомлениями.

Sweet Alert 2 является мощным инструментом для создания визуальных уведомлений, которые играют важную роль в информировании пользователей о событиях и результатах операций в веб-приложениях. Она обеспечивает создание интерактивных, привлекательных и функциональных уведомлений, которые могут быть настроены в соответствии с дизайном и стилем приложения.

jQuery - это мощная библиотека JavaScript, которая значительно упрощает и ускоряет разработку веб-приложений. Она предоставляет удобный API для работы с DOM (Document Object Model), обработкой событий и созданием анимаций.

Основные возможности jQuery:

- манипуляции с DOM: jQuery позволяет легко выбирать, изменять и манипулировать элементами HTML-документа с помощью мощных селекторов и методов. Это значительно упрощает доступ и изменение структуры веб-страницы;
- обработка событий: библиотека предоставляет простой и интуитивно понятный API для привязки обработчиков событий к элементам DOM. Это позволяет создавать интерактивные веб-приложения, реагирующие на действия пользователя;
- анимации и эффекты: jQuery включает в себя богатый набор встроенных анимаций и эффектов, которые могут быть применены к элементам страницы. Это позволяет создавать динамичные и привлекательные интерфейсы;
- AJAX-запросы: библиотека упрощает отправку асинхронных запросов на сервер и обработку ответов. Это делает возможным обновление содержимого страницы без перезагрузки, улучшая пользовательский опыт;
- кроссбраузерность: jQuery обеспечивает кроссбраузерную совместимость, скрывая различия в реализации DOM API между разными браузерами. Это позволяет писать код, который будет корректно работать в большинстве современных браузеров;
- плагины и расширения: Существует огромное количество плагинов и расширений для jQuery, которые добавляют дополнительную функциональность, такую как слайдеры, всплывающие окна, формы и многое другое. Это позволяет быстро реализовывать сложные возможности без необходимости писать код с нуля.

jQuery стала одной из самых популярных и широко используемых библиотек JavaScript благодаря своей простоте, мощи и гибкости. Она значительно ускоряет разработку веб-приложений, позволяя разработчикам

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	<p>Это позволяет создавать интерактивные веб-приложения, реагирующие на действия пользователя;</p> <ul style="list-style-type: none"><li>– анимации и эффекты: jQuery включает в себя богатый набор встроенных анимаций и эффектов, которые могут быть применены к элементам страницы. Это позволяет создавать динамичные и привлекательные интерфейсы;</li><li>– AJAX-запросы: библиотека упрощает отправку асинхронных запросов на сервер и обработку ответов. Это делает возможным обновление содержимого страницы без перезагрузки, улучшая пользовательский опыт;</li><li>– кроссбраузерность: jQuery обеспечивает кроссбраузерную совместимость, скрывая различия в реализации DOM API между разными браузерами. Это позволяет писать код, который будет корректно работать в большинстве современных браузеров;</li><li>– плагины и расширения: Существует огромное количество плагинов и расширений для jQuery, которые добавляют дополнительную функциональность, такую как слайдеры, всплывающие окна, формы и многое другое. Это позволяет быстро реализовывать сложные возможности без необходимости писать код с нуля.</li></ul> <p>jQuery стала одной из самых популярных и широко используемых библиотек JavaScript благодаря своей простоте, мощи и гибкости. Она значительно ускоряет разработку веб-приложений, позволяя разработчикам</p>
Изм.	Лист	№ докум.	Подп.	Дата	

ДП 09.02.03 86.09.24ПЗ

Лист
20

фокусироваться на создании функциональности, а не на решении кроссбраузерных проблем и низкоуровневых деталей DOM API.

JSON Web Token (JWT) - это открытый стандарт (RFC 7519) для создания токенов, используемых для безопасной передачи информации между сторонами в компактной и самодостаточной форме. Эти токены могут быть использованы для авторизации и аутентификации, обеспечивая целостность и безопасность передаваемых данных.

JWT состоит из трех частей, разделенных точками:

- заголовок (Header): содержит тип токена (JWT) и алгоритм шифрования, например HMAC SHA256 или RSA;
- полезная нагрузка (Payload): содержит утверждения, такие как эмитент, время истечения срока действия и другие данные пользователя;
- подпись (Signature): создается путем кодирования заголовка и полезной нагрузки с секретным ключом с использованием алгоритма, указанного в заголовке.

Основные преимущества использования JWT:

- Компактность: JWT токены компактны, что делает их удобными для передачи в URL-адресах, заголовках HTTP и в теле сообщений POST;
- самодостаточность: вся необходимая для аутентификации информация находится в самом токене, что исключает необходимость дополнительных запросов на сервер;
- безопасность: токены подписываются с использованием криптографии, что гарантирует целостность и подлинность данных;
- многоплатформенность: JWT совместим с множеством платформ, включая Node.js, Java, .NET и многие другие.

Библиотека jsonwebtoken версии 9.0.2 для Node.js предоставляет простой и удобный API для работы с JWT токенами, включая генерацию, верификацию и декодирование токенов.

В целом, JSON Web Token (JWT) является надежным и эффективным стандартом для авторизации и аутентификации в веб-приложениях, обеспечивая безопасную передачу данных между клиентом и сервером.

Razor - это шаблонизатор для .NET, который позволяет создавать динамические веб-страницы, сочетая в себе мощь языка программирования C# и простоту разметки HTML. Он предоставляет интуитивно понятный синтаксис для встраивания серверного кода в HTML-шаблоны, что делает процесс разработки веб-приложений на .NET более эффективным и продуктивным.

Основные особенности Razor:

- встроенный C#: Razor позволяет использовать C# для добавления логики и функциональности на сервере непосредственно в HTML-шаблоны. Это обеспечивает плавную интеграцию между представлением и бизнес-логикой;
- простой синтаксис: Синтаксис Razor интуитивно понятен и легок в освоении для разработчиков, знакомых с HTML и C#. Он использует специальные символы, такие как @, для переключения между HTML и C#-кодом;

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.
Подп. и дата	

Изм.	Лист	№ докум.	Подп.	Дата
------	------	----------	-------	------

ДП 09.02.03 86.09.24ПЗ

- повторное использование кода: Razor поддерживает создание повторно используемых компонентов, частичных представлений и макетов, что позволяет организовывать код в модульные и легко поддерживаемые структуры;
- безопасность: Razor автоматически кодирует специальные символы в выводимых данных, что помогает защитить от уязвимостей, таких как межсайтовый скриптинг (XSS);[6]
- интеграция с ASP.NET: Razor тесно интегрируется с фреймворком ASP.NET, предоставляя доступ к его функциям и возможностям, таким как маршрутизация, аутентификация и управление состоянием;
- расширяемость: Razor можно расширять с помощью пользовательских хелперов и тегов, что позволяет создавать специализированные инструменты и абстракции для конкретных предметных областей;

б) серверная часть веб-приложения - это часть веб-приложения, которая выполняется на сервере и отвечает за обработку логики приложения, взаимодействие с базой данных, управление пользовательскими сессиями и безопасность. Она предоставляет API для клиентской части и обрабатывает запросы от пользователей.

C# - это современный, объектно-ориентированный язык программирования, разработанный Microsoft для платформы .NET. Он сочетает в себе простоту и выразительность, предоставляя разработчикам мощные средства для создания широкого спектра приложений, от веб-сайтов до игр и мобильных приложений.

C# был впервые представлен в 2000 году как часть платформы .NET. С тех пор он постоянно развивается, добавляя новые функции и улучшая существующие. Последняя версия, C# 11, была выпущена в 2022 году вместе с .NET 7 и включает множество интересных возможностей, таких как улучшенная поддержка шаблонов, новые литералы и многое другое.[7]

Ключевые особенности:

- объектно-ориентированность: C# является объектно-ориентированным языком, что позволяет создавать модульные и расширяемые приложения;[8]
- безопасность типов: C# строго типизирован, что помогает предотвратить ошибки во время выполнения и обеспечивает надежность кода;
- интеграция с .NET: C# тесно интегрирован с платформой .NET, предоставляя доступ ко всем ее возможностям и библиотекам;
- выразительность: C# предоставляет множество выразительных средств, таких как LINQ, async/await и лямбда-выражения, которые упрощают решение сложных задач;
- кроссплатформенность: C# может использоваться для создания кроссплатформенных приложений, работающих на Windows, macOS и Linux.

C# используется для создания широкого спектра приложений, включая:

- веб-приложения и веб-сервисы: C# может использоваться для создания веб-приложений и веб-сервисов с помощью ASP.NET Core;

Инв. № подл.	Подп. и дата					ДП 09.02.03 86.09.24ПЗ	Лист 22
	Инв. № дубл.						
	Взам. Инв. №						
	Подп. и дата						
	Изм.	Лист	№ докум.	Подп.	Дата		

- мобильные приложения: C# может использоваться для создания мобильных приложений для iOS и Android с помощью Xamarin или .NET MAUI;
- игры и приложения с высокой производительностью: C# может использоваться для создания игр и других приложений с высокой производительностью с помощью Unity или Godot;
- настольные приложения: C# может использоваться для создания настольных приложений с помощью Windows Forms или WPF;
- облачные и серверные приложения: C# может использоваться для создания облачных и серверных приложений с помощью .NET Core или .NET.

C# - это мощный и выразительный язык программирования, который идеально подходит для создания широкого спектра приложений на платформе .NET. Его объектно-ориентированность, безопасность типов, интеграция с .NET и кроссплатформенность делают его привлекательным выбором для разработчиков, стремящихся создавать надежные, масштабируемые и эффективные приложения.

.NET - это мощная и универсальная платформа разработки, предназначенная для создания различных типов приложений, от веб-сайтов до мобильных приложений и настольных программ. Одним из значимых релизов этой платформы является .NET 7, который был выпущен Microsoft в 2022 году. .NET 7 представляет собой современную версию платформы, ориентированную на улучшение производительности, расширение возможностей разработки и обеспечение единого подхода к созданию приложений для различных платформ.

Основные особенности .NET 7 включают:

- улучшенная производительность: .NET 7 фокусируется на повышении производительности за счет усовершенствованного JIT-компилятора, аппаратных встроенных функций и переработанного сборщика мусора. Это позволяет создавать более эффективные и быстрые приложения;
- унификация разработки: .NET 7 обеспечивает более развитую унификацию разработки настольного и мобильного программного обеспечения, веб-приложений, облачных, игровых, IoT и ИИ-решений. Это делает процесс разработки более гибким и эффективным;
- новые технологии и инструменты: Вместе с .NET 7 были выпущены Entity Framework Core 7, ASP.NET Core 7, .NET MAUI, C# 11, F# 7, Windows Presentation Foundation, Windows Forms, Windows Presentation Foundation и Orleans 7. Эти инструменты обогащают возможности разработчиков и расширяют функциональность платформы.[9]

.NET 7 также предлагает улучшенную поддержку облачных нативных сценариев, инструменты для работы с легаси-кодом и упрощение работы с контейнерами. Этот релиз стал важным шагом в развитии .NET, обеспечивая разработчикам современные инструменты и технологии для создания инновационных и высокопроизводительных приложений.

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

ASP.NET Core - это свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений на платформе .NET с открытым исходным кодом. Он разрабатывается компанией Microsoft совместно с сообществом и имеет большую гибкость и масштабируемость.

ASP.NET Core имеет несколько ключевых особенностей и функций, которые делают его привлекательным для разработчиков веб-приложений:

- кроссплатформенность: ASP.NET Core может работать на различных платформах, включая Windows, Linux и macOS;
- открытый исходный код: ASP.NET Core имеет открытый исходный код, что позволяет разработчикам вносить изменения и улучшения в фреймворк;
- высокая производительность: ASP.NET Core оптимизирован для высокой производительности и может обрабатывать большое количество запросов одновременно;
- модульность: ASP.NET Core имеет модульную архитектуру, что позволяет легко добавлять или удалять функции в зависимости от потребностей приложения;
- поддержка различных типов приложений: ASP.NET Core поддерживает создание различных типов приложений, включая веб-API, интерактивные веб-приложения, приложения на базе шаблонов MVC и приложения реального времени.

ASP.NET Core может быть использован в различных сценариях разработки, включая:

- создание веб-API: ASP.NET Core может быть использован для создания веб-API, которые могут быть вызваны из различных приложений;
- разработка интерактивных веб-приложений: ASP.NET Core может быть использован для создания интерактивных веб-приложений с помощью Blazor, который позволяет создавать интерактивные веб-интерфейсы с помощью C# и Razor;
- разработка приложений на базе шаблонов MVC: ASP.NET Core может быть использован для создания приложений на базе шаблонов MVC, которые имеют четкое разделение ответственностей между моделью, представлением и контроллером.

ASP.NET Core - это мощный и гибкий фреймворк для создания веб-приложений, который имеет открытый исходный код и может работать на различных платформах. Он оптимизирован для высокой производительности и имеет модульную архитектуру, что позволяет легко добавлять или удалять функции в зависимости от потребностей приложения. ASP.NET Core является идеальным выбором для разработчиков, которые хотят создавать быстрые и безопасные кроссплатформенные или облачные веб-приложения и службы.

Entity Framework Core (EF Core) - это объектно-реляционное отображение (ORM), разработанное Microsoft для .NET, которое значительно упрощает взаимодействие с базами данных в приложениях. Оно предоставляет абстракцию над базами данных, позволяя разработчикам работать с данными в виде объектов

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ



.NET без необходимости глубокого знания SQL или деталей реализации базы данных.

Основные возможности EF Core:

- автоматическое создание моделей данных: EF Core может автоматически создавать классы .NET, представляющие таблицы базы данных, на основе существующей схемы базы данных;
- запросы LINQ: EF Core поддерживает запросы LINQ, позволяя разработчикам выражать запросы к данным с помощью синтаксиса C#, который затем преобразуется в эффективные запросы базы данных;
- миграции базы данных: EF Core предоставляет систему миграций, которая позволяет управлять изменениями схемы базы данных с помощью кода C#, обеспечивая контроль версий и возможность откатывать изменения;
- поддержка множества баз данных: EF Core поддерживает широкий спектр баз данных, включая SQL Server, PostgreSQL, MySQL, SQLite и другие, что делает его гибким выбором для различных сценариев;
- расширяемость: EF Core является расширяемым и может быть адаптирован к специфическим требованиям приложения с помощью пользовательских провайдеров баз данных, конвенций именования и других механизмов расширения.

EF Core 7, выпущенный в 2022 году, включает ряд улучшений и новых функций:

- поддержка столбцов JSON: EF Core 7 добавляет поддержку хранения и манипулирования данными в формате JSON в базе данных;
- улучшенная производительность: Производительность EF Core 7, особенно при сохранении изменений, значительно улучшилась по сравнению с предыдущими версиями;
- пользовательские шаблоны обратной инженерии: Теперь можно настроить шаблонный код при обратном проектировании модели EF из базы данных, предоставляя разработчикам больший контроль над процессом маппинга.

EF Core может использоваться в различных типах приложений .NET, включая веб-приложения, мобильные приложения, настольные приложения и микросервисы. Оно обеспечивает абстракцию над базами данных, позволяя разработчикам фокусироваться на бизнес-логике приложения, а не на деталях доступа к данным.

В целом, Entity Framework Core является мощным и гибким инструментом для работы с данными в приложениях .NET. Его автоматическое создание моделей данных, поддержка запросов LINQ, миграций базы данных и множества баз данных делают его идеальным выбором для большинства сценариев разработки .NET.

Язык SQL (Structured Query Language) в СУБД - это язык для управления и обработки данных в базах данных. Он был разработан в 1970-х годах и с тех пор стал стандартом для управления реляционными базами данных.

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.
Подп. и дата	
Инв. № подл.	

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

Лист  
25

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

пользователям. Это обеспечивает стабильную работу системы, оперативное реагирование на возможные проблемы и постоянное совершенствование функционала.

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

Лист
27

## 2 СПЕЦИАЛЬНАЯ ЧАСТЬ

### 2.1 Постановка задачи

#### 2.1.1 Формулировка задачи

Разработка информационной системы обратной связи и учёта задач работников АО УПП «Вектор».

Разрабатываемая программа должна:

- предоставить работникам возможность авторизоваться для отображения им задач и обращений, заданным им и от них, видеть статус выполнения задач и обращений в веб-интерфейсе, изменять статус задания и обращения, подавать обращения своему начальнику и задания своим подчинённым, выполнять выборку задач и обращений по ФИО работника, статусу выполнения, диапазону дат, формировать отчёт в удобном для печати формате по задачам, выданным их подчиненным с заданными параметрами (ФИО работника, статус выполнения, диапазон дат);
- шифровать текстовые данные при передаче в базу данных и дешифровать для представления работнику с помощью симметричного алгоритма блочного шифрования Advanced Encryption Standart;
- сохранять действия администраторов в отдельный файл;
- предоставить возможность администратору редактирования базы данных «Информационная система обратной связи.dbo» через веб-интерфейс (создавать, редактировать, удалять пользователей и отделы);
- ограничивать ввод только верной информацией (число в числовые поля и т.д.), а при вводе неверной информации выводить ошибку.

#### 2.1.2 Входные данные

1. База данных «Информационная система обратной связи» состоит из 5 объектов.

Данные предоставлены в таблицах 2.1, 2.2, 2.3, 2.4, 2.5.

Таблица 2.1 – «Отделы»:

Имя поля	Тип	Комментарии, ограничения поля
1	2	3
Код отдела	Счетчик	Ключевое, обязательное, длинное целое (4 байта), последовательные значения от 1, индексированное, совпадения не допускаются
Наименование отдела	Текстовый	Обязательное, до 255 символов
Ключ шифрования	Текстовый	Обязательное, до 255 символов
Инициализирующий вектор шифрования	Текстовый	Обязательное, до 255 символов

ДП 09.02.03 86.09.24ПЗ

Лист

28

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	

Изм.	Лист	№ докум.	Подп.	Дата

Таблица 2.2 – «Роли»:

Имя поля	Тип	Комментарии, ограничения поля
1	2	3
Код роли	Счетчик	Ключевое, обязательное, длинное целое (4 байта), последовательные значения от 1, индексированное, совпадения не допускаются
Наименование роли	Текстовый	Обязательное, до 255 символов

Таблица 2.3 – «Задачи»:

Имя поля	Тип	Комментарии, ограничения поля
1	2	3
Код задачи	Счетчик	Ключевое, обязательное, длинное целое (4 байта), последовательные значения от 1, индексированное, совпадения не допускаются
Статус выполнения задачи	Логический	Обязательное, ограничивается только вводом «Истина» или «Ложь»

Таблица 2.4 – «Пользователи»:

Имя поля	Тип	Комментарии, ограничения поля
1	2	3
Код пользователя	Счетчик	Ключевое, обязательное, длинное целое (4 байта), последовательные значения от 1, индексированное, совпадения не допускаются
ФИО	Текстовый	Обязательное, до 255 символов

Таблица 2.5 – «Иерархия»:

Имя поля	Тип	Комментарии, ограничения поля
1	2	3
Код записи	Счетчик	Ключевое, обязательное, длинное целое (4 байта), последовательные значения от 1, индексированное, совпадения не допускаются
Код начальника	Текстовый	Обязательное, ограничивается списком подстановки из таблицы «Пользователи»

### 2.1.3 Выходные данные

- База данных «Информационная система обратной связи.db» обновлённая;
- logAdmin - файл с записями действий администраторов (.log);
- отчет в удобном для печати формате по задачам, выданным их подчиненным с заданными параметрами (ФИО работника, статус выполнения, диапазон дат);
- сообщения об ошибках.

ДП 09.02.03 86.09.24ПЗ

Лист

29

Подп. и дата

Инв. № дубл.

Взам. Инв. №

Подп. и дата

Инв. № подл.

Изм. Лист № докум. Подп. Дата

2.2 Функциональная диаграмма IDEF0

2.2.1 Точка зрения пользователя

Представлена функциональная диаграмма IDEF0 базы данных «Информационная система обратной связи» с точки зрения пользователя (рис 2.1, 2.2)

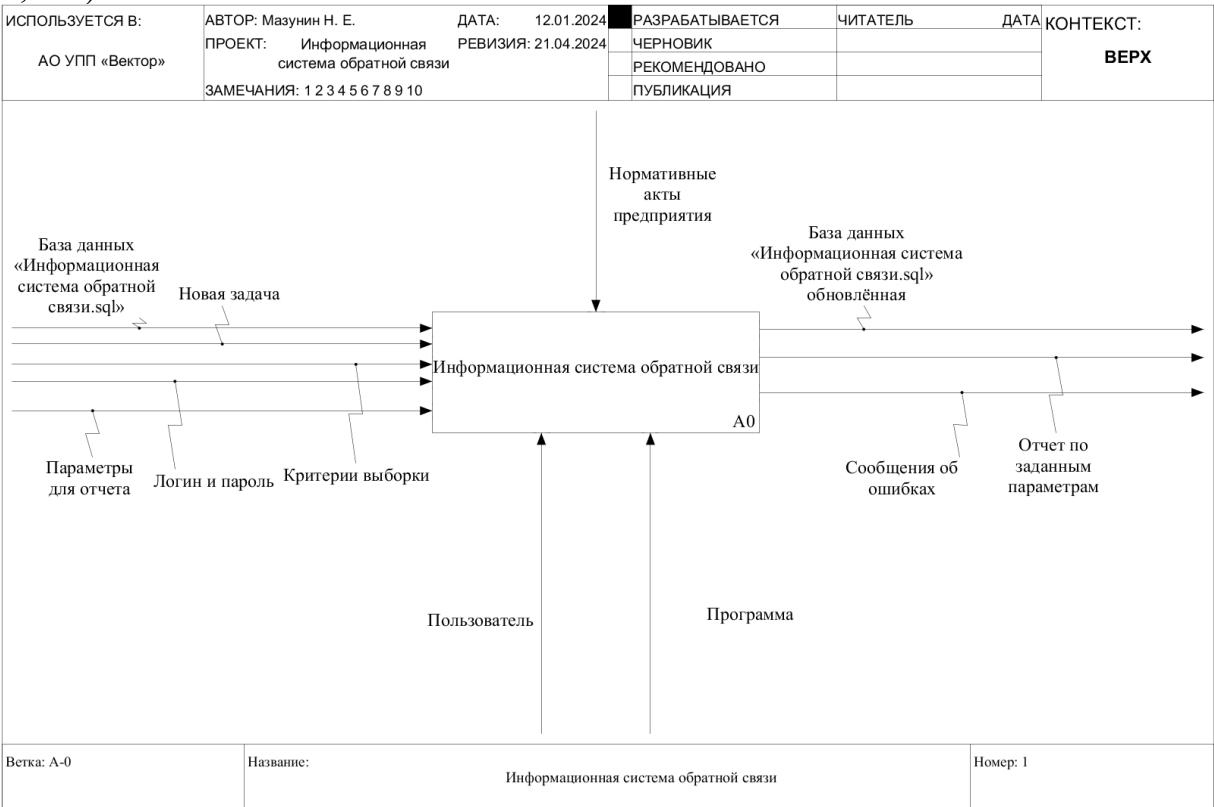


Рисунок 2.1 – функцияональная диаграмма IDEF0 с точки зрения пользователя

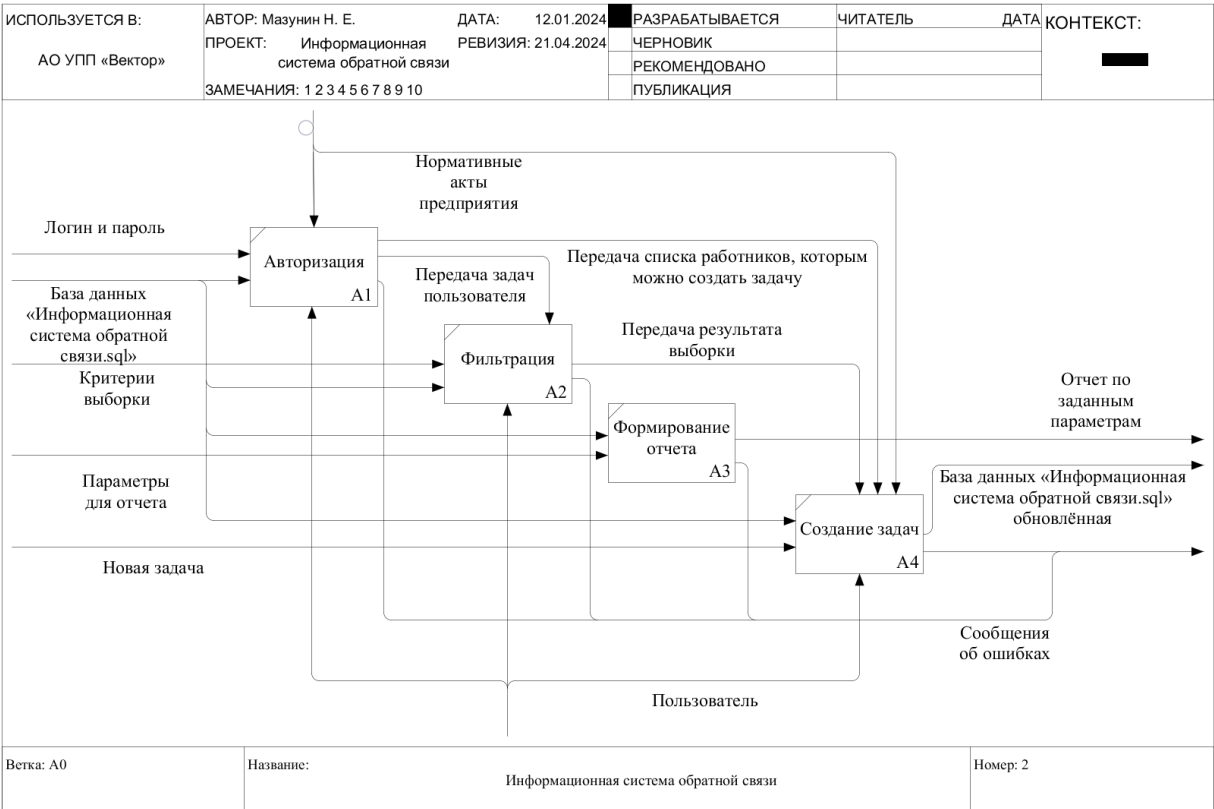


Рисунок 2.2 – функциональная диаграмма IDEF0 с точки зрения пользователя

Подп. и дата

Инв. № дубл.

Взам. Инв. №

Подп. и дата

Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

Лист

30

## 2.2.2 Точка зрения администратора

Представлена функциональная диаграмма IDEF0 базы данных «Информационная система обратной связи» с точки зрения администратора (рис 2.3, 2.4)

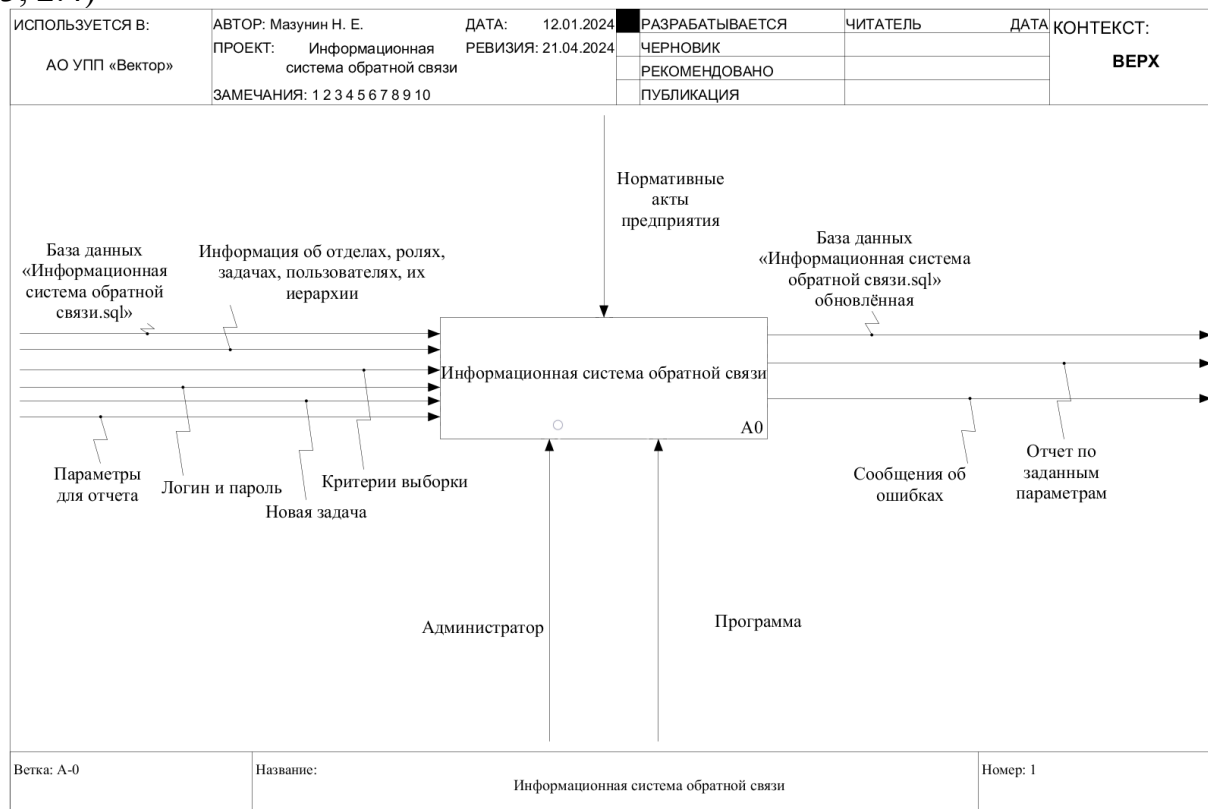


Рисунок 2.3 – функциональная диаграмма IDEF0 с точки зрения администратора

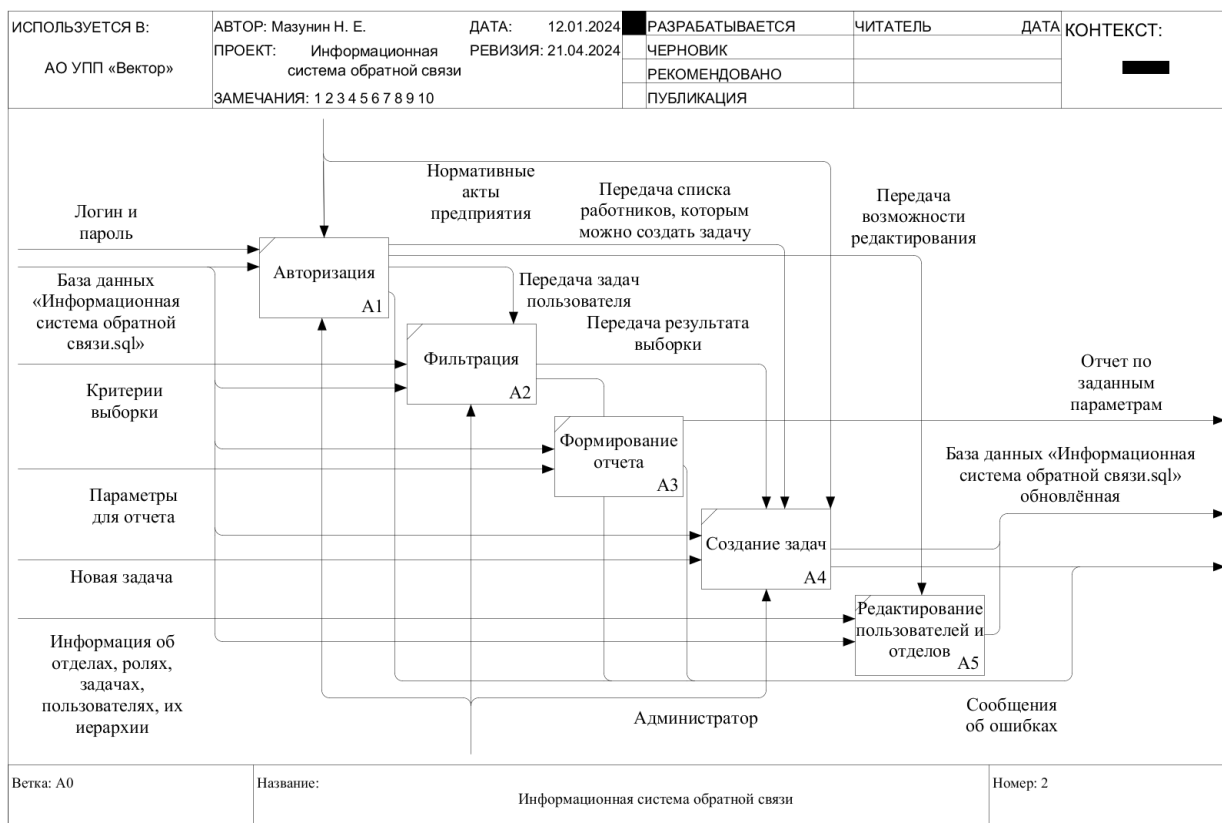


Рисунок 2.4 – функциональная диаграмма IDEF0 с точки зрения администратора

2.3 Диаграмма потоков данных DFD

Представлена диаграмма потоков данных DFD базы данных «Информационная система обратной связи» (рис. 2.5)

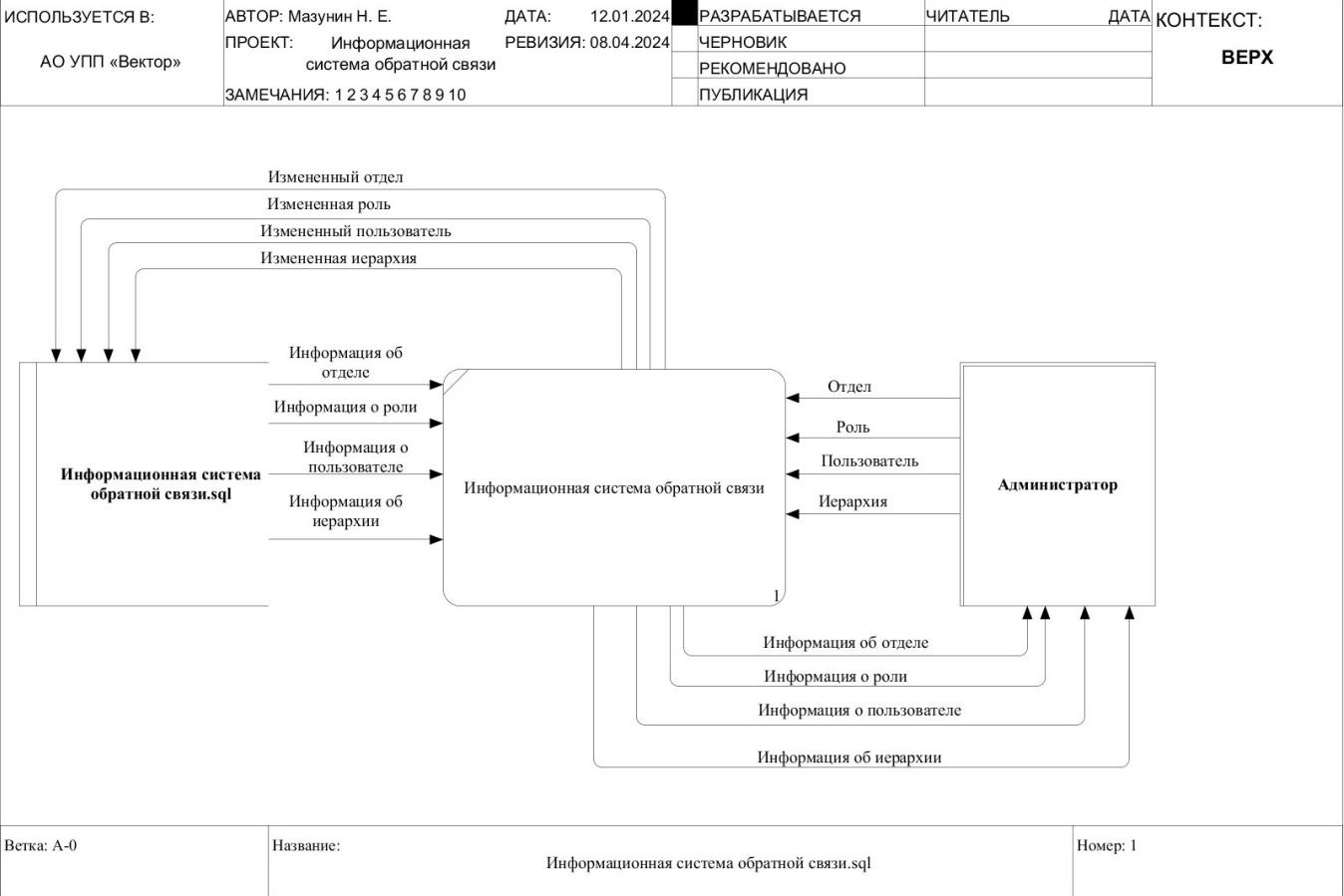


Рисунок 2.5 – диаграмма потоков данных DFD

2.4 Диаграмма «сущность-связь»

Представлена схема базы данных «Информационная система обратной связи» (рис. 2.6)



Рисунок 2.6 – диаграмма «сущность-связь»

Подп. и дата

Инв. № дубл.

Взам. Инв. №

Подп. и дата

Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата
------	------	----------	-------	------

ДП 09.02.03 86.09.24ПЗ



## 2.5 Схема данных

Представлена схема данных программы «Информационная система обратной связи» (рис 2.7)

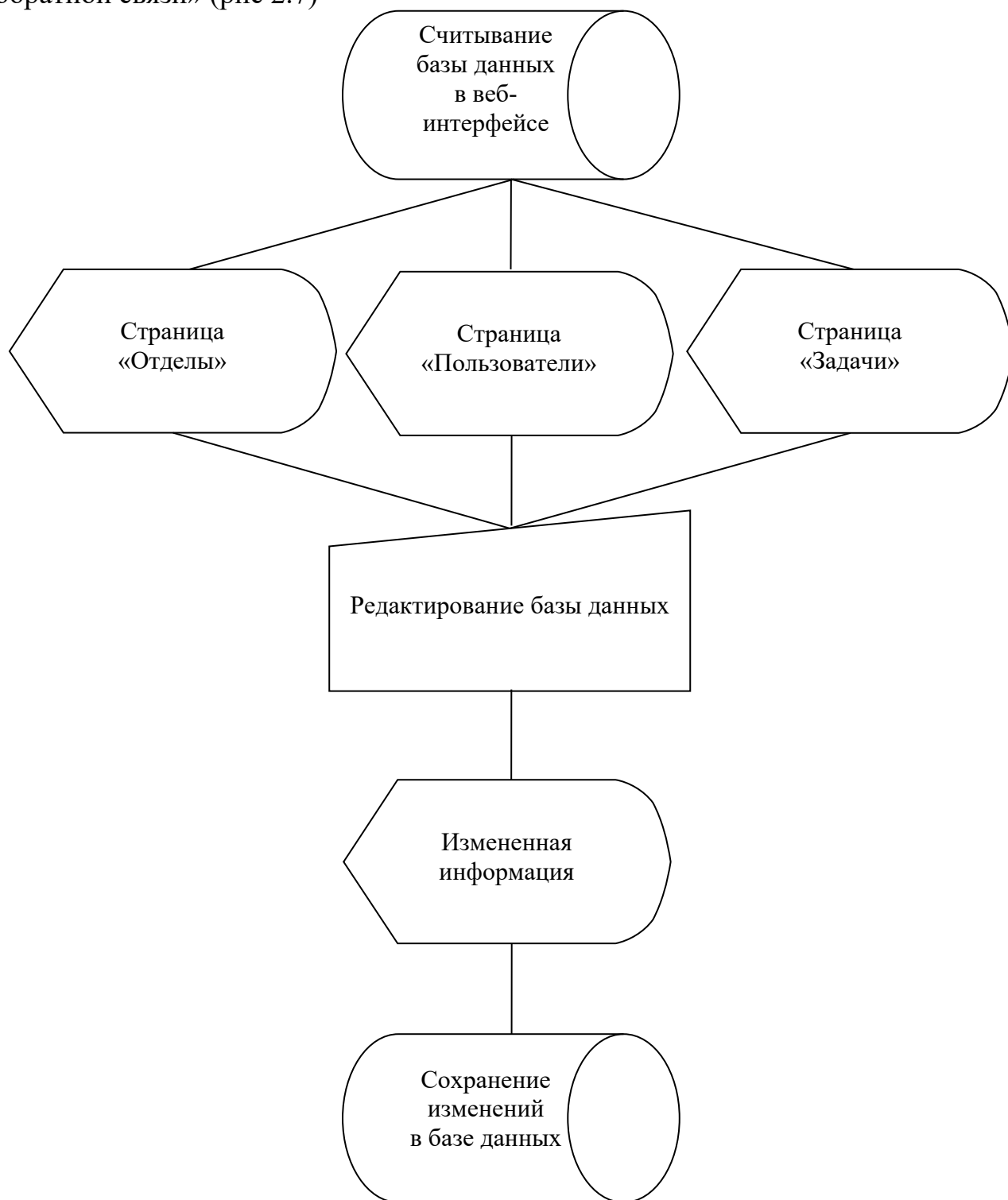


Рисунок 2.7 – схема данных

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

Лист

33

## 2.6 Структура программы

Представлена структура программы «Информационная система обратной связи» (рис 2.8)

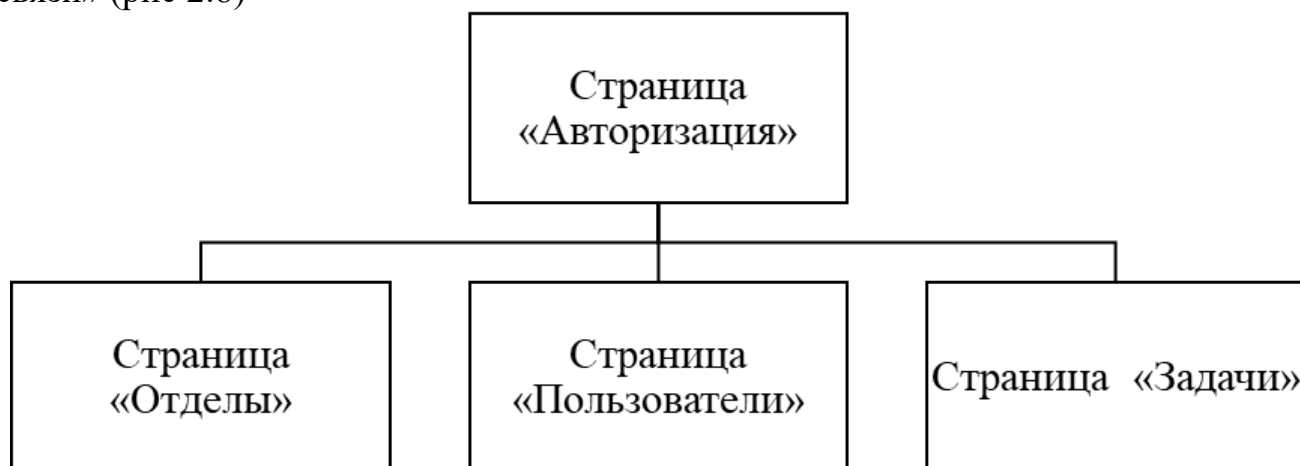


Рисунок 2.8 – структура программы

## 2.7 Описание программы

### 2.7.1 Общие сведения

Название программы: «Информационная система обратной связи и учёта задач работников АО УПП «Вектор»».

Для функционирования программы необходимо наличие установленной программы Microsoft SQL Server Express, папки «Информационная система обратной связи», .NET 7.0, интернет соединение.

Программа написана при использовании программы разработки баз данных SQL Server Management Studio 19.

Операционная система Microsoft Windows 7/8/10/11.

### 2.7.2 Функциональное назначение

Программа предназначена для упрощения взаимодействия между сотрудниками АО УПП «Вектор» и формирования отчетов о проделанной работе их подчиненными.

### 2.7.3 Используемые технические средства

Минимальные системные требования: MS Windows 7 или выше, Intel Core i3 с частотой 4,2 GHz, 1 ГБ оперативной памяти, видеокарта с поддержкой SVGA (800x600, 65536 цветов), подключения к интернету, манипулятор «мышь» и клавиатура.

## 2.7.4 Установка программы

Для установки программы следует установить Microsoft SQL Server Express и .NET 7.0 с официального сайта и скопировать папку «Информационная система обратной связи».

## 2.8 Описание алгоритма

### 2.8.1 Модуль «UserService.Create»

Код модуля представлен в приложении А.

### 2.8.1.1 Функциональное назначение

Сохранение пользователя в базу данных, если его логин уникален.

### 2.8.1.2 Описание логической структуры

- получение всех логинов пользователей из таблицы с пользователями;
- выполнение поиска первого расшифрованного логина с заданным

логином в списке пользователей;

- проверка, существует ли пользователь с таким же логином, если пользователь с таким логином уже существует, то возвращение ошибки «Пользователь с таким логином уже существует», а если нет, то продолжение выполнения алгоритма;
- установка случайных значений в ключ и инициализирующий вектор шифрования;
- если начальника нет, то создается сущность пользователя с шифрованными ФИО, логином и паролем, а если есть начальник, то в сущность записывается номер пользователя, который является начальником;
- сохранение нового пользователя в базе данных;
- возвращение ответа с кодом успешного выполнения и описанием «Пользователь создан».

### 2.8.1.3 Входные данные

Модель создания пользователя(ФИО, логин, пароль, роль, код отдела и код начальника, если начальник есть).

### 2.8.1.4 Выходные данные

- сообщение об успехе;
- сообщение об ошибке.

Код модуля представлен в приложении а.

## 2.9 Инструкция пользователя

### 2.9.1 Назначение программы

Разрабатываемая программа предназначена для упрощения взаимодействия между сотрудниками АО УПП «Вектор» и формирования отчетов о проделанной работе их подчиненными.

### 2.9.2 Условия выполнения программы

Минимальные системные требования: MS Windows 7 или выше, Intel Core i3 с частотой 4,2 GHz, 1 ГБ оперативной памяти, видеокарта с поддержкой SVGA (800x600, 65536 цветов), подключения к интернету, манипулятор «мышь» и клавиатура.

### 2.9.3 Установка программы

Для установки программы следует установить Microsoft SQL Server Express и .NET 7.0 с официального сайта и скопировать папку «Информационная система обратной связи».

### 2.9.4 Выполнение программы

Для функционирования программы необходимо:

- операционная система MS Windows 7 или выше;
- подключение к интернету;
- установленный .NET 7.0;
- установленный MS SQL Express.

### 2.9.5 Основные элементы управления

В программе используются следующие элементы управления:

- манипулятор мышь
- клавиатура.

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	ДП 09.02.03 86.09.24ПЗ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		35

## 2.9.6 Работа с программой

### 2.9.6.1 Авторизация

Зайдите на сайт с программой, введите свой логин и пароль, нажмите на кнопку «Войти».

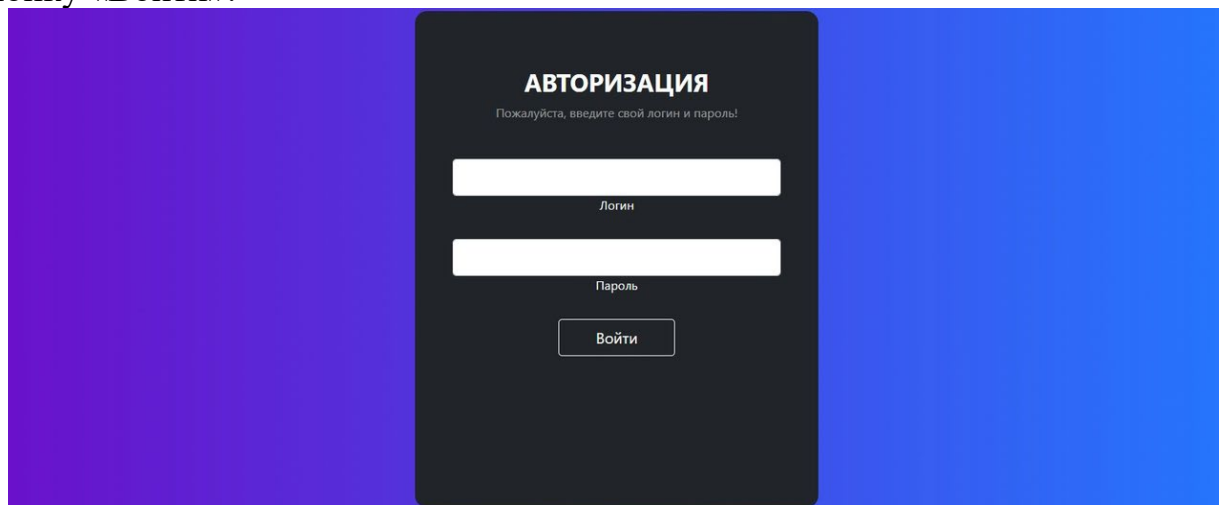


Рисунок 2.9 – авторизация

### 2.9.6.1 Задачи

Для создания задачи необходимо заполнить соответствующие поля, выбрать закрепленного работника и нажать на кнопку «Создать».

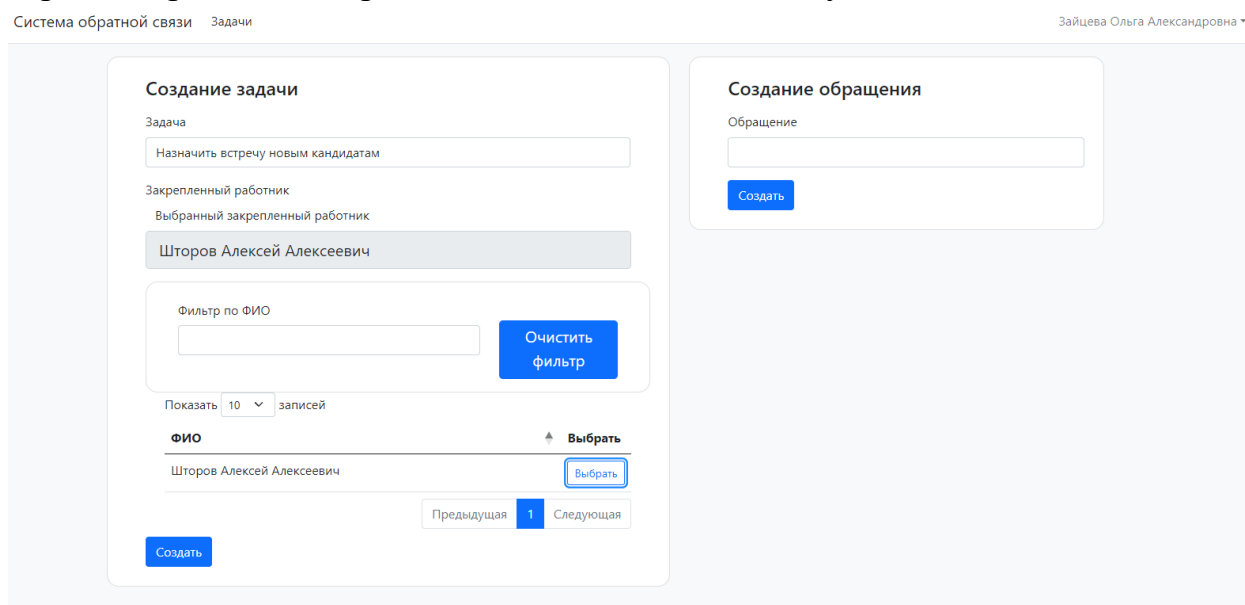


Рисунок 2.10 – создание задачи

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.

Изм.	Лист	№ докум.	Подп.	Дата
------	------	----------	-------	------

ДП 09.02.03 86.09.24ПЗ

Лист  
36

Для того, чтобы поменять статус выданной вам задачи наведите на её нынешний статус в таблице «Полученные задачи», кликните и выберите её новый статус.

Полученные задачи

Фильтры

Фильтр по названию задачи

Фильтр по статусу выполнения

Отображать по дате создания

Очистить фильтры

Показать 10 записей

Задача	Статус выполнения	Дата создания задачи
Рассмотреть новых кандидатов	Не взято	4 июня 2024 г.
Составить отчёт до 08.06.2024	Не взято	4 июня 2024 г.

Рисунок 2.11 – смена статуса выполнения задачи

2.9.6.1 Отчет

В таблице «Выданные задачи», введите нужные параметры в поля рядом с таблицей и нажмите на кнопку «Создать отчёт».

Выданные задачи

Фильтры

Фильтр по названию задачи

Фильтр по работнику

Фильтр по статусу выполнения

Отображать по дате создания

Отображать по дате выполнения

Очистить фильтры

Отчёт

Создать отчёт

Показать 10 записей

Задача/Обращение	Работник	Статус выполнения	Дата создания задачи	Дата выполнения задачи
Назначить встречу новым кандидатам	Шторов Алексей Алексеевич	Не взято	4 июня 2024 г.	
Найти новых кандидатов	Шторов Алексей Алексеевич	Выполнено	4 июня 2024 г.	4 июня 2024 г.
Отсортировать новых кандидатов	Шторов Алексей Алексеевич	В работе	4 июня 2024 г.	

Рисунок 2.12 – создание отчета

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.
Подп. и дата	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата
------	------	----------	-------	------

В появившемся окне выберите куда будет сохранён отчёт и нажмите кнопку «Сохранить».

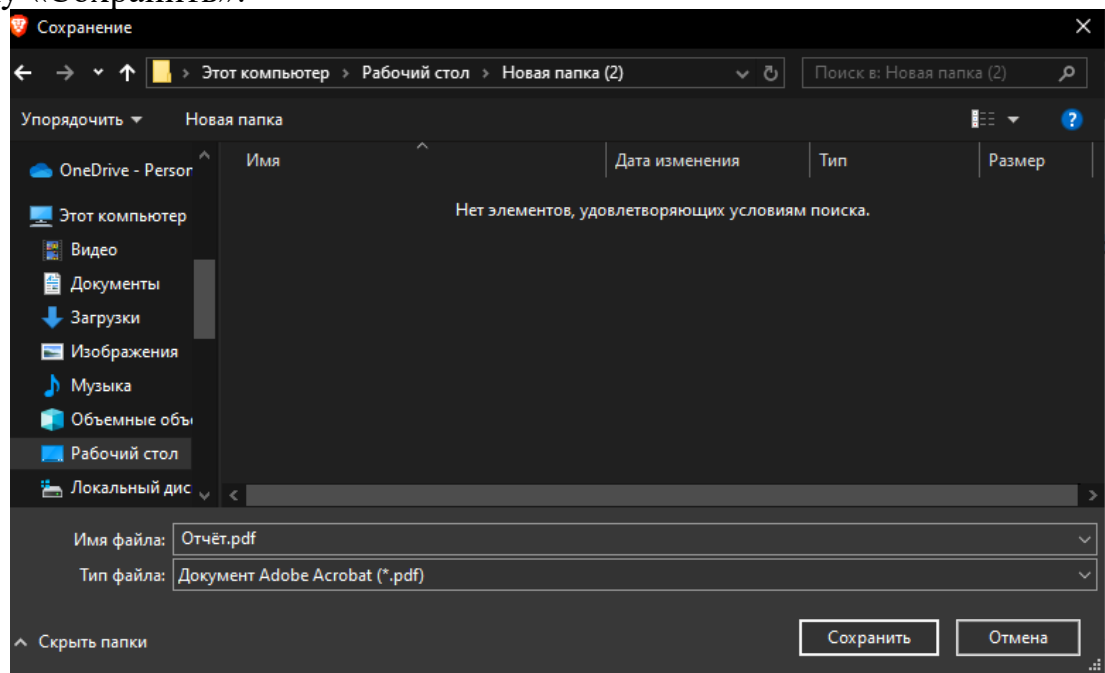


Рисунок 2.13 – сохранение отчета.

В получившемся файле для вашего удобства уже будет добавлено место для даты и подписи.

Название	Статус	Создано	Завершено	Работник
Найти новых кандидатов	Выполнено	4 июня 2024 г.	4 июня 2024 г.	Шторов Алексей Алексеевич
Отсортировать новых кандидатов	В работе	4 июня 2024 г.		Шторов Алексей Алексеевич
Назначить встречу новым кандидатам	Не взято	4 июня 2024 г.		Шторов Алексей Алексеевич

Подпись: \_\_\_\_\_ Дата: \_\_\_\_\_

Рисунок 2.14 – отчет.

Результаты работы программы представлены в приложении б.

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.

Изм.	Лист	№ докум.	Подп.	Дата
------	------	----------	-------	------

ДП 09.02.03 86.09.24ПЗ

Лист
38

### 3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

#### 3.1 Экономическое обоснование эффективности программы

Данная задача относится к классу "Баз данных"

Накладные расходы (в %) - 30

Отчисления на социальные нужды (в %) - 30

Для расчета стоимости программного продукта используются данные о временных и стоимостных затратах на создание программного продукта из таблиц 3.1 – 3.2

Таблица 3.1 – Временные затраты на создание программного продукта

Этапы выполняемых работ	Время работы программиста, час	Время работы постановщика, час
1	2	3
Формирование требований ПП	2	6
Разработка концепции ПП	39	5
Постановка задачи	55	8
Структурирование данных, программы	10	5
Отладка	128	33
Оформление отчета	4	3
Ввод в действие	1	1

Таблица 3.2 – Данные для расчета стоимости разработки программы

	Программист	Постановщик задачи
1	2	3
Среднемесячная зарплата, руб.	200 000	70 000
Количество рабочих дней в месяце	20	20
Продолжительность рабочего дня, час	8	8
Стоимость часа машинного времени, руб./час	0,5	0,5

##### 3.1.1 Основные расчетные формулы

##### 3.1.1.1 Расчет средней стоимости одного часа работы

$$ЗЧ = (СРМЗП) / (КРДМ * ПРД), \text{руб.}, \quad (1)$$

где

ЗЧ – средняя стоимость часа работы, руб.

СРМЗП – среднемесячная зарплата, руб.

КРДМ – количество рабочих дней в месяце, дней

ПРД – продолжительность рабочего дня, час.

Данные из таблицы 3.2.

ДП 09.02.03 86.09.24ПЗ

Лист

39

Инв. № подл. Подп. и дата Взам. Инв. № Инв. № дубл. Подп. и дата

Изм. Лист № докум. Подп. Дата

$ЗЧ(\text{програм.}) = (200\ 000) / (20 \cdot 8) = 1\ 250$  руб.

$ЗЧ(\text{пост.}) = (70\ 000) / (20 \cdot 8) = 437,5$  руб.

### 3.1.1.2 Расчет расходов на оплату труда по программе

$$ЗП = ЗЧ \cdot Т_{\text{прог}}, \text{руб.}, \quad (2)$$

где

ЗП – расходы на оплату труда по программе, руб.

ЗЧ – средняя стоимость часа работника, руб.

Т<sub>прог</sub> – время подготовки программы, час.

Данные из таблицы 3.1 и таблицы 3.2.

$ЗП(\text{програм.}) = 1250 \cdot 240 = 300\ 000$  руб.

$ЗП(\text{пост.}) = 437,5 \cdot 240 = 27\ 125$  руб.

### 3.2.3 Расчет отчислений на социальные нужды

$$СН = ЗП \cdot (ПСН / 100), \text{руб.}, \quad (3)$$

где

СН – отчисления на социальные нужды, руб.

ЗП – расходы на оплату труда по программе, руб.

ПСН – процент отчислений на социальные нужды, %.

$СН(\text{програм.}) = 300\ 000 \cdot (30/100) = 90\ 000$  руб.

$СН(\text{пост.}) = 27\ 125 \cdot (30/100) = 8\ 137,5$  руб.

### 3.2.4 Расчет накладных расходов

$$НР = ЗП \cdot (ПНР / 100), \text{руб.}, \quad (4)$$

Где

НР – накладные расходы, руб.

ЗП – расходы на оплату труда по программе, руб.

ПНР – процент накладных расходов, %.

$НР(\text{програм.}) = 300\ 000 \cdot (30/100) = 90\ 000$  руб.

$НР(\text{пост.}) = 27\ 125 \cdot (30/100) = 8\ 137,5$  руб.

### 3.1.1.3 Расчет оплаты машинного времени

$$МВ = Т_{\text{прм}} \cdot СМЧ, \text{руб.}, \quad (5)$$

где

МВ – оплата машинного времени, руб.

Т<sub>прм</sub> – время отладки программы, час.

СМЧ – стоимость часа машинного времени, руб/час.

Данные из таблицы 3.1 и таблицы 3.2.

$МВ(\text{програм.}) = 128 \cdot 0,5 = 64$  руб.

$МВ(\text{пост.}) = 33 \cdot 0,5 = 16,5$  руб.

### 3.1.1.4 Расчет стоимости программного продукта

$$СП = ЗП + СН + НР + МВ, \text{руб.}, \quad (6)$$

где

СП – стоимость программного продукта, руб.

ЗП – расходы на оплату труда, руб.

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	Подп. и дата	Изм.	Лист	№ докум.	Подп.	Дата	ДП 09.02.03 86.09.24ПЗ	Лист
												40



СН – отчисления на социальные нужды, руб.

НР – накладные расход, руб.

МВ – стоимость часа машинного времени, руб.

СП(програм.)=300 000+90 000+90 000+64=480 064 руб.

СП(пост.)=27 125+8 137,5+8 137,5+16,5=43 316,5 руб.

### 3.1.2 Расчет стоимости программного продукта

Таблица 3.3 – Расчет стоимости программного продукта

	Программист	Постановщик задачи
1	2	3
Средняя стоимость 1 часа работы, руб.	1 250	437,5
Расходы на оплату труда по программе, руб.	300 000	27 125
Отчисление на социальные нужды, руб.	90 000	8 137,5
Накладные расходы, руб.	102 000	9 222,5
Оплата машинного времени, руб.	64	16,5

Суммарная стоимость программного продукта - 523 380,5 руб.

### 3.1.3 Определение эффективности программного продукта

#### 3.1.3.1 Определение годовых эксплуатационных затрат

$$\text{ЭЗ} = 31 + 32 + 33 + 34, \text{ руб.}, \quad (7)$$

где

ЭЗ – общие эксплуатационные затраты, руб.;

31 – годовой фонд заработной платы персонала, руб.;

32 – годовые амортизационные отчисления, руб.;

33 – годовые затраты на электроэнергию, руб.;

34 – прочие затраты (картриджи, диски, бумага и т.п.), руб.;

Расчеты приведены в таблице 3.4.

Таблица 3.4 – Расчет времени на обработку информации ручным и автоматизированным способами

Вид работы	Ручная обработка (день)	Ручная обработка (месяц)	Машинная обработка (день)	Машинная обработка (месяц)
1	2	3	4	5
Количество операций	27	540	1000	80 000
Время на обработку одного документа, ч	1,5		0,003	
Время на обработку всех документов, ч	40,5	810	33	666
Итого часов		810		666

ДП 09.02.03 86.09.24ПЗ

Лист

41

Таблица 3.5 – Расчет годовых эксплуатационных затрат

1	2
Средняя месячная з\п работника, руб.	100 000
Стоимость ЭВМ с ПО	80 000
Коэффициент амортизации, в %	19
Мощность ЭВМ, кВтч	0,2
Стоимость кВтч	5
Прочие расходы	10 000

### 3.1.3.2 Вычисление годовых эксплуатационных затрат

1) годовой фонд заработной платы персонала, который занимается обслуживанием ЭВМ:

$$31 = q * Cз / 26 / t * (k + 1), \text{ руб.}, \quad (8)$$

где

q – время работы с системой в месяц, час;

t – продолжительность рабочего дня, час;

Cз – средняя месячная заработная плата работника

$$31 = 666 * 100\,000 / 26 / 8 * (0,054 + 1) = 337\,482,69 \text{ руб.}$$

2) годовые амортизационные отчисления:

$$32 = CЭВМ * K, \text{ руб.}, \quad (9)$$

где

CЭВМ – стоимость ЭВМ с установленным программным обеспечением, руб.

K – нормативный коэффициент амортизации - 19%

$$32 = 80\,000 * 19 / 100 = 15\,200 \text{ руб.}$$

3) годовые затраты на электроэнергию:

$$33 = KЧЭВМ * МЭВМ * ЦКВТ, \text{ руб.}, \quad (10)$$

где

33 – затраты на электроэнергию за год, руб.;

KЧЭВМ – количество часов работы ЭВМ в год, руб.;

МЭВМ – мощность ЭВМ, кВтч.;

ЦКВТ – стоимость одного кВтч, руб.

$$33 = 7992 * 0,2 * 5 = 7992 \text{ руб.}$$

4) прочие затраты:

$$34 = CЭВМ * 0,1, \text{ руб.}, \quad (11)$$

где

34 – прочие затраты, руб.;

CЭВМ – Стоимость ЭВМ с установленным программным обеспечением, руб.

$$34 = 80000 * 0,1 \text{ руб.} = 8000, \text{ руб.}$$

5) годовые эксплуатационные затраты:

$$ЭЗ = 337\,482,69 + 15\,200 + 7992 + 8000 = 368\,674,69 \text{ руб.}$$

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

### 3.1.3.3 Расчет ожидаемого годового экономического эффекта

$$\Xi = \Xi_1 + \Xi_2 - \Xi_3, \text{ руб.}, \quad (12)$$

где

$\Xi$  – общая экономия;

$\Xi_1$  – экономия от снижения стоимости обработки информации;

$\Xi_2$  – экономия от увеличения производительности труда;

$\Xi_3$  – годовые эксплуатационные затраты, руб.

$$\Xi_1 = (PO - AO) * Cp, \text{ руб.}, \quad (13)$$

где

АО – время на автоматическую обработку информации за год = 3744 час;

РО – время на ручную обработку информации за год = 6240 час;

Ср – среднечасовая ставка работника = 144,23 руб./час.

$$\Xi_1 = (9720 - 7992) * 1250 = 2\,160\,000 \text{ руб.}$$

$$\Xi_2 = ОСРО - ОСАО, \text{ руб.}, \quad (14)$$

где

$\Xi_2$  – экономия от увеличения производительности труда, руб.;

ОСРО – общая стоимость ожидания запросов за год при ручной обработке, руб.;

ОСАО – общая стоимость ожидания запросов за год при автоматизированной, руб.

$$\Xi_2 = 150\,000 - 86,4 = 594\,000 \text{ руб.}$$

Расчеты приведены в таблице 3.6.

Таблица 3.6 – Расчет ожидаемого экономического эффекта

	Ручная обработка	Автоматизированная обработка
1	2	3
Среднее количество запросов в месяц	540	20 000
Среднечасовая з\п работника, руб.	1250	45
Среднее время ожидания на запрос, час.	0,5	0,08
Среднее время ожидания в месяц, час.	10	0,16
Стоимость времени ожидания в месяц, руб.	12 500	7,2
Стоимость времени ожидания в год, руб.	150 000	86,4

Экономия от увеличения производительности труда составляет 149 913,6 руб.

Общая экономия с учетом эксплуатационных расходов:

$$\Xi = 2\,160\,000 + 149\,913,6 - 368\,674,69 = 1\,941\,238,91 \text{ руб.}$$

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.
Подп. и дата	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата
------	------	----------	-------	------

ДП 09.02.03 86.09.24ПЗ

Лист

43

3.1.3.4. Срок окупаемости капиталовложений

$$T = L / Э, \text{ год}, \tag{15}$$

где

T – срок окупаемости капиталовложений, год;

L – дополнительные капиталовложения (стоимость программного продукта);

Э – общая экономия от использования программного продукта.

$$T = 523\,380,5 / 1\,941\,238,91 = 0,27 \text{ год.}$$

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

Лист
44

## 4 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

### 4.1. Общие требования к охране труда

1.1. К работе программистами допускаются лица старше 18 лет, имеющие соответствующую квалификацию для выполняемой работы и прошедшие вводный и основной инструктажи по технике безопасности на рабочем месте, медицинские осмотры, обучение и проверку знаний по охране труда.

1.2. Для выполнения работ на персональном компьютере программист должен изучить руководство по эксплуатации персонального компьютера, на котором будет работать сотрудник, пройти инструктаж по электробезопасности и получить группу I.

1.3. Независимо от квалификации или опыта работы, программисты, работающие на компьютере, должны проходить повторный инструктаж по охране труда не реже 6 раз в 1 месяц. В случае нарушения требований охраны труда и перерыва в работе более чем на 60 календарных дней программист обязан пройти внеплановый инструктаж.

1.4. Программисты, проявившие недостаточную квалификацию и знание требований безопасности при работе на персональном компьютере, не могут работать самостоятельно.

1.5. Программисты, допущенные к самостоятельной работе, должны знать: правила эксплуатации и требования безопасности при работе с персональными компьютерами, методы рациональной организации рабочего места, санитарно-гигиенические требования к условиям труда, опасные и вредные производственные факторы, которые могут негативно повлиять на программиста.

1.6. Программист, направленный для участия в необычной для его профессии работе, должен получить целенаправленный инструктаж по безопасному выполнению предстоящей работы.

1.7. Во время работы на программиста могут оказывать негативное воздействие, в основном, следующие опасные и вредные производственные факторы:

- \* Перенапряжение зрительного анализатора при работе за экраном дисплея;
- \* Длительное статическое напряжение мышц спины, шеи, рук и ног, которое может привести к статической перегрузке программатора, ;
- \* Повышенный уровень шума;
- \* Ионизирующее и неионизирующее излучение, источником которого является видеодисплейный терминал;
- Статическое электричество;
- \* Ток, по пути прохождения которого через тело человека в случае короткого замыкания может попасть в корпус.

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

1.8. Программист, работающий на персональном компьютере, должен соблюдать установленный для него режим труда и отдыха.

1.9. Для предотвращения возможных пожаров программисты должны самостоятельно соблюдать требования пожарной безопасности и не допускать нарушений со стороны других сотрудников.

1.10. Чтобы предотвратить заболевание, программист должен знать и соблюдать правила личной гигиены.

1.11. В случае болезни, плохого самочувствия или недостатка отдыха программист должен сообщить о своем состоянии своему непосредственному руководителю и обратиться за медицинской помощью.

1.12. Если программист был свидетелем несчастного случая, он должен оказать первую медицинскую помощь пострадавшему и сообщить о случившемся руководителю.

1.13. Программист должен уметь оказывать первую медицинскую помощь и пользоваться аптечкой первой помощи, в том числе в случае поражения электрическим током.

1.14. Любой программист, нарушивший требования Инструкции по охране труда, будет нести ответственность в соответствии с действующим законодательством.

#### **4.2 Требования по охране труда перед началом работы**

2.1. Перед началом работы программист должен разумно организовать рабочее место.

2.2. Программисты должны иметь в виду, что при наличии в помещении нескольких персональных компьютеров расстояние между ними должно составлять не менее 1,5 м для обеспечения безопасности.

2.3. Программисты должны иметь в виду, что взаимное расположение персональных компьютеров влияет на уровень производимого ими излучения.:

2.3.1. Левая панель персонального компьютера должна быть обращена либо к стене, либо к проходу, где нет рабочего места.

2.3.2. Не устанавливайте мониторы экранами друг к другу.

2.4. Не рекомендуется размещать экран монитора в витрине.

2.5. Чтобы избежать перенапряжения зрительного анализатора во время работы, программист должен следить за тем, чтобы на клавиатуре и экране монитора не было бликов.

2.6. Для повышения контрастности изображения перед началом работы программист должен очистить экран монитора от пыли, которая будет интенсивно оседать под воздействием статического электричества.

2.7. Программист должен убрать с рабочего места все ненужные предметы, которые не используются в работе.

2.8. Перед включением персонального компьютера программист должен визуально проверить исправность электропроводки, вилок, розеток и

Инв. № подл.	Взам. Инв. №	Инв. № дубл.	Подп. и дата					
Инв. № подл.	Взам. Инв. №	Инв. № дубл.	Подп. и дата					
Изм.	Лист	№ докум.	Подп.	Дата	ДП 09.02.03 86.09.24ПЗ			Лист
								46



3.12. Клавиатура должна располагаться на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной рабочей поверхности с регулируемой высотой, отделенной от столешницы основного стола.

3.13. Чтобы уменьшить искажение зрения, программист должен установить оптимальный цветовой режим на экране монитора (если это возможно). Рекомендуются ненасыщенные цвета: светло-зеленый, желто-зеленый, желто-оранжевый, коричневый. По возможности, программист должен избегать насыщенных цветов, особенно красного, синего и ярко-зеленого.

3.14. Для снижения зрительного утомления программисту предпочтительнее работать в таком режиме, чтобы на ярком экране видеомонитора был темный символ.

3.15. Чтобы снизить утомление зрительной и опорно-двигательной систем, программистам необходимо соблюдать установленный режим труда и отдыха.

3.16. Режим труда и отдыха при работе с персональным компьютером должен быть организован в соответствии с видом и категорией трудовой деятельности.

3.17. Виды трудовой деятельности разделены на 3 группы:

\* Группа А - Работа над предварительными запросами и считывание информации с экрана видеомонитора;

\* Группа В - Задание по вводу информации;

\* Группа В - Творческая работа в режиме взаимодействия с персональным компьютером.

3.18. При выполнении работ, связанных с различными видами работ в течение рабочей смены, основная работа с использованием компьютера должна занимать не менее 50% времени в течение рабочей смены или в течение рабочего дня.

3.19. Продолжительность непрерывной работы с видеомонитором без регламентированного перерыва не должна превышать 2 часов.

3.20. Для обеспечения оптимальной производительности и поддержания здоровья программиста необходимо установить регламентированный перерыв в течение рабочей смены.

3.21. Время регламентированного перерыва в течение рабочей смены должно устанавливаться в зависимости от его продолжительности, вида и категории трудовой деятельности.

3.22. При работе за компьютером в ночную смену (с 22 до 6 часов) продолжительность регламентированного перерыва должна быть увеличена на 60 минут, независимо от категории и вида трудовой деятельности.

3.23. Чтобы снять зрительное и посттеническое напряжение, программист должен в процессе работы принять микропаузу на 1-3 минуты.

3.24. Во время регламентированного перерыва программистам

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	ДП 09.02.03 86.09.24ПЗ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		48





микроорганизмами, которые находятся на поврежденном предмете, коже пострадавшего и в пыли, на руках работника, оказывающего помощь, и на грязной повязке материал.

4.6. При возникновении пожара или признаков пожара (дым, запах гари, повышение температуры и т.д.). При обнаружении пожарной тревоги необходимо немедленно сообщить в пожарную службу по телефону 01.

4.7. До прибытия пожарного подразделения необходимо принять меры по эвакуации людей и имущества и приступить к тушению пожара.

4.8. В случае возникновения пожара на персональном компьютере программист должен отключить его от источника тока и самостоятельно приступить к тушению. Следует помнить, что для тушения установок, находящихся под напряжением, используются углекислотные или порошковые огнетушители.

#### 4.5 Требования охраны труда по окончании работы

5.1. По окончании работ программист обязан соблюдать следующую последовательность выключения техники: • произвести закрытие всех активных задач; • выключить питание системного блока (процессора); • выключить питание всех периферийных устройств; • привести в порядок свое рабочее место.

5.2. После окончания работ убрать рабочее место и привести в порядок используемое в работе оборудование.

5.4. По окончании работ работник должен вымыть руки теплой водой с мылом.

5.5. Об окончании работы и всех недостатках, обнаруженных во время работы, известить своего непосредственного руководителя.

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.
Подп. и дата	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата
------	------	----------	-------	------

ДП 09.02.03 86.09.24ПЗ

## ЗАКЛЮЧЕНИЕ

Введение информационной системы обратной связи и учёта задач для работников АО УПП «Вектор» является важным шагом в направлении оптимизации процессов выполнения задач и улучшения взаимодействия между работниками. Веб-приложение, разработанное в рамках проекта, обеспечивает безопасность и контроль над доступом к информации, а также предоставляет функции для отображения и изменения статуса задач и обращений.

В результате проекта произошло ожидаемое улучшение эффективности выполнения задач и обращений, а также повышение уровня безопасности и контроля над доступом к информации.

В ходе выполнения проекта были достигнуты следующие результаты:

- разработано веб-приложение, обеспечивающее безопасность и контроль над доступом к информации;
- созданы функции для отображения и изменения статуса задач и обращений;
- обеспечена возможность авторизации и аутентификации для работников и администраторов;
- разработана система шифрования текстовых данных для обеспечения безопасности передачи информации;
- создана функция отчёта по задачам и обращениям для удобного анализа и планирования;
- разработана система сохранения действий администраторов в отдельном файле для отслеживания изменений.

В целом, разработанное веб-приложение является важным инструментом для оптимизации процессов выполнения задач и улучшения взаимодействия между работниками АО УПП «Вектор». Оно обеспечивает безопасность и контроль над доступом к информации, а также предоставляет функции для отображения и изменения статуса задач и обращений.

В будущем, разработанное веб-приложение может быть расширено и улучшено, добавляя новые функции и возможности для работников и администраторов АО УПП «Вектор». Это позволит еще более эффективно управлять задачами и обращениями, а также обеспечивать безопасность и контроль над доступом к информации.

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

В заключение, разработанный проект информационной системы обратной связи и учёта задач для работников АО УПП «Вектор» является важным шагом в направлении оптимизации процессов выполнения задач и улучшения взаимодействия между работниками. Веб-приложение, разработанное в рамках проекта, обеспечивает безопасность и контроль над доступом к информации, а также предоставляет функции для отображения и изменения статуса задач и обращений.

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

Лист
52

## Список используемых источников

1. Мартин Р. Чистый код. Создание, анализ и рефакторинг. Изд-во Addison-Wesley, 2018. - 42 с.
2. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. Изд-во Addison-Wesley, 2018. - 12 с.
3. Макконнелл С. Совершенный код. 2-е издание. Изд-во Addison-Wesley, 2018. - 154 с.
4. Фаулер М. Рефакторинг. Улучшение существующего кода. Изд-во Addison-Wesley, 2018. - 58 с.
5. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. Изд-во Addison-Wesley, 2018. - 63 с.
6. Мовеакс А. ASP.NET Core. Аутентификация в API. Часть 3: Реализация с JWE. 2019. URL: <https://habr.com/ru/company/it-knowledge/blog/444444/> (дата обращения: 12.04.2024).
7. Керниган Б. Практика программирования на языке C#. Изд-во Addison-Wesley, 2018. - 168 с.
8. Фаулер М. Архитектура корпоративных программных приложений. Изд-во Addison-Wesley, 2018. - 20 с.
9. Фримен А. Entity Framework Core 2 для ASP.NET Core MVC для профессионалов. 2019. URL: <https://www.packtpub.com/product/entity-framework-core-2-for-aspnet-core-mvc-for-professionals/9781789954444> (дата обращения: 12.03.2024).

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата	Wesley, 2018. - 168 с.
					8. Фаулер М. Архитектура корпоративных программных приложений. Изд-во Addison-Wesley, 2018. - 20 с.
					9. Фримен А. Entity Framework Core 2 для ASP.NET Core MVC для профессионалов. 2019. URL: <a href="https://www.packtpub.com/product/entity-framework-core-2-for-aspnet-core-mvc-for-professionals/9781789954444">https://www.packtpub.com/product/entity-framework-core-2-for-aspnet-core-mvc-for-professionals/9781789954444</a> (дата обращения: 12.03.2024).

## СПИСОК СОКРАЩЕНИЙ

ПК – персональный компьютер;  
 CPU – центральный процессор;  
 RAM – оперативная память;  
 ALU – арифметико-логическое устройство;  
 CU – устройство управления;  
 SRAM – статистическое оперативное запоминающее устройство;  
 DRAM – динамическое оперативное запоминающее устройство;  
 ROM – постоянное запоминающее устройство;  
 SSD – твердотельный накопитель;  
 HDD – жесткий диск;  
 ОЗУ – оперативное запоминающее устройство;  
 DIMM – слот для модуля оперативной памяти;  
 SATA – разъем для подключения накопителя данных;  
 ПО – программное обеспечение;  
 IDE – интегрированная система разработки;  
 СУБД – система управления базами данных;  
 ОС – операционная система;  
 HTML – язык гипертекстовой разметки;  
 CSS – каскадные таблицы стилей;  
 DOM – это стандартная объектная модель;  
 ORM – объектно-реляционное отображение;  
 SQL – язык структурированных запросов.

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

# ПРИЛОЖЕНИЕ А

## ТЕКСТ ПРОГРАММЫ

Модуль «UserService.Create»

```
public async Task<IBaseResponse<UserEntity>>
Create(CreateUserViewModel model)
{
    try
    {
        var user = new UserEntity();
        _logger.LogInformation($"Запрос на создание
пользователя - {model.NachalnikId}");

        var users = _userRepository.GetAll()
            .AsNoTracking()
            .ToList();

        //ПРОВЕРИТЬ НА СОЗДАНИЕ АНАЛОГИЧНЫХ ЛОГИНОВ
        var existingUser = users.FirstOrDefault(u =>
Decrypt(u.EnLogin, u.EncryptionKey, u.InitializationVector) ==
model.login);

        if (existingUser != null)
        {
            return new BaseResponse<UserEntity>()
            {
                Description = "Пользователь с таким
логинном уже существует",
                StatusCode = StatusCode.TaskIsHasAlready
            };
        }

        //Шифрование
        byte[] key = new byte[16];
        byte[] iv = new byte[16];
        using (RandomNumberGenerator rng =
RandomNumberGenerator.Create())
        {
            rng.GetBytes(key);
            rng.GetBytes(iv);
        }

        if (model.NachalnikId.Equals(null))
        {
            user = new UserEntity()
            {
                EnFIO = Encrypt(model.FIO, key, iv),
                EnLogin = Encrypt(model.login, key, iv),
                EnPass = Encrypt(model.password, key, iv),
                role = model.role,
                DepartamentId = model.DepartamentId,
                EncryptionKey = key,
                InitializationVector = iv
            }
        }
    }
}
```

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

Лист  
55

## Продолжение Приложения А

```

        };
    }
    else
    {
        user = new UserEntity()
        {
            EnFIO = Encrypt(model.FIO, key, iv),
            EnLogin = Encrypt(model.login, key, iv),
            EnPass = Encrypt(model.password, key, iv),
            role = model.role,
            DepartamentId = model.DepartamentId,
            NachalnikId = model.NachalnikId,
            EncryptionKey = key,
            InitializationVector = iv
        };
    }

    await _userRepository.Create(user);
    //
    _logger.LogInformation(model.logID.ToString());
    // var logFIO =
    GetById(System.Convert.ToUInt32(model.logID));
    using (StreamWriter writer = new
    StreamWriter("logs\\app.log", true))
    {
        writer.WriteLine($"{DateTime.Now} | Создание
пользователя {model.FIO}");
    }
    //_logger.LogInformation($"Пользователь создан:
{user.FIO} ");
    return new BaseResponse<UserEntity>()
    {
        Description = "Пользователь создан",
        StatusCode = StatusCode.OK
    };
}
catch (Exception ex)
{
    _logger.LogError(ex, $"[UserService.Create]:
{ex.Message}");
    return new BaseResponse<UserEntity>()
    {
        Description = "Пользователь не создан",
        StatusCode = StatusCode.InternalServerError
    };
}

}

public async Task<DataTableResult> GetAll(UserFilter
filter)
{

```

Инв. № подл.	Подп. и дата
Взам. Инв. №	Инв. № дубл.
Подп. и дата	

Изм.	Лист	№ докум.	Подп.	Дата
------	------	----------	-------	------

ДП 09.02.03 86.09.24ПЗ

Лист  
56



## Продолжение Приложения А

```

try
{
    var query = await
_userRepository.GetAll().ToListAsync();

    var decryptedUsers = new List<UserViewModel>();

    foreach (var currentUser in query)
    {
        string NachFIO = string.Empty;
        string Depart = string.Empty;
        string DepNum = string.Empty;

        if (currentUser.NachalnikId != null)
        {
            var Nach = await
_userRepository.GetAll().FirstOrDefaultAsync(x => x.Id ==
currentUser.NachalnikId);
            if (Nach != null)
            {
                NachFIO = Decrypt(Nach.EnFIO,
Nach.EncryptionKey, Nach.InitializationVector);
            }
        }
        if (currentUser.DepartmentId != null)
        {
            DepFilter fil = new DepFilter()
            {
                idDep =
currentUser.DepartmentId.ToString()
            };
            var departament = await
_depRepository.GetAll().FirstOrDefaultAsync(x=>x.DepartmentId==cu
rrentUser.DepartmentId);
            if (departament != null)
            {
                Depart =
Decrypt(departament.DepartmentName, departament.EncryptionKey,
departament.InitializationVector);
                DepNum
=departament.DepartmentNumber.ToString();
            }
        }

        var decryptedUser = new UserViewModel()
        {
            Id = currentUser.Id,
            FIO = Decrypt(currentUser.EnFIO,
currentUser.EncryptionKey, currentUser.InitializationVector),
            login = Decrypt(currentUser.EnLogin,
currentUser.EncryptionKey, currentUser.InitializationVector),
            password = Decrypt(currentUser.EnPass,
currentUser.EncryptionKey, currentUser.InitializationVector),
            role = currentUser.role.GetDisplayName(),

```

Подп. и дата	<pre>        {             idDep = currentUser.DepartmentId.ToString()         };         var department = await _depRepository.GetAll().FirstOrDefaultAsync(x=&gt;x.DepartmentId==cu rrentUser.DepartmentId);         if (department != null)         {             Depart = Decrypt(department.DepartmentName, department.EncryptionKey, department.InitializationVector);             DepNum =department.DepartmentNumber.ToString();         }          var decryptedUser = new UserViewModel()         {             Id = currentUser.Id,             FIO = Decrypt(currentUser.EnFIO, currentUser.EncryptionKey, currentUser.InitializationVector),             login = Decrypt(currentUser.EnLogin, currentUser.EncryptionKey, currentUser.InitializationVector),             password = Decrypt(currentUser.EnPass, currentUser.EncryptionKey, currentUser.InitializationVector),             role = currentUser.role.GetDisplayName(),</pre>					
Инв. № дудл.						
Взам. Инв. №						
Подп. и дата						
Инв. № подл.						
					ДП 09.02.03 86.09.24ПЗ	Лист
Изм	Лист	№ докум	Подп.	Дата		57

## Продолжение Приложения А

```

        DepartamentId = DepNum,
        NachalnikId = currentUser.NachalnikId != null
? currentUser.NachalnikId.ToString() : string.Empty,
        Nachalnik = NachFIO,
        Departament = Depart // Устанавливаем
DepartamentName
    };

    decryptedUsers.Add(decryptedUser);
}
if (!string.IsNullOrEmpty(filter.FIO))
{
    string filterFIO = filter.FIO.ToLower(); //
Приводим filter.FIO к нижнему регистру

    decryptedUsers = decryptedUsers.Where(x =>
x.FIO.ToLower().StartsWith(filterFIO)).ToList();
}
if (filter.role.HasValue)
{
    decryptedUsers = decryptedUsers.Where(x => x.role
== filter.role.Value.GetDisplayName()).ToList();
}
if (filter.departamentId != 0)
{
    decryptedUsers = decryptedUsers.Where(x =>
x.DepartamentId == filter.departamentId.ToString()).ToList();
}

if (filter.NachalnikId != 0)
{
    decryptedUsers = decryptedUsers.Where(x =>
x.NachalnikId == filter.NachalnikId.ToString()).ToList();
}
if
(!string.IsNullOrEmpty(filter.departamentName))
{
    string filterDepartamentName =
filter.departamentName.ToLower(); // Приводим
filter.departamentName к нижнему регистру

    decryptedUsers = decryptedUsers.Where(x =>
x.Departament.ToLower().StartsWith(filter.departamentName)).ToList
();
}
// Остальной код остается без изменений
// Сортировка
if (!string.IsNullOrEmpty(filter.SortColumn) &&
!string.IsNullOrEmpty(filter.SortDirection))
{
    decryptedUsers = ApplySorting(decryptedUsers,
filter.SortColumn, filter.SortDirection);
}
else

```

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	

Изм.	Лист	№ докум.	Подп.	Дата
------	------	----------	-------	------

ДП 09.02.03 86.09.24ПЗ

## Окончание Приложения А

```

    {
        decryptedUsers = decryptedUsers.OrderBy(x =>
x.Id).ToList(); // Сортировка по умолчанию по Id
    }
    var pagedUsers = decryptedUsers
        .Skip(filter.Skip)
        .Take(filter.PageSize)
        .ToList();

    var count = pagedUsers.Count;

    return new DataTableResult()
    {
        Data = pagedUsers,
        Total = count
    };
}
catch (Exception ex)
{
    _logger.LogError(ex, $"[UserService.GetAll]:
{ex.Message}");
    return new DataTableResult()
    {
        Data = null,
        Total = 0
    };
}
}

```

Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

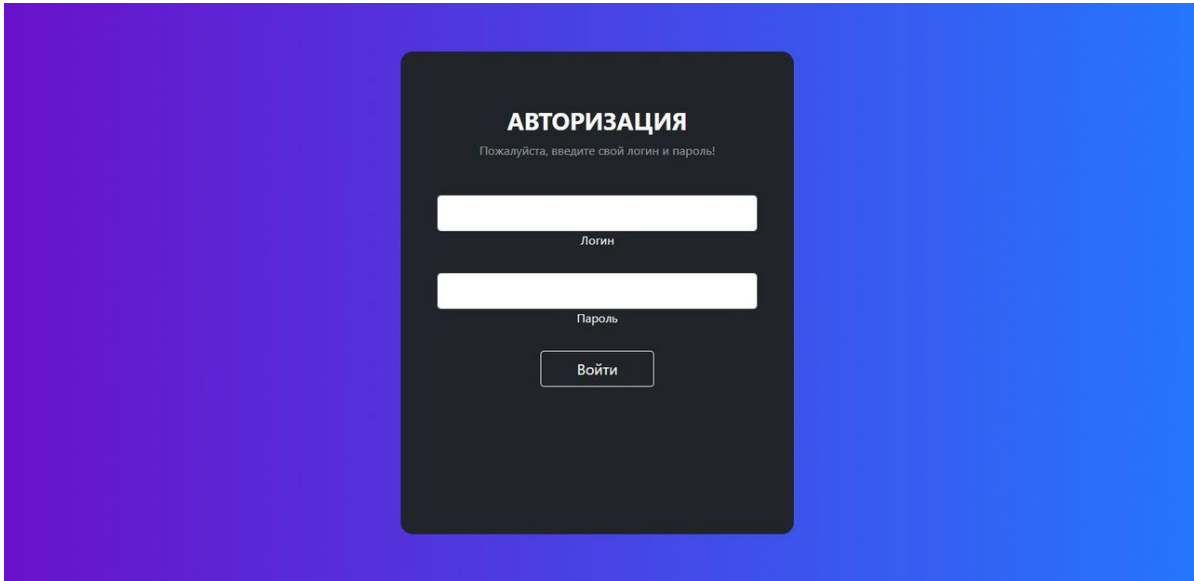
Лист

59

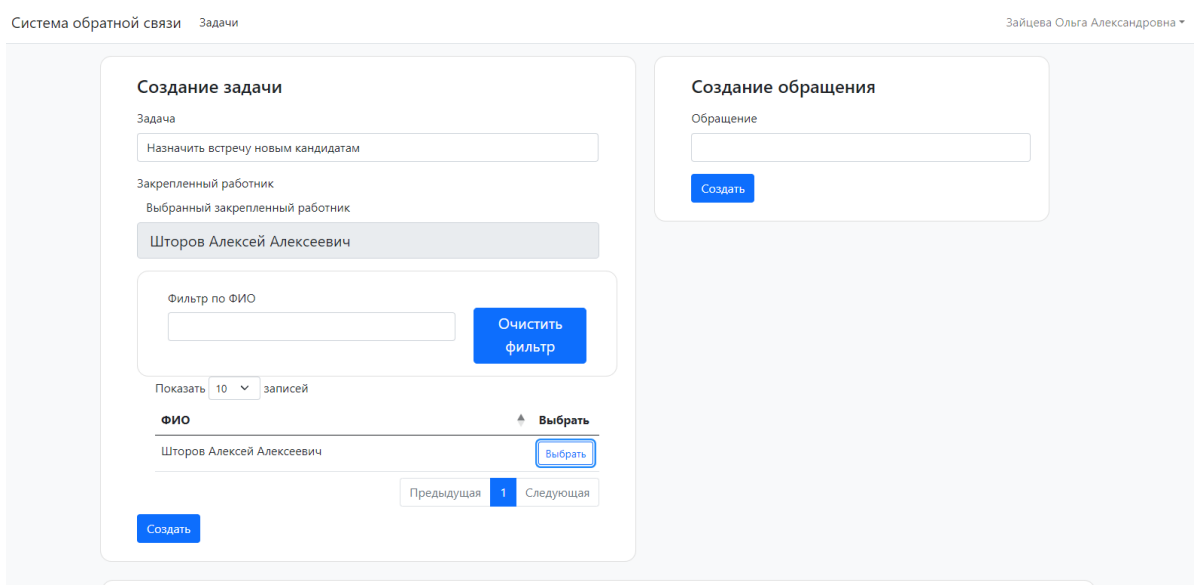
ПРИЛОЖЕНИЕ Б

РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

Страница «Авторизация».



Страница «Задачи».



Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ

Лист
60

Окончание Приложения Б  
Пример полученного отчёта.

Название	Статус	Создано	Завершено	Работник
Найти новых кандидатов	Выполнено	4 июня 2024 г.	4 июня 2024 г.	Шторов Алексей Алексеевич
Отсортировать новых кандидатов	В работе	4 июня 2024 г.		Шторов Алексей Алексеевич
Назначить встречу новым кандидатам	Не взято	4 июня 2024 г.		Шторов Алексей Алексеевич

Подпись: \_\_\_\_\_ Дата: \_\_\_\_\_

Страница «Отделы».

Система обратной связи

Задачи

Регистрация пользователя

Пользователи

Отделы

superAdmin

Фильтры

Фильтр по номеру

Фильтр по названию

Очистить фильтры

Регистрация отдела

Название

Зарегистрировать

Показать 10 записей

Номер	Название	Удалить
1	Кадровый	Удалить
2	Информационные технологии	Удалить
3	Ракетостроительный	Удалить

Предыдущая 1 Следующая

Страница «Пользователи».

Система обратной связи

Задачи

Регистрация пользователя

Пользователи

Отделы

superAdmin

Фильтры

Фильтр по ФИО

Фильтр по роли

Фильтр по номеру отдела

Фильтр по названию отдела

Очистить фильтры

Показать 10 записей

ФИО	Логин	Пароль	Роль	Номер отдела	Название отдела	Начальник	Удалить
Борисов	123	123	Администратор	1	Кадровый		Удалить
Зайцева Ольга Александровна	312	312	Заместитель начальника отдела	3	Ракетостроительный	Фурманов Олег Владимирович	Удалить
Коркин	666	666	Начальник отдела	1	Кадровый		Удалить
Фурманов Олег Владимирович	dds	ff	Заместитель начальника отдела	1	Кадровый		Удалить
Фурманов Олег Владимирович	321	321	Начальник отдела	3	Ракетостроительный		Удалить

Подп. и дата

Инв. № дубл.

Взам. Инв. №

Подп. и дата

Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата

ДП 09.02.03 86.09.24ПЗ