

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

Клиентское приложение электронной контактной книги с возможностью ведения заметок

Студент _____ Макаричева Елизавета Михайловна
(подпись, дата) (ФИО полностью)

Руководитель _____ Шибанов С.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«25» декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Макаричева Елизавета Михайловна гр. 20ВП1 факультета ФВТ
<Фамилия, имя, отчество полностью>
2. Тема работы Клиентское приложение электронной контактной книги с возможностью ведения заметок
- Тема утверждена приказом ПГУ № _____ от " ____ " _____ 20 _____
3. Руководитель работы _____ к.т.н., доцент кафедры «МО и ПЭВМ» Шибанов С. В.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Назначение разработки: Приложение электронной контактной книги с возможностью ведения заметок предназначено для структуризации и хранения информации пользователя, включая контактные данные и личные заметки.
- Средства выполнения: Операционная система Windows 10 Pro
- Средства разработки: Visual Studio Code
- Языковые средства разработки: JavaScript
- Технологии разработки: UML, функциональное программирование
5. Перечень подлежащих разработке вопросов
- 1) Анализ предметной области и постановка задачи на разработку приложения;
- 2) Планирование и оценка процесса разработки;
- 3) Разработка требований и проектирование;
- 4) Разработка клиентского приложения электронной контактной книги;
- 5) Контроль качества программного обеспечения клиентского приложения.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок вы- полнения	Подпись ру- ководителя
1	Анализ предметной области и требо- ваний к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 98 страниц, 30 рисунков, 21 таблица, 24 использованных источника.

ЗАМЕТКИ, КОНТАКТЫ, ТЕГИ, WEB-ПРИЛОЖЕНИЕ, ИНФОРМАЦИОННЫЙ ОБЪЕКТ, АНАЛОГИ, REST API, MPA, UML.

Объект исследования – клиентское приложение электронной контактной книги с возможностью ведения заметок.

Цели работы – разработать клиентское приложение электронной контактной книги.

Технологии разработки – разработка производилась с использованием HTML и CSS на языке программирования JavaScript. В качестве среды разработки использовался редактор программного кода Visual Studio Code.

Среда исполнения – Windows 10 Pro, web-браузер.

Результаты работы: спроектирован и реализован прототип клиентского приложения электронной контактной книги с возможностью ведения заметок в соответствии с требованиями к разработке приложения.

					ПГУ 09.03.04 – 08ПП201.17 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Макаричева Е.М			«Клиентское приложение электронной контактной книги с возможностью ведения заметок» Пояснительная записка	Лит.	Лист	Листов
Провер.		Шибанов С. В.					4	98
Реценз.						Группа 20ВП1		
Н. контр.		Такташкин Д.В.						
Утв.		Козлов А. Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и постановка задачи на разработку приложения. 9	9
1.1 Анализ предметной области приложения	9
1.1.1 Анализ структуры электронной контактной книги	9
1.1.2 Анализ свойств информационных объектов электронной контактной книги	10
1.1.3 Анализ процессов ведения электронной контактной книги.....	10
1.2 Сравнительный анализ аналогичных разработок	13
1.2.1 Выбор приложений-аналогов для анализа	13
1.2.2 Обзор приложений-аналогов для работы с заметками и контактами	14
1.2.3 Результаты сравнительного анализа	17
1.3 Постановка задачи на разработку приложения.....	19
1.3.1 Цели разработки и назначение приложения	19
1.3.2 Перечень автоматизируемых функций	20
1.4 Планирование и оценка процесса разработки приложения.....	21
1.4.1 Выбор нотации и программных средств для планирования и оценки процесса разработки приложения	21
1.4.2 Планирование процесса разработки приложения	22
1.4.3 Оценка временных и финансовых затрат на разработку	25
2 Проектирование приложения.....	28
2.1 Моделирование функциональных возможностей приложения	28
2.2 Клиент-серверная архитектура электронной контактной книги.....	31
2.3 Разработка прототипа пользовательского интерфейса клиентского приложения	33
2.4 Требования к аппаратно-программному обеспечению клиентского приложения	38
2.5 Обоснование выбора технологий и средств разработки приложе-ния	39
3 Разработка клиентского приложения.....	41

3.1 Разработка структуры клиентского приложения.....	41
3.2 Разработка пользовательского интерфейса клиентского приложения	43
3.3 Разработка средств взаимодействия с серверным приложением.....	49
4 Контроль качества программного обеспечения клиентского приложения	53
4.1 Планирование процесса контроля качества программного обеспечения ..	53
4.1.1 Обоснование выбора модели тестирования	53
4.1.2 Обоснование выбора метрик программного кода	55
4.2 Тестирование программных средств.....	56
4.2.1 Функциональное тестирование клиентского приложения	56
4.2.2 Тестирование адаптивности интерфейса.....	57
4.2.3 Оценка времени выполнения задач пользователем.....	58
4.3 Оценка качества программного кода на основе метрик	60
4.3.1 Метрика SLOC.....	60
4.3.2 Метрика Холстеда.....	61
Заключение	64
Список использованных источников	65
Приложение А (справочное) Глоссарий	68
Приложение Б (рекомендуемое) Код приложения	70
Приложение В (рекомендуемое) Результаты функционального тестирования .	85
Приложение Г (рекомендуемое) Результаты тестирования адаптивности интерфейса	92

Введение

В настоящее время, в век компьютерных технологий мы все имеем доступ к информации, она окружает нас повсюду. Человеческая память чисто физически не рассчитана на такие большие объемы информации, соответственно возникает необходимость в ее структурировании и хранении. В связи с этим возникает необходимость в использовании различных приложений, способных хранить и работать с данными: заметки, календари, блокноты, контактные книги и др.

Актуальность разработки приложения электронной контактной книги с возможностью ведения заметок заключается в том, что такое приложение избавит пользователя от необходимости использования нескольких приложений, так как вся необходимая информация будет храниться в одном месте. Это позволит сделать процесс поиска и хранения оптимальнее, а также сэкономит время.

Целью выпускной квалификационной работы является разработка клиентского приложения электронной контактной книги с возможностью заметок.

Для достижения поставленной цели необходимо:

- произвести анализ предметной области разрабатываемой системы;
- рассмотреть аналогичные приложения и провести их сравнительный анализ;
- провести планирование и расчет бюджета разрабатываемой системы;
- выявить требования к архитектуре, интерфейсу и программному обеспечению приложения;
- реализовать приложение в соответствии с поставленной задачей;
- провести тестирование разработанного приложения.

В результате выполнения работы было реализовано клиентское приложение, позволяющее работать с заметками и контактами, включая добавление,

удаление и редактирование, а также осуществлять поиск нужной информации и фильтрацию элементов по категориям.

На данный момент реализован прототип приложения, функциональные возможности которого будут расширены в дальнейшем.

1 Анализ предметной области и постановка задачи на разработку приложения

1.1 Анализ предметной области приложения

1.1.1 Анализ структуры электронной контактной книги

Разрабатываемая система должна предоставить пользователям удобный и эффективный способ хранения и систематизации контактной информации о людях, включая имена, номера телефонов, адреса электронной почты другие данные, а также ведения заметок и настройки уведомлений о событиях.

С целью эффективности и упрощения процесса поиска информации в системе, заметки и контакты должны быть разбиты на категории – теги, которые создаются самим пользователем.

Структура электронной контактной книги представлена информационными объектами, представленными на рисунке 1.



Рисунок 1 – Информационные объекты контактной книги

Контакты и заметки являются независимыми друг от друга информационными объектами. Теги необходимы для организации групп контактов или заметок на основе различных критериев (например, сложности выполнения, вида деятельности, срочности и др.). Напоминание – это уведомление о наступлении

события или мероприятия, которое должно произойти в определенное время. Напоминания прикрепляются к конкретным контактам или заметкам.

1.1.2 Анализ свойств информационных объектов электронной контактной книги

Каждый информационный объект разрабатываемого приложения имеет набор свойств, характеризующих его.

Основным свойством тега является название. Помимо этого, тег может иметь уникальный цвет, что позволит пользователю с легкостью однозначно выделить нужную категорию на фоне остальных.

Напоминания имеют названия, дату и время напоминания, а также частоту напоминания.

Контакт содержит информацию о конкретном человеке, включая ФИО, дату рождения, номер телефона, адрес проживания, место работы, адрес электронной почты, социальные сети и дополнительное поле с другой информацией. Также при указании даты рождения пользователю должна быть предоставлена возможность для установки напоминания.

Свойствами заметки являются название заметки и ее описание. Заметке можно назначить тег или установить напоминание, при этом название напоминания будет совпадать с названием заметки.

1.1.3 Анализ процессов ведения электронной контактной книги

Как было отмечено ранее, в электронной контактной книге существует два независимых друг от друга информационных объекта: Контакты и Заметки. Работа с данными объектами представляет собой последовательность операций, необходимых для реализации функциональных возможностей приложения.

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций, когда необходимо представить алгоритмы их выполнения. В контексте языка

UML деятельность (activity) представляет собой совокупность отдельных вычислений, выполняемых автоматом, приводящих к некоторому результату или действию (action). На диаграмме деятельности отображается логика и последовательность переходов от одной деятельности к другой, а внимание аналитика фокусируется на результатах. Результат деятельности может привести к изменению состояния системы или возвращению некоторого значения [1].

Диаграмма деятельности, описывающая сценарий выполнения действий с Заметками и Kontakтами в электронной контактной книге, представлена на рисунке 2.

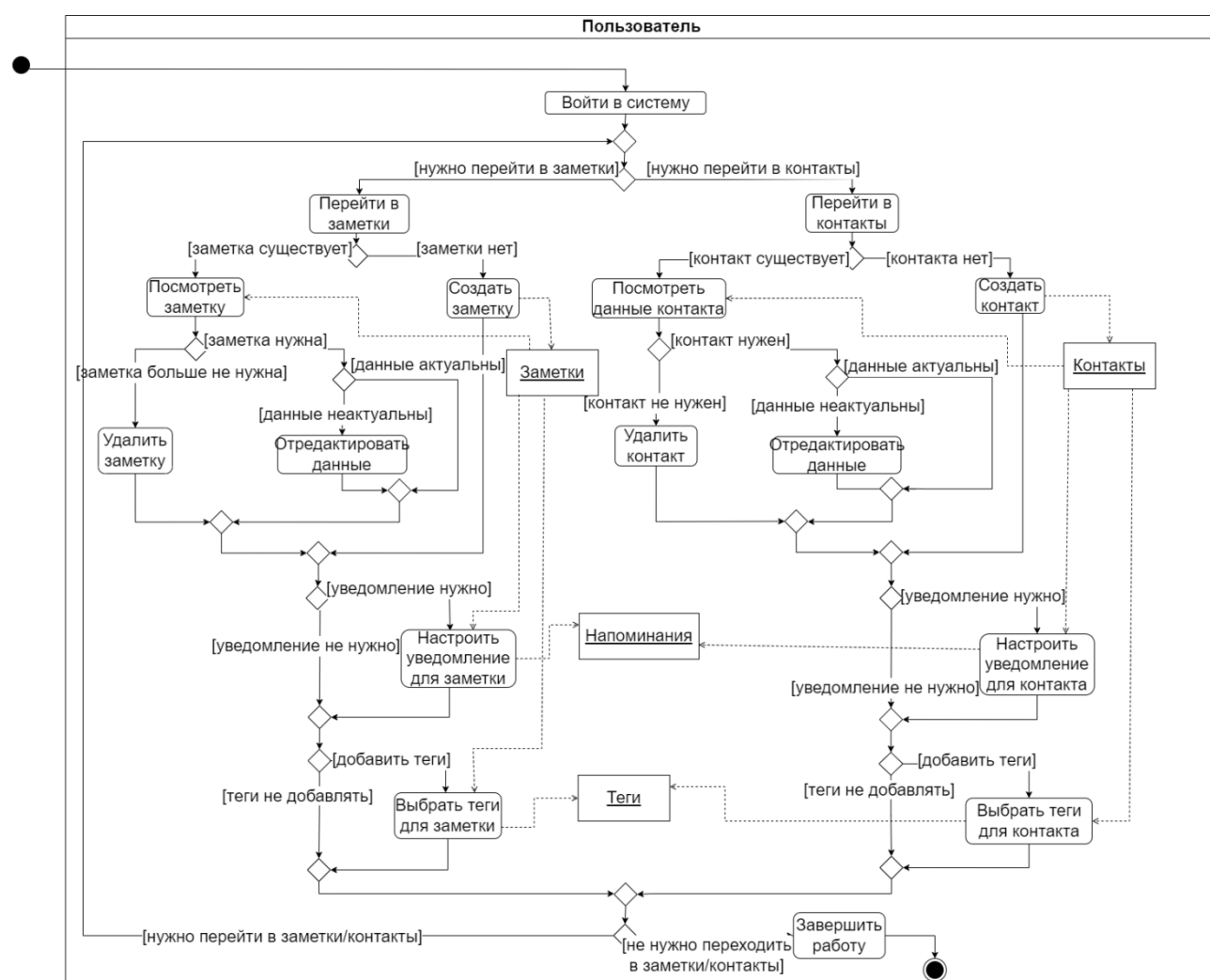


Рисунок 2 – Диаграмма деятельности для процессов ведения Контактов и Заметок

К процессам ведения и заметок, и контактов относятся работа с данными объекта, настройка уведомлений и выбор тегов.

Спецификации для процессов ведения контактов и заметок в электронной контактной книге приведены в таблице 1.

Таблица 1 – Спецификации процессов

№ п/п	Процесс	Используемые объекты	Входные данные	Выходные данные
1	Войти в систему		Логин и пароль	Доступ к функционалу системы и данным аккаунта
2	Перейти в заметки	Заметки	Переход на страницу с заметками	Список созданных заметок (при наличии)
3	Посмотреть заметку	Заметки	Переход на страницу с заметками	Отображение данных заметки
4	Создать заметку	Заметки	Переход на форму создания заметки	Отображение созданной заметки
5	Удалить заметку	Заметки	Исходная заметка	Заметка удалена
6	Отредактировать данные заметки	Заметки	Исходная заметка	Отображение заметки с измененными данными
7	Настроить уведомление для заметки	Заметки, Напоминания	Существующая заметка	Уведомления для заметки настроены
8	Выбрать теги для заметки	Заметки, Теги	Существующая заметка	Теги для заметки выбраны
9	Перейти в контакты	Контакты	Переход на страницу с контактами	Список созданных контактов (при наличии)
10	Посмотреть данные контакта	Контакты	Переход на страницу с контактами	Отображение данных контакта
11	Создать контакт	Контакты	Переход на форму создания контакта	Отображение созданного контакта
12	Удалить контакт	Контакты	Исходный контакт	Контакт удален
13	Отредактировать данные контакта	Контакты	Исходный контакт	Отображение контакта с измененными данными
14	Настроить уведомление для контакта	Контакты, Напоминания	Существующий контакт	Уведомления для контакта настроены
15	Выбрать теги для контакта	Контакты, Теги	Существующий контакт	Теги для контакта выбраны
16	Завершить работу		Переход на страницу личного кабинета	Страница входа в приложение

Процессы, выделенные для заметок и контактов, а также их спецификации применяются при проектировании и разработке клиентского приложения электронной контактной книги.

1.2 Сравнительный анализ аналогичных разработок

1.2.1 Выбор приложений-аналогов для анализа

Сравнительный анализ аналогичных разработок позволит рассмотреть уже существующие приложения и выделить наиболее актуальные из них.

Основной задачей анализа является выделение базовых функциональных возможностей рассматриваемых приложений. Помимо этого, необходимо обратить внимание на особенности каждого приложения для дальнейшей реализации или расширения возможностей разрабатываемой системы.

В настоящее время существует множество приложений, позволяющих вести заметки и работать с контактной информацией.

В связи с разработкой web-приложения для проведения сравнительного анализа были выбраны онлайн-сервисы:

- Google Keep;
- Яндекс Заметки;
- Simplenote;
- Google Контакты;
- ContactBook.

Среди представленных сервисов присутствуют как платные, так и бесплатные. Подробная информация об этих приложениях представлена в таблице 2.

Таблица 2 – Информация о приложениях-аналогах

Приложение	URL	Фирма-производитель	Доступность
Google Keep	https://www.google.ru/keep/	Google LLC	Бесплатно
Яндекс Заметки	https://disk.yandex.ru/notes/	Яндекс	Бесплатно
Simplenote	https://app.simplenote.com/	Automattic, Inc	Бесплатно
Google Контакты	https://contacts.google.com/	Google LLC	Бесплатно
ContactBook	https://www.contactbook.app/	Logicwind	Платно Доступна пробная версия на 14 дней

Таким образом, для обзора и оценки были выбраны приложения, ориентированные на работу с двумя типами информационных объектов, а именно с контактами и с заметками.

1.2.2 Обзор приложений-аналогов для работы с заметками и контактами

Для рассмотрения были выбраны следующие приложения для ведения заметок: Google Keep, Яндекс Заметки и Simplenote.

Google Keep – бесплатный сервис для создания заметок. Пользователь может создать заметки 3 типов: список, фотозаметку или заметку с рисунком.

Сервис позволяет создавать напоминания для выбранных заметок. Чтобы найти заметку, достаточно ввести запрос в строке поиска или выбрать нужную категорию – ярлык. Ярлыки создаются самим пользователем. Для удобства пользователей все заметки можно сортировать по цвету, содержанию или наличию совместного доступа [2]. Интерфейс данного сервиса представлен на рисунке 3.

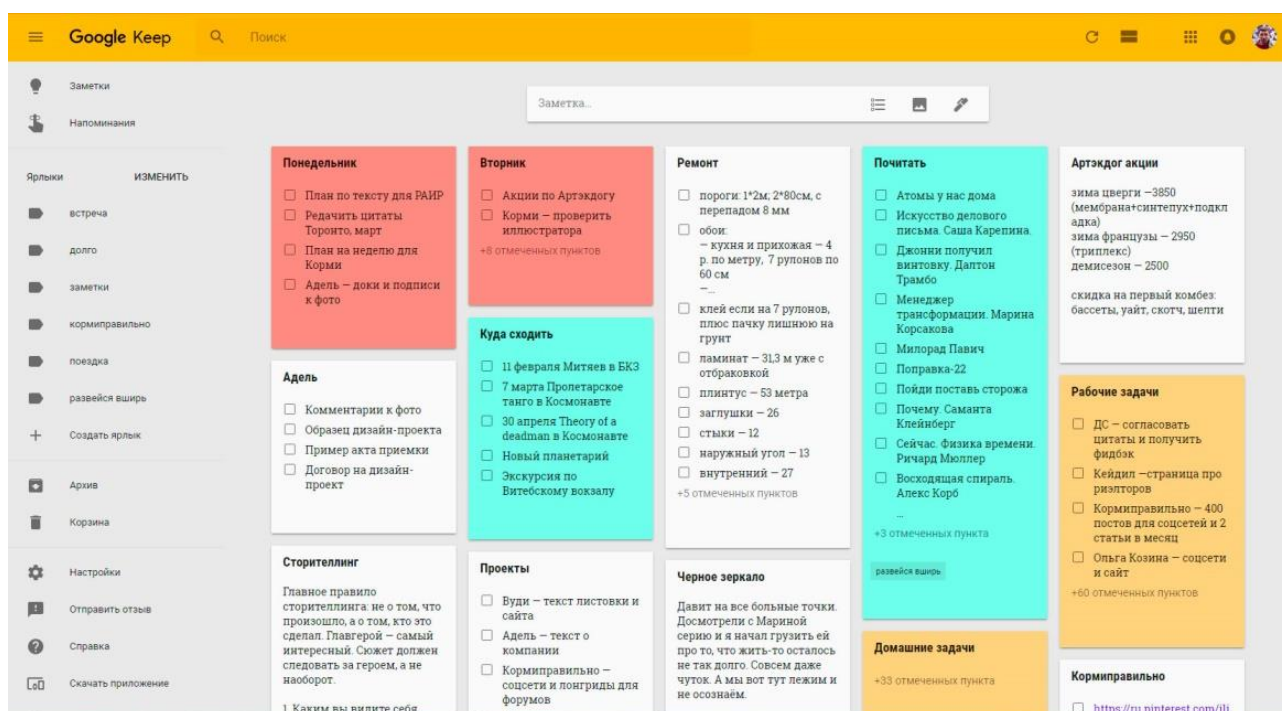


Рисунок 3 – Интерфейс Google Keep

Яндекс Заметки – сервис для заметок внутри Яндекс.Диска. Он позволяет записывать интересные идеи, конспектировать лекции, хранить рецепты и многое другое. Недавно отредактированные заметки перемещаются в списке наверх, старые спускаются вниз. Важные заметки можно закрепить, чтобы они всегда были на виду. В **Заметках** можно: создавать списки, выделять текст жирным, курсивом и подчёркивать, а также прикреплять фотографии (до 40 картинок в одной записи) [3]. На рисунке 4 приведен интерфейс этого сервиса.

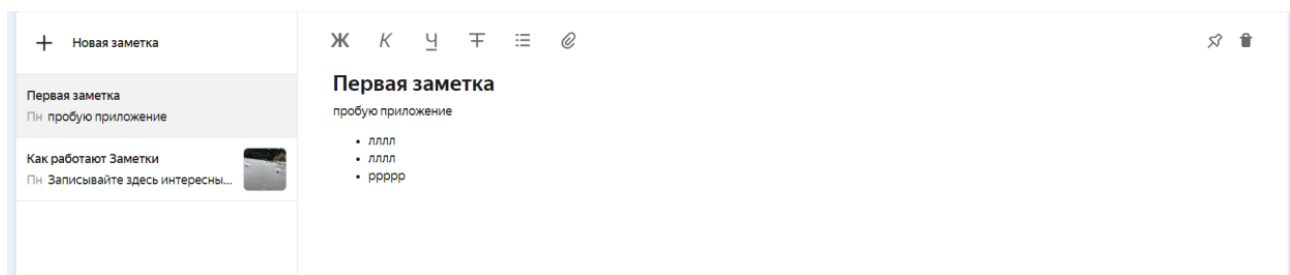


Рисунок 4 – Интерфейс Яндекс Заметок

Simplenote – простое приложение для ведения заметок, которое позволяет создавать, просматривать и публиковать свои заметки в формате Markdown. Заметкам можно добавлять теги, чтобы быстро находить нужные записи с помощью мгновенного поиска. При каждом изменении создаются резервные копии заметок, поэтому существует возможность просмотра того, что было отмечено на прошлой неделе или в прошлом месяце. Приложение совершенно бесплатное [4]. Интерфейс показан на рисунке 5.

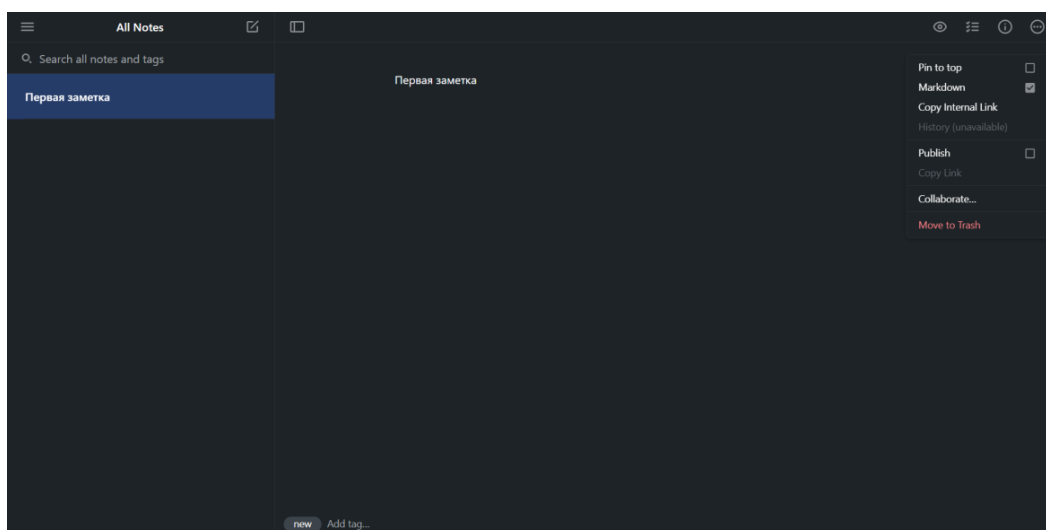


Рисунок 5 – Интерфейс Simplenote

Среди приложений для хранения контактных данных были выбраны Google Контакты и ContactBook.

Google Контакты – бесплатный онлайн-сервис для сбора контактов. Каждый контакт может содержать основную информацию, такую как имена, адреса электронной почты и номера телефонов, но может включать в себя и расширенные сведения, вроде физического адреса, места работы, отдел или должность [5]. Помимо поиска по имени контакта, сервис позволяет, как экспортировать, так и импортировать контакты в формате csv. Для удобства и эффективности работы с контактами пользователь может разбить их на группы.

Особенностью приложения является возможность печати данных выбранных контактов. Веб-интерфейс сервиса приведен на рисунке 6.

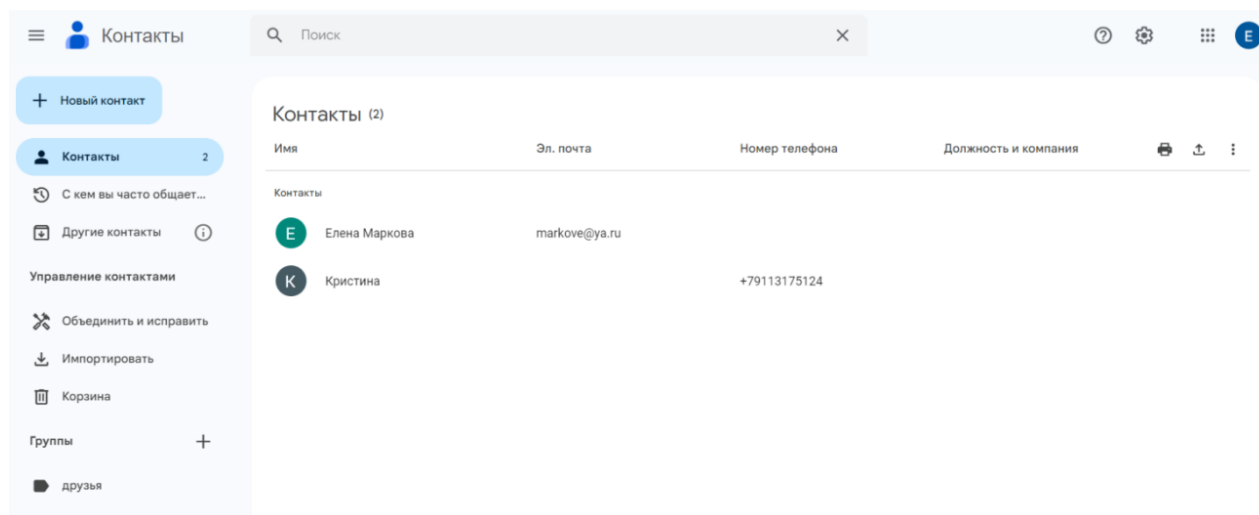


Рисунок 6 – Интерфейс Google Контакты

ContactBook – платное веб-приложение для управления контактами с возможностью импорта существующих контактов из разных источников, таких как Gmail или LinkedIn. Kontakтами можно делиться с другими пользователями с помощью общего доступа. Для классификации контактов используются пользовательские теги, что упрощает процесс поиска и фильтрации нужной информации. Помимо этого к контактам можно добавлять заметки, вложения и напоминания [6].

На рисунке 7 представлен интерфейс приложения ContactBook.

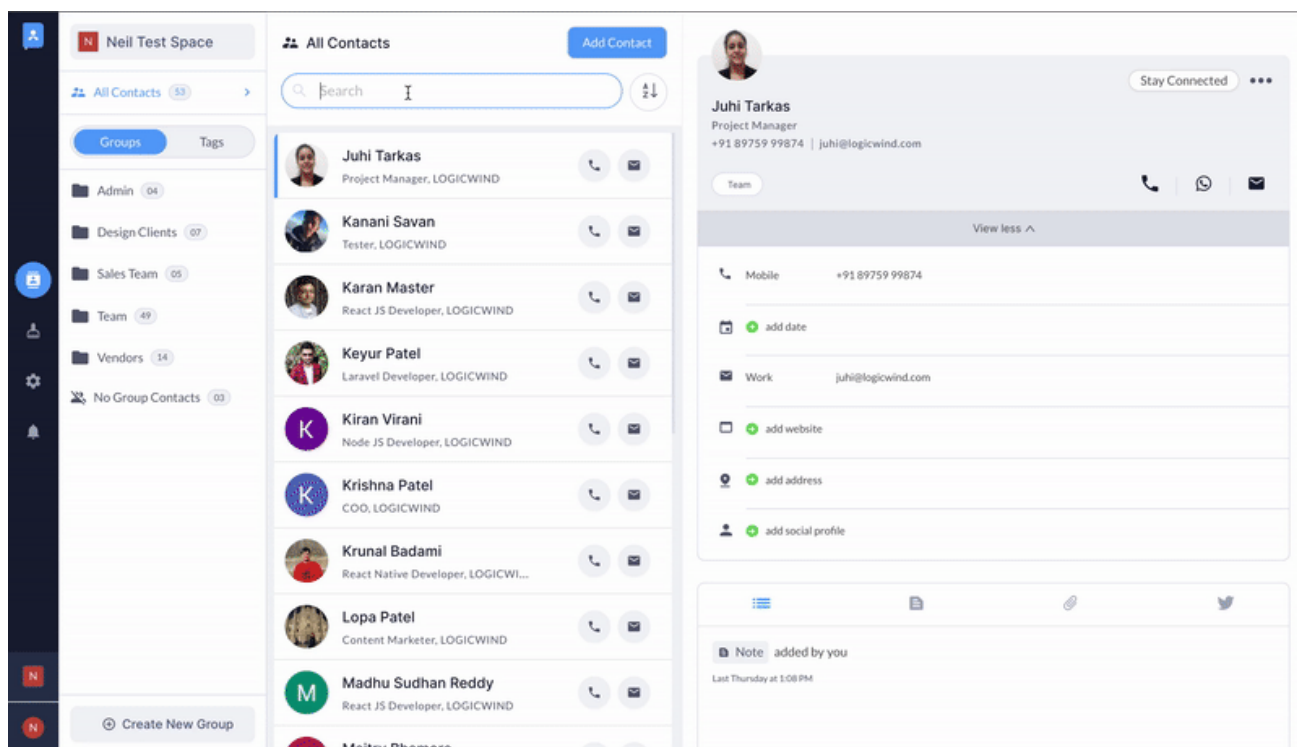


Рисунок 7 – Интерфейс ContactBook

При рассмотрении приложений-аналогов были выявлены как схожие функциональные возможности, так и уникальные для каждого приложения.

1.2.3 Результаты сравнительного анализа

Для сравнения приложений-аналогов были выделены следующие критерии:

- возможность работы с объектами – создание, изменение, удаление заметок или контактов;
- персонализация объекта – возможность настройки внешнего вида заметки или контакта, включая цветовое оформление, шрифты и др.;
- возможность создания групп – выделение категорий объектов для эффективной работы с ними;
- импорт – возможность загрузки данных из сторонних файлов;
- экспорт – возможность сохранения данных в различных текстовых форматах;

- печать выбранных контактов или заметок;
- возможность поиска необходимых данных;
- общий доступ – предоставление возможности просмотра или редактирования объектов другим пользователям;
- напоминания – уведомление пользователя о событиях, важных событиях.

Результаты анализа рассмотренных приложений по выбранным критериям представлены в таблице 3.

Таблица 3 – Итоговая таблица сравнительного анализа

Функции	Приложения				
	Google Keep	Яндекс Заметки	Simplenote	Google Контакты	ContactBook
Возможность работы с объектами	Создание, изменение, удаление, архивирование	Создание, изменение, удаление	Создание, изменение, удаление	Создание, изменение, удаление	Создание, изменение, удаление
Создание групп	+ (ярлыки)	-	+ (теги)	+ (группы)	+ (теги)
Импорт	-	-	-	+	+
Экспорт	-	-	-	В формате csv	-
Печать	-	-	-	+	-
Поиск	+	-	+	+	+
Общий доступ	+	-	+	-	+
Напоминания	+	-	-	-	+

В результате анализа было выявлено, что основные функции работы с объектами, включая создание, удаление и изменение данных, поддерживаются во всех рассмотренных приложениях. Также все средства, кроме Google Контактов, позволяют задавать стилизацию объектов. Помимо этого, в большинстве приложений есть возможность поиска необходимой информации, исключение составляет сервис Яндекс Заметки.

Среди приложений, предназначенных для ведения заметок сложно отдать предпочтение какому-либо. Все три онлайн-сервиса имеют одинаковый основной функционал, однако Яндекс Заметки уступают Google Keep и Simplenote в

возможности категоризации заметок и предоставлении общего доступа другим пользователям.

Для рассмотрения приложений для создания и работы с контактами были выбраны ContactBook и Google Контакты. Google Контакты имеют дополнительные функциональные возможности в виде экспорта выбранных контактов и печати данных, однако в ContactBook поддерживается функция установки напоминаний. Платный сервис ContactBook поддерживает только импорт данных, в то время как бесплатный – Google Контакты имеет поддержку, как импорта, так и экспорта данных. В целом, выбор приложения для работы с контактами зависит от целей и предпочтений пользователя.

В связи с разработкой системы, работающей как с заметками, так и с контактами необходимо определить ее функциональные возможности.

В большинстве рассмотренных веб-приложений присутствуют базовые функции, которые необходимо будет реализовать в разрабатываемой системе. Среди них:

- работа с основными объектами системы (заметками, контактами), включая создание, модификацию и удаление из системы;
- создание категорий объектов системы;
- поиск объектов в системе.

Кроме базовых функций были выделены дополнительные возможности, которые позволят обеспечить конкурентоспособность реализуемой системы. Одной из таких возможностей является настройка напоминаний, прикрепляемых к объектам и позволяющих уведомить пользователя о событиях. Также считается необходимым предусмотреть в системе возможность экспорта данных в текстовый файл с сохранением в локальное хранилище пользователя.

1.3 Постановка задачи на разработку приложения

1.3.1 Цели разработки и назначение приложения

Целью разработки является создание многофункциональной системы.

Такая система должна предоставить доступ как к заметкам пользователя, так и к информации о контактах.

Приложение предназначено для хранения и эффективной работы с данными, включая добавление и модификацию актуальной информации или удаление ненужной. Система должна предоставить инструменты для организации поиска и категоризации информации, а также автоматизировать процесс отправки напоминаний, установленных пользователем.

Реализуемая система ориентирована на пользователей, у которых возникает необходимость организации заметок и управления контактными данными, т. е. может быть применена как для личного использования, так и для работы.

1.3.2 Перечень автоматизируемых функций

Разрабатываемое приложение должно обеспечивать работу с двумя основными типами информационных объектов – Kontakтами и Заметками.

Соответственно для контактов необходимо реализовать следующие функции:

- добавление контакта – внесение контактной информации конкретного человека;
- удаление контакта – удаление контактной информации конкретного человека;
- модификация контакта – редактирование контактной информации о конкретном человеке;
- настройка напоминания – установка уведомления о дне рождении конкретного контакта;
- добавление тега контакту – занесения контакта в существующую категорию.

Для заметок необходимо реализовать следующие функции:

- добавление заметки – внесение информации о конкретной заметке;
- удаление заметки – удаление информации конкретной заметки;

- модификация заметки – редактирование информации конкретной заметки;
- настройка напоминания – установка уведомления о событии, которое описывает заметка;
- добавление тега заметке – занесение заметки в существующую категорию.

Приложение также должно поддерживать такие функциональные возможности как:

- заполнение данных о пользователе, реализация личного кабинета;
- создание тега – категории для систематизации контактов и заметок;
- просмотр и фильтрация заметок и контактов по тегам;
- поиск информационных объектов в системе по различным параметрам.

Кроме того, система должна позволять сохранять пользователю выбранные данные в текстовом формате в локальное хранилище.

1.4 Планирование и оценка процесса разработки приложения

1.4.1 Выбор нотации и программных средств для планирования и оценки процесса разработки приложения

Планирование процесса разработки приложения является одним из первоначальных этапов в управлении проектом. На этом этапе составляется перечень работ и имеющихся ресурсов, производится распределение ресурсов по работам и оптимизация плана по критериям времени завершения проекта, суммарной стоимости проекта и равномерного распределения ресурсов. С целью своевременного выявления отклонений от первоначального плана необходим контроль выполнения проекта, а в случае значительных отклонений корректировка существующего плана [1].

Для осуществления процесса планирования разработки приложения электронной контактной книги была выбрана диаграмма Ганта. Данная

диаграмма представляет собой инструмент в управлении проектами, который отражает информацию по этапам реализации проекта и срокам каждого из участков работ [7].

Планирование и оценка разработки приложения производилось с помощью программного средства Microsoft Project 2007.

Microsoft Project – программа для управления проектами, которая позволяет распределить ресурсы по задачам, определить длительность задач, отслеживать прогресс и анализировать объем работ.

Аналогами данного средства являются ADVANTA, Asana, Jira и др.

1.4.2 Планирование процесса разработки приложения

Для планирования в Microsoft Project 2007 процесс разработки необходимо разбить на задачи. Задача – это фиксированное действие, предназначенное для реализации отдельной части проекта. Каждая задача обрабатывает длительностью, т. е. временем, которое необходимо для ее выполнения. Есть особый вид задач, которые не имеют длительности – вехи. Их предназначение заключается в обозначении моментов начала или завершения определенных этапов проекта. Для выполнения тех или иных задач назначаются ресурсы [8].

Ресурсы бывают двух видов: материальные и трудовые. Материальными ресурсами проекта считается все то, что оплачивается за количество. В случае с проектом электронной контактной книги, материальным ресурсом является ноутбук. Трудовые ресурсы – это люди, выполняющие работу по задаче, разработчики, участвующие в создании проекта. Основной характеристикой данного ресурса является почасовая ставка.

На рисунке 8 представлены запланированные задачи проекта, их длительность, предшественники, а также ресурсы, назначенные на задачи.

№	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало реализации проекта	0 дней	Чт 04.04.24	Чт 04.04.24		
2	☒ Анализ предметной области и постановка задачи на разработку	6 дней	Чт 04.04.24	Пт 12.04.24		
3	Анализ предметной области приложения	1 день	Чт 04.04.24	Пт 05.04.24	1	Аналитик
4	Сравнительный анализ аналогичных разработок	2 дней	Пт 05.04.24	Вт 09.04.24	3	Аналитик
5	Постановка задачи на разработку приложения	1 день	Вт 09.04.24	Ср 10.04.24	4	Аналитик
6	Планирование и оценка процесса разработки приложения	2 дней	Ср 10.04.24	Пт 12.04.24	5	Аналитик
7	Анализ предметной области и постановка задачи завершения(а)	0 дней	Пт 12.04.24	Пт 12.04.24	6	
8	☒ Проектирование приложения	7,5 дней	Пт 12.04.24	Вт 23.04.24	2	
9	Моделирование функциональных возможностей	2,25 дней	Пт 12.04.24	Вт 16.04.24	7	Бэкенд-разработчик[50%]; Фронтенд-разработчик[50%]
10	Клиент-серверная архитектура электронной контактной книги	2 дней	Вт 16.04.24	Чт 18.04.24	9	Фронтенд-разработчик
11	Разработка прототипа пользовательского интерфейса клиентского	1 день	Чт 18.04.24	Пт 19.04.24	10	Фронтенд-разработчик
12	Разработка требований к пользовательскому интерфейсу клиента	1 день	Пт 19.04.24	Сб 20.04.24	11	Фронтенд-разработчик
13	Требования к аппаратно-программному обеспечению клиентского	1 день	Пн 22.04.24	Пн 22.04.24	12	Фронтенд-разработчик[50%]; Бэкенд-разработчик[50%]
14	Обоснование выбора технологий и средств разработки приложения	1 день	Вт 23.04.24	Вт 23.04.24	13	Фронтенд-разработчик
15	Проектирование приложения завершено	0 дней	Вт 23.04.24	Вт 23.04.24	14	
16	☒ Разработка клиентского приложения	15,5 дней	Ср 24.04.24	Ср 15.05.24	8	
17	Разработка архитектуры клиентского приложения	2 дней	Ср 24.04.24	Чт 25.04.24	15	Фронтенд-разработчик
18	Разработка пользовательского интерфейса	12 дней	Ср 24.04.24	Ср 08.05.24	15	Фронтенд-разработчик
19	Разработка средств взаимодействия с серверным приложением	3 дней	Чт 09.05.24	Пн 13.05.24	18;17	Фронтенд-разработчик
20	Разработка бизнес-логики клиентского приложения	2 дней	Пн 13.05.24	Ср 15.05.24	19	Фронтенд-разработчик
21	Разработка клиентского приложения завершена	0 дней	Ср 15.05.24	Ср 15.05.24	20	
22	☒ Контроль качества программного обеспечения клиентского прил	10 дней	Ср 15.05.24	Ср 29.05.24	16	
23	Планирование процесса контроля качества программного обеспе	1 день	Ср 15.05.24	Чт 16.05.24	21	Тестировщик
24	Тестирование программных средств	6 дней	Чт 16.05.24	Пт 24.05.24	23	Тестировщик
25	Оценка качества программного кода на основе метрик	3 дней	Пт 24.05.24	Ср 29.05.24	24	Тестировщик
26	Контроль качества завершен	0 дней	Ср 29.05.24	Ср 29.05.24	25	
27	☒ Сдача проекта	6,5 дней	Ср 29.05.24	Чт 06.06.24	22	
28	Составление документации	5 дней	Ср 29.05.24	Вт 04.06.24	26	Фронтенд-разработчик
29	Ввод в эксплуатацию	2 дней	Ср 05.06.24	Чт 06.06.24	28	Бэкенд-разработчик[50%]; Фронтенд-разработчик[50%]
30	Сдача проекта окончена	0 дней	Чт 06.06.24	Чт 06.06.24	29	
31	Конец проекта	0 дней	Чт 06.06.24	Чт 06.06.24	30	

Рисунок 8 – Перечень задач и распределение ресурсов проекта

После заполнения таблицы с задачами и распределения ресурсов программой была сгенерирована диаграмма Ганта, представленная на рисунке 9.

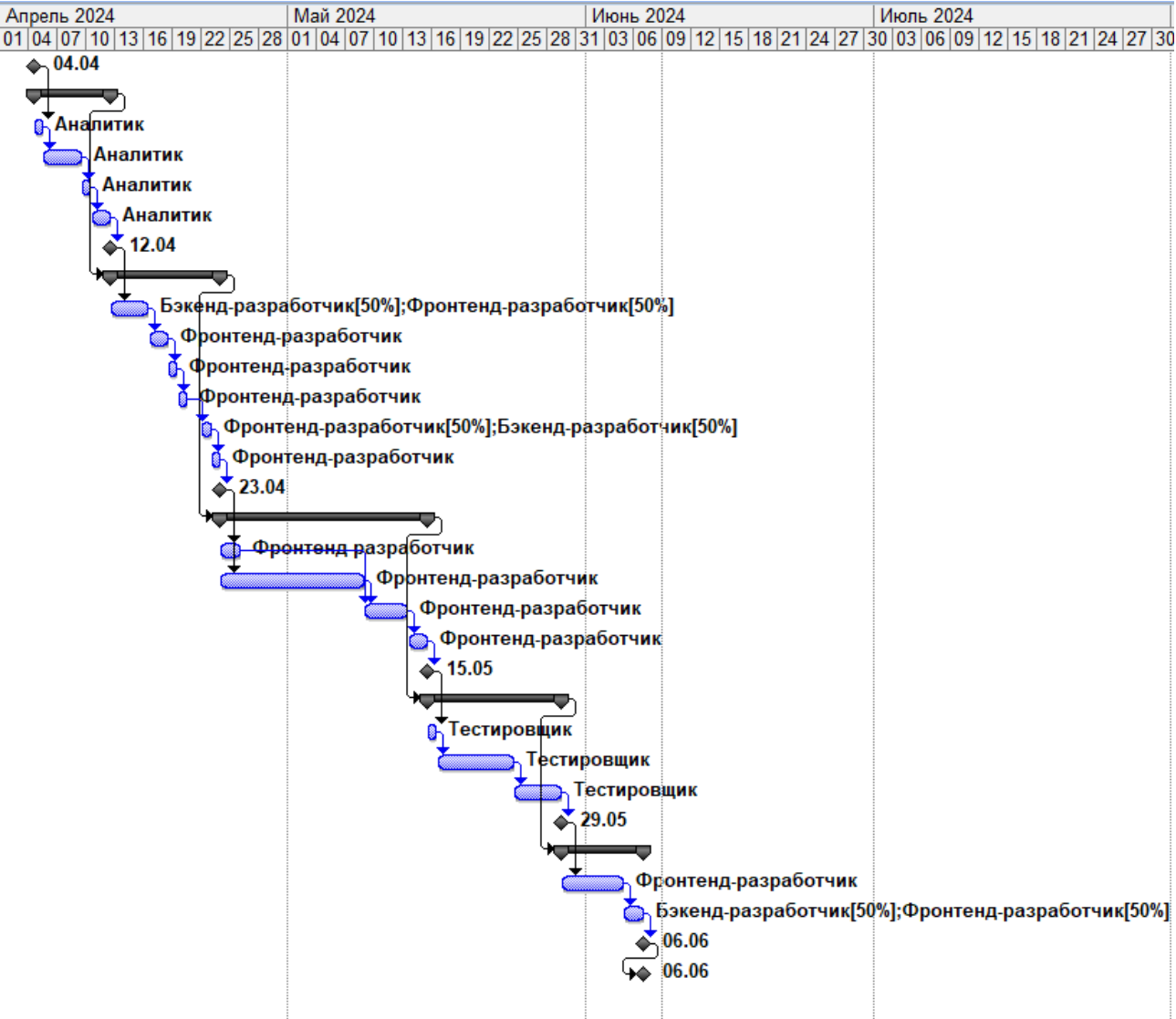


Рисунок 9 – Диаграмма Ганта

Диаграмма Ганта отображает следующие параметры проекта:

- структуру работ, полученную на основе сетевого графика;
- состав используемых ресурсов и их распределение между работами;
- календарные даты, к которым привязываются моменты начала и завершения работ.

Согласно диаграмме завершение работ по проекту планируется на 6 июня.

1.4.3 Оценка временных и финансовых затрат на разработку

Для оценки затрат в Microsoft Project существует возможность генерации отчетов.

На рисунке 10 представлены временные затраты и ресурсы, необходимые для реализации проекта. Отчет о финансовых затратах приведен на рисунке 11.

	03.06.24	10.06.24	Итого
Аналитик			48 ч
Анализ предметной области приложения			8 ч
Сравнительный анализ аналогичных разработок			16 ч
Постановка задачи на разработку приложения			8 ч
Планирование и оценка процесса разработки приложения			16 ч
Тестировщик			80 ч
Планирование процесса контроля качества программного обеспечения			8 ч
Тестирование программных средств			48 ч
Оценка качества программного кода на основе метрик			24 ч
Бэкенд-разработчик	8 ч		20 ч
Моделирование функциональных возможностей			8 ч
Требования к аппаратно-программному обеспечению клиентского приложения			4 ч
Ввод в эксплуатацию	8 ч		8 ч
Фронтенд-разработчик	15,7 ч		186,7 ч
Моделирование функциональных возможностей			3 ч
Клиент-серверная архитектура электронной контактной книги			16 ч
Разработка прототипа пользовательского интерфейса клиентского приложения			8 ч
Разработка требований к пользовательскому интерфейсу клиентского приложения			8 ч
Требования к аппаратно-программному обеспечению клиентского приложения			4 ч
Обоснование выбора технологий и средств разработки приложения			8 ч
Разработка архитектуры клиентского приложения			16 ч
Разработка пользовательского интерфейса			56,5 ч
Разработка средств взаимодействия с серверным приложением			19,5 ч
Разработка бизнес-логики клиентского приложения			12,5 ч
Составление документации	12,5 ч		32 ч
Ввод в эксплуатацию	3,2 ч		3,2 ч
Ноутбук			
Итого	23,7 ч		334,7 ч

Рисунок 10 – Отчет об использовании ресурсов

В проекте используются как материальные, так и трудовые ресурсы. В качестве материального ресурса используется ноутбук, к трудовым ресурсам относятся аналитик, тестировщик, бэкенд-разработчик, а также фронтенд-разработчик.

По данным отчета можно определить, что общая занятость ресурсов составила 334,7 ч.

Ид.	Название задачи	Фиксированные затраты	Начисление фикс. затрат	Общие затраты	Базовые	Отклонение
18	Разработка пользовательского инте	0,00 Р	Пропорциональное	14 125,00 Р	0,00 Р	14 125,00 Р
24	Тестирование программных средств	0,00 Р	Пропорциональное	8 160,00 Р	0,00 Р	8 160,00 Р
28	Составление документации	0,00 Р	Пропорциональное	8 000,00 Р	0,00 Р	8 000,00 Р
19	Разработка средств взаимодействия	0,00 Р	Пропорциональное	4 875,00 Р	0,00 Р	4 875,00 Р
25	Оценка качества программного кода	0,00 Р	Пропорциональное	4 080,00 Р	0,00 Р	4 080,00 Р
10	Клиент-серверная архитектура элект	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
17	Разработка архитектуры клиентского	0,00 Р	Пропорциональное	4 000,00 Р	0,00 Р	4 000,00 Р
20	Разработка бизнес-логики клиентско	0,00 Р	Пропорциональное	3 125,00 Р	0,00 Р	3 125,00 Р
4	Сравнительный анализ аналогичны	0,00 Р	Пропорциональное	3 040,00 Р	0,00 Р	3 040,00 Р
6	Планирование и оценка процесса ра	0,00 Р	Пропорциональное	3 040,00 Р	0,00 Р	3 040,00 Р
29	Ввод в эксплуатацию	0,00 Р	Пропорциональное	2 800,00 Р	0,00 Р	2 800,00 Р
9	Моделирование функциональных во	0,00 Р	Пропорциональное	2 750,00 Р	0,00 Р	2 750,00 Р
11	Разработка прототипа пользователя	0,00 Р	Пропорциональное	2 000,00 Р	0,00 Р	2 000,00 Р
12	Разработка требований к пользоват	0,00 Р	Пропорциональное	2 000,00 Р	0,00 Р	2 000,00 Р
13	Требования к аппаратно-программн	0,00 Р	Пропорциональное	2 000,00 Р	0,00 Р	2 000,00 Р
14	Обоснование выбора технологий и с	0,00 Р	Пропорциональное	2 000,00 Р	0,00 Р	2 000,00 Р
3	Анализ предметной области прилож	0,00 Р	Пропорциональное	1 520,00 Р	0,00 Р	1 520,00 Р
5	Постановка задачи на разработку пр	0,00 Р	Пропорциональное	1 520,00 Р	0,00 Р	1 520,00 Р
23	Планирование процесса контроля ка	0,00 Р	Пропорциональное	1 360,00 Р	0,00 Р	1 360,00 Р
1	Начало реализации проекта	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
7	Анализ предметной области и поста	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
15	Проектирование приложения завер	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
21	Разработка клиентского приложения	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
26	Контроль качества заверше	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
30	Сдача проекта окончена	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
31	Конец проекта	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
				74 395,00 Р	0,00 Р	74 395,00 Р

Рисунок 11 – Отчет о бюджете проекта

В результате оценки финансовых затрат было установлено, что стоимость создания клиентского приложения электронной контактной книги с возможностью ведения заметок составляет 74 395,00 руб.

2 Проектирование приложения

2.1 Моделирование функциональных возможностей приложения

Для описания функциональных возможностей разрабатываемого приложения была создана диаграмма вариантов использования в нотации UML. Компонентами диаграммы вариантов использования являются следующие:

- граница системы – прямоугольник, внутри которого размещаются варианты использования;
- актер – сущность, которая взаимодействует с системой;
- варианты использования – поведение системы, действия, которые она выполняет при взаимодействии с актерами;
- отношения – связи, описывающие взаимодействие актеров и вариантов использования.

В языке UML существует несколько стандартных видов отношений:

- ассоциации (association relationship);
- расширения (extend relationship);
- обобщения (generalization relationship);
- включения (include relationship) [1].

Как можно видеть на рисунке 12, на диаграмме используются два вида отношений: ассоциации и включения. Отношение ассоциации служит для обозначения роли актера при его взаимодействии с конкретным вариантом использования. Отношение ассоциации обозначается сплошной линией. Отношение включения устанавливается между двумя вариантами использования и указывает на то, что поведение одного варианта использования включается в качестве составной части в поведение другого варианта использования. Отношение включения обозначается пунктирной линией со стрелкой, направленной к включаемому варианту использования [9].

Диаграмма вариантов использования, описывающая функциональные возможности разрабатываемого приложения, представлена на рисунке 12.

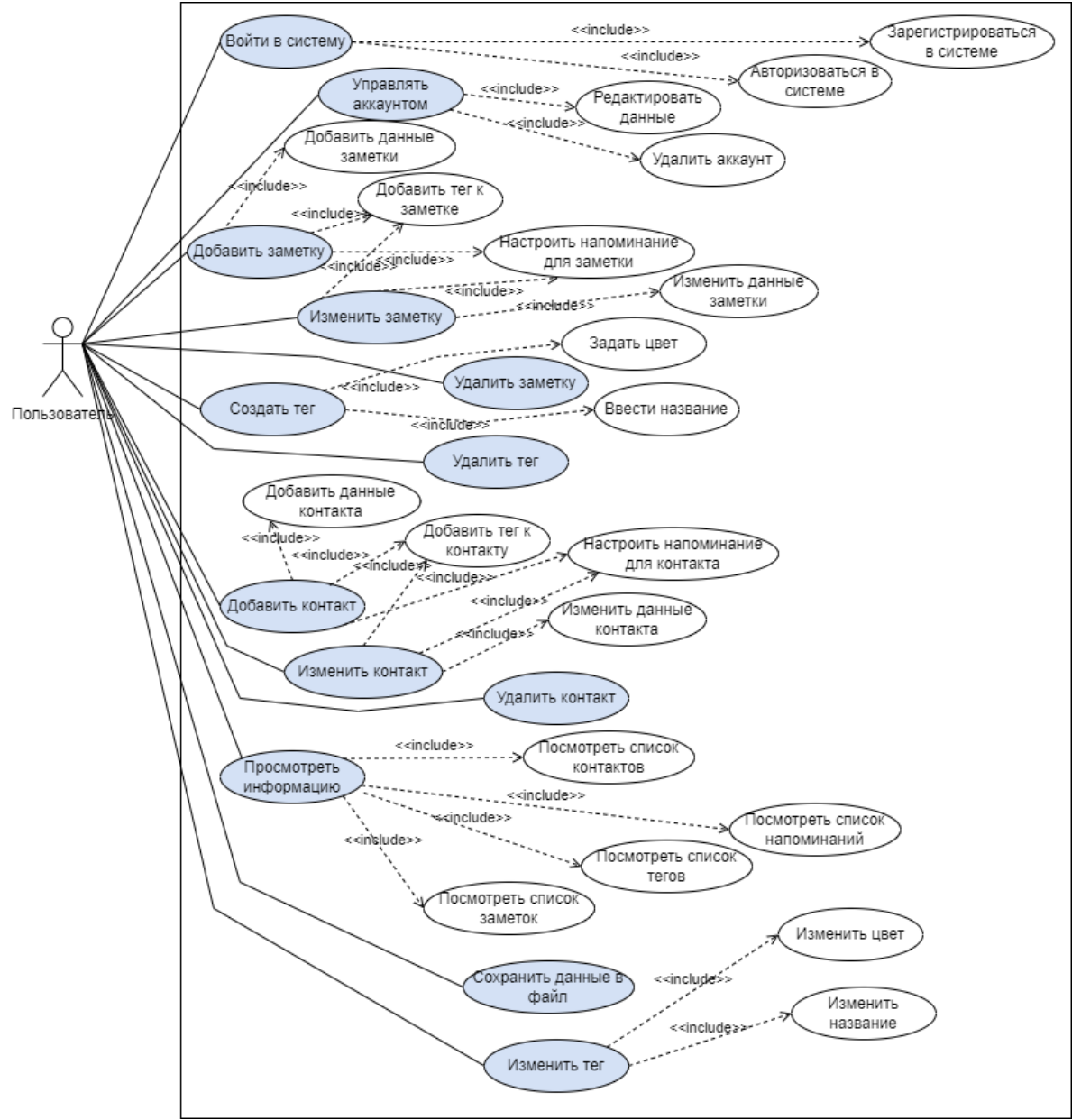


Рисунок 12 – Диаграмма вариантов использования приложения

Краткое описание основных вариантов использования диаграммы представлено в таблице 4.

Таблица 4 – Описание вариантов использования

Вариант использования	Описание
Войти в систему	Получение доступа к данным системы и возможности работы с ними с помощью регистрации или авторизации в системе

Продолжение таблицы 4

Вариант использования	Описание
Управлять аккаунтом	Редактирование данных в личном кабинете или удаление профиля пользователя
Добавить заметку	Создание новой заметки в системе
Изменить заметку	Модификация данных заметки, изменение категории заметки или напоминания
Удалить заметку	Удаление заметки из системы
Создать тег	Создание категории с новым названием (цветом)
Изменить тег	Модификация
Удалить тег	Удаление категории объектов, при этом сами объекты остаются в системе
Добавить контакт	Создание нового контакта в системе
Изменить контакт	Модификация данных контакта, изменение категории или напоминания
Удалить контакт	Удаление контакта из системы
Просмотреть информацию	Обращение к информации системы: контактам, заметкам, тегам или напоминаниям
Сохранить данные в файл	Экспорт выбранных объектов в текстовый файл

Сценарии выполнения некоторых вариантов использования приведены на рисунках 13 и 14.

На рисунке 13 представлен сценарий выполнения варианта использования «Войти в систему».

Наименование	Войти в систему
ID	1
Краткое описание	Пользователь регистрируется или авторизуется в приложении
Действующие лица	Пользователь
Предусловие	Пользователь не был зарегистрирован/авторизован
Основной поток	1. Пользователь переходит на страницу входа в приложение 2. Пользователь вводит данные для входа в существующий аккаунт (логин и пароль) 3. Пользователь получает доступ к функционалу приложения
Альтернативный поток	2.а. Пользователь вводит данные для регистрации (логин и пароль)
Постусловие	Пользователь вошел в приложение, ему доступен функционал

Рисунок 13 – Спецификация варианта использования «Войти в систему»

На рисунке 14 представлен сценарий выполнения варианта использования «Добавить контакт».

Наименование	Добавить контакт
ID	2
Краткое описание	Пользователь создает новый контакт в системе
Действующие лица	Пользователь
Предусловие	Пользователь прошел авторизацию и перешел во вкладку «Контакты»
Основной поток	<ol style="list-style-type: none"> 1. Пользователь заносит данные о контакте в соответствующие поля, включая фамилию, имя, отчество, адрес электронной почты, дату рождения и др. 2. Пользователь добавляет тег контакту, выбирая из списка существующих категорий 3. Пользователь устанавливает напоминание о дне рождения контакта
Альтернативный поток	<ol style="list-style-type: none"> 2.а. Пользователь не добавляет тег контакту 3.а. Пользователь отключает напоминание о дне рождения контакта
Постусловие	Контакт создан

Рисунок 14 – Спецификация варианта использования «Добавить контакт»

В результате моделирования функциональных возможностей были определены границы системы, а также сформулированы общие требования к функциональному поведению проектируемой системы.

2.2 Клиент-серверная архитектура электронной контактной книги

Архитектура электронной контактной книги представляет собой клиент-серверную архитектуру. Данная архитектура состоит из двух компонентов: клиента, который запрашивает информацию и сервера, который владеет необходимой информацией и в случае запроса возвращает ее клиенту. Благодаря разделению функционала, эффективно распределяется нагрузка между элементами архитектуры, что гарантирует увеличение скорости обработки информации. Сервер централизованно управляет ресурсами и контролирует информацию, предоставляемую клиентам. Это облегчает управление и позволяет применить единые политики безопасности и доступа к сведениям [10]. При этом низкие требования к клиентским устройствам делают архитектуру доступной для широкого круга пользователей.

Для взаимодействия между клиентом и сервером был выбран архитектурный подход REST API. При этом обмен сообщениями осуществляется обычно по протоколу HTTP(S). Протокол передачи данных определяет методы запроса к ресурсу.

Как правило, достаточно четырех методов:

- GET – получение информации об объекте (ресурсе);
- POST – создание, а иногда и изменение объекта (ресурса);
- PUT – изменение объекта (ресурса). Обычно используется для регулярного обновления данных;
- DELETE – удаление информации об объекте (ресурсе).

Главная особенность REST API – обмен сообщениями без сохранения состояния. Каждое сообщение самостоятельное и содержит всю информацию, необходимую для его обработки. Сервер не хранит результаты предыдущих сессий с клиентскими приложениями. Это обеспечивает гибкость и масштабируемость серверной части, позволяет поддерживать асинхронные взаимодействия и реализовывать алгоритмы обработки любой сложности. Кроме того, такой формат взаимодействия является универсальным – он не зависит от технологий, используемых на клиенте и на сервере [11].

Для визуализации архитектуры была разработана диаграмма развертывания, представленная на рисунке 15.

Диаграмма развертывания – это UML-диаграмма, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их [12]. Одним из элементов диаграммы является узел. Он представляет собой либо среду выполнения программного обеспечения, либо физическое устройство и изображается в виде куба. Элементы, представляющие собой результаты разработки, называются артефактами. Артефактами могут быть файлы конфигурации, библиотеки, исполняемые файлы и др.

Диаграмма состоит из трех элементов: сервера приложений (Web Server), компьютера клиента с любым web-браузером и хранилища данных.

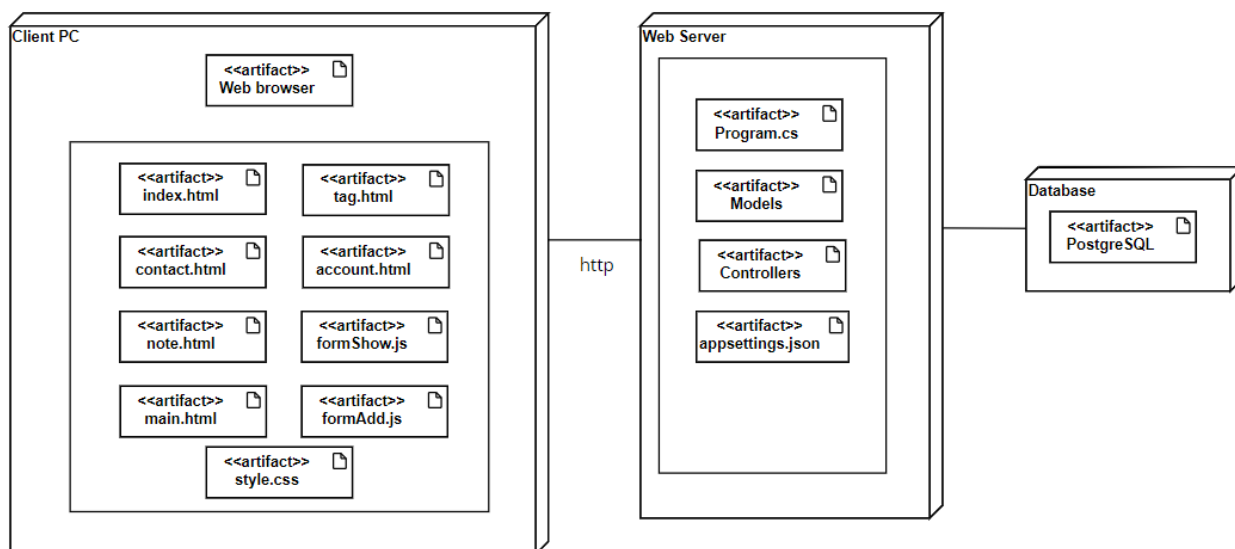


Рисунок 15 – Архитектура приложения

Клиентское приложение представляет собой многостраничное приложение. Для создания web-приложения будет использован фреймворк ASP.NET Core. Он поддерживает развертывание приложений на веб-сервере IIS (Internet Information Services) компании Microsoft. Взаимодействие клиента и сервера производится по протоколу HTTP.

2.3 Разработка прототипа пользовательского интерфейса клиентского приложения

Интерфейс клиентского приложения должен обеспечивать привычное и удобное представление информации, содержать знакомую терминологию для обычного пользователя.

Поскольку электронная контактная книга содержит несколько информационных объектов, а также инструменты для работы с ними, то для навигации пользователя интерфейс веб-приложения должен содержать меню. В связи с использованием системы количество заметок и контактов будет увеличиваться, что затруднит возможность ориентирования пользователя в информации. В связи с этим необходимо реализовать поле для поиска нужных объектов.

Графический интерфейс должен иметь интуитивно понятные элементы управления, включая меню, формы для заполнения, кнопки для выполнения операций, а также знакомые обозначения, чтобы облегчить понимание и использование приложения пользователями.

Интерфейс клиентского приложения представляет собой совокупность статических страниц и форм, необходимых для работы с пользовательскими данными. Разработка прототипа интерфейсов страниц клиентского приложения проводилась в графическом редакторе Figma.

На рисунке 16 представлен прототип интерфейса страницы входа в приложение, содержащей формы регистрации и авторизации, а также основную информацию о приложении и его функциональных возможностях.

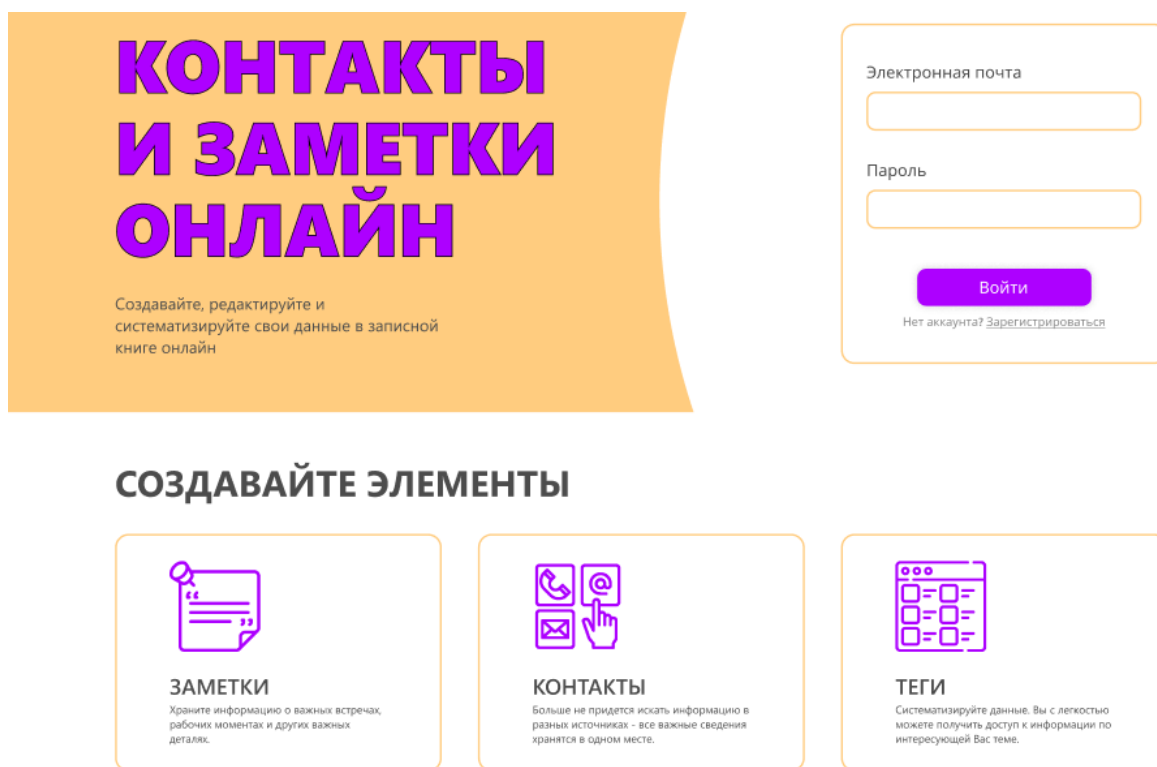


Рисунок 16 – Прототип интерфейса страницы входа

Приложение должно иметь навигацию между страницами, соответственно в верхней части интерфейса необходимо расположить панель навигации.

В зависимости от раздела приложения, открытого пользователем, на странице отображаются либо контакты, либо заметки, либо теги. На главной

странице должно быть представлено отображение недавно созданных элементов.

Главная страница приложения с отображением недавно созданных объектов представлена на рисунке 17.

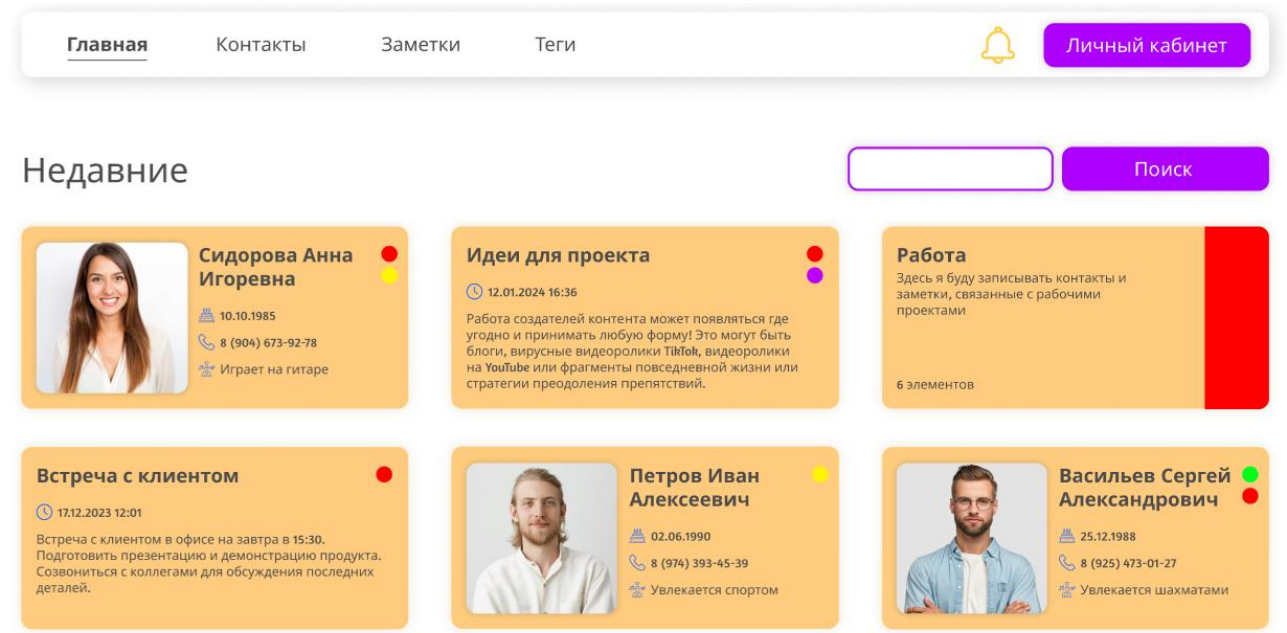


Рисунок 17 – Прототип интерфейса главной страницы

Прототип интерфейса страницы для работы с контактами приведен на рисунке 18.

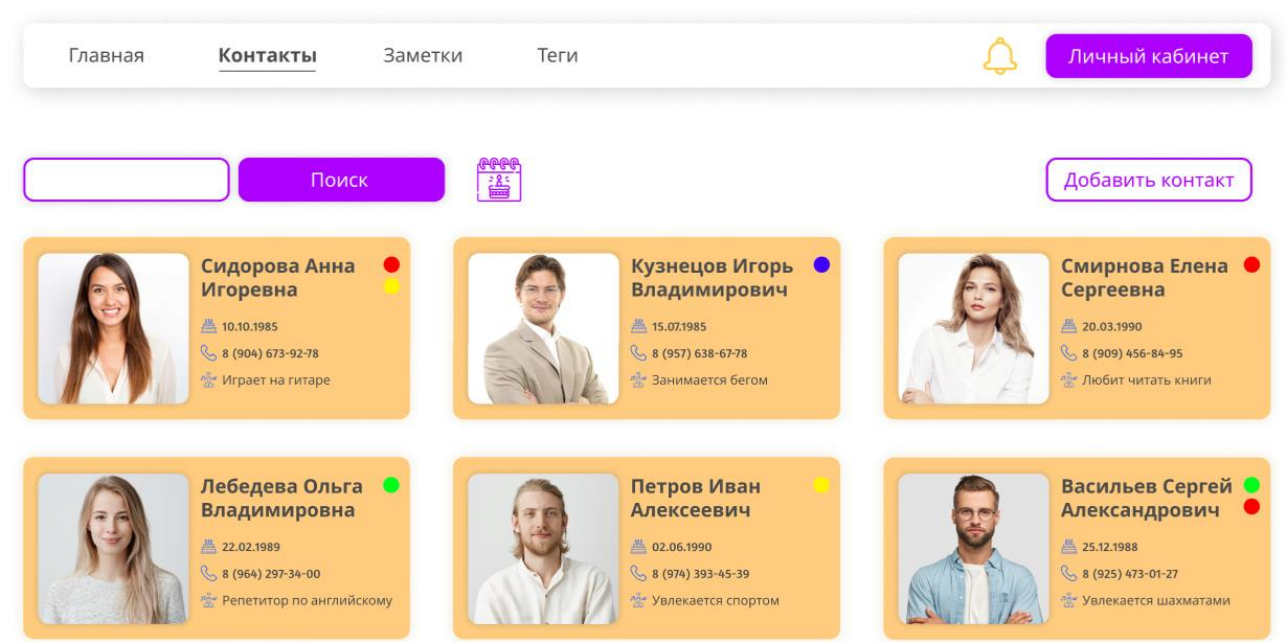


Рисунок 18 – Прототип интерфейса страницы контактов

На рисунке 19 представлен прототип интерфейса страницы для работы с заметками.

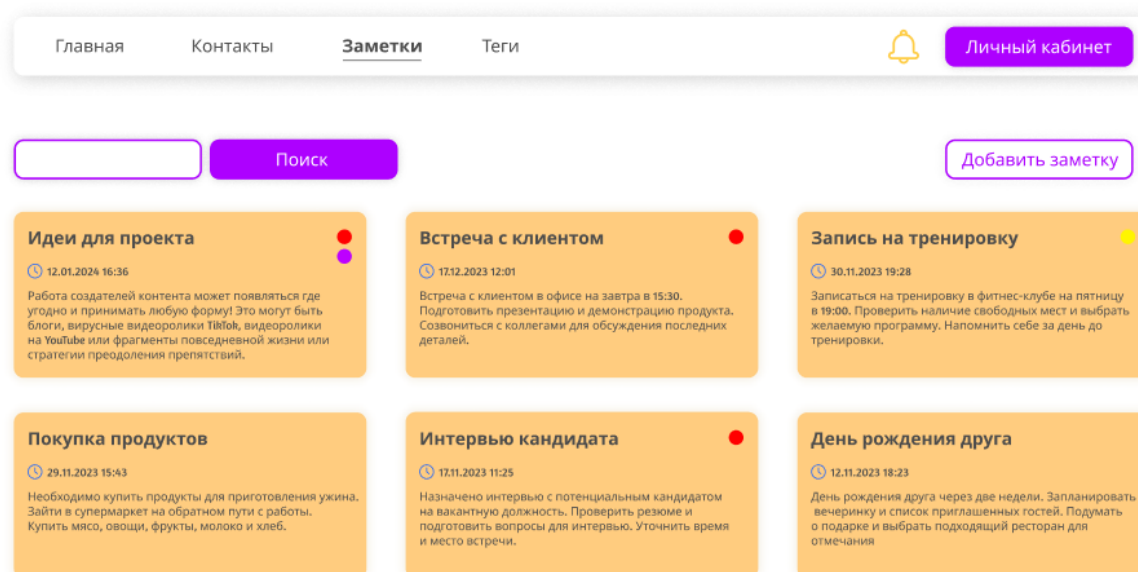


Рисунок 19 – Прототип интерфейса страницы заметок

Страница для создания тегов и просмотра объектов в существующих категориях представлена на рисунке 20.

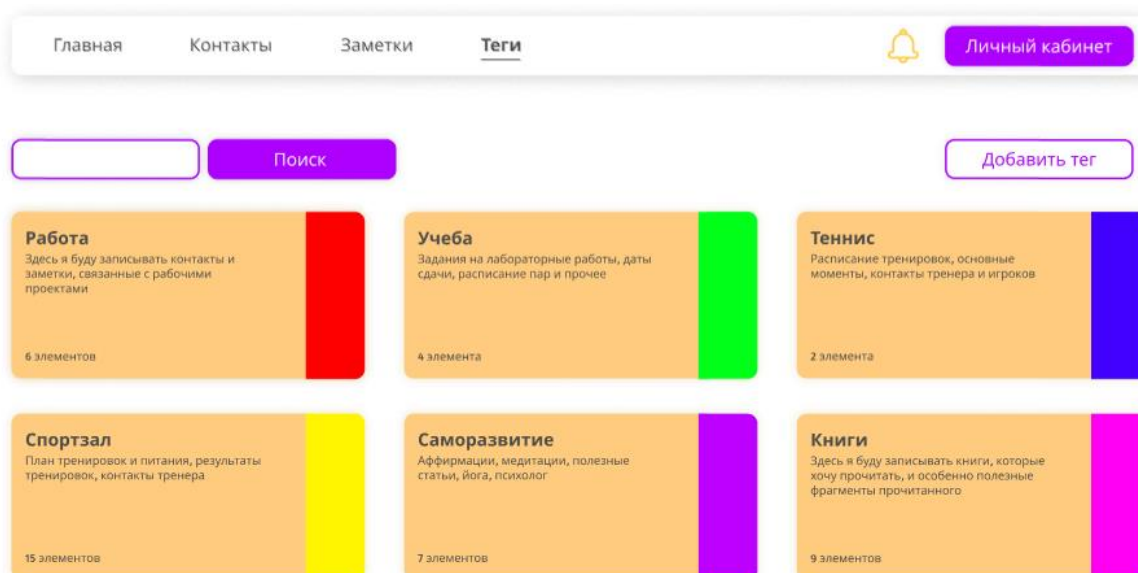


Рисунок 20 – Прототип интерфейса страницы тегов

Для просмотра и редактирования информации о себе пользователю предоставляется личный кабинет, интерфейс которого приведен на рисунке 21.

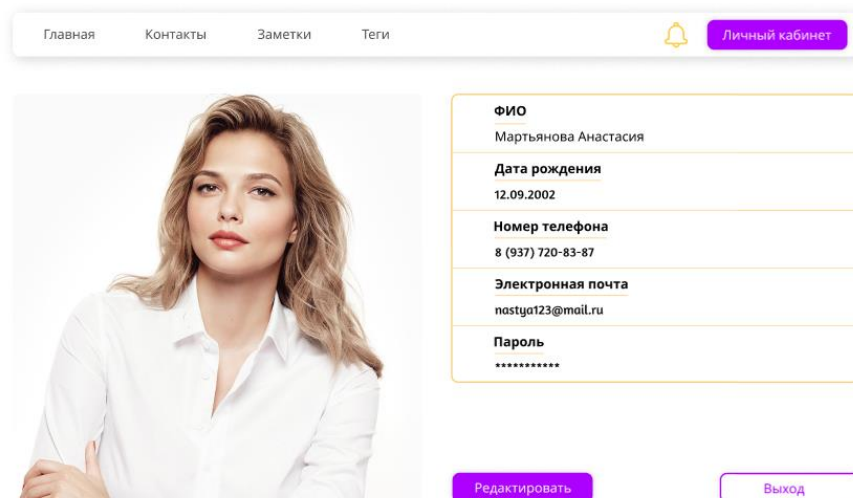


Рисунок 21 – Прототип интерфейса страницы личного кабинета пользователя

Для понимания связи страниц между собой была разработана схема переходов между страницами, приведенная на рисунке 22.

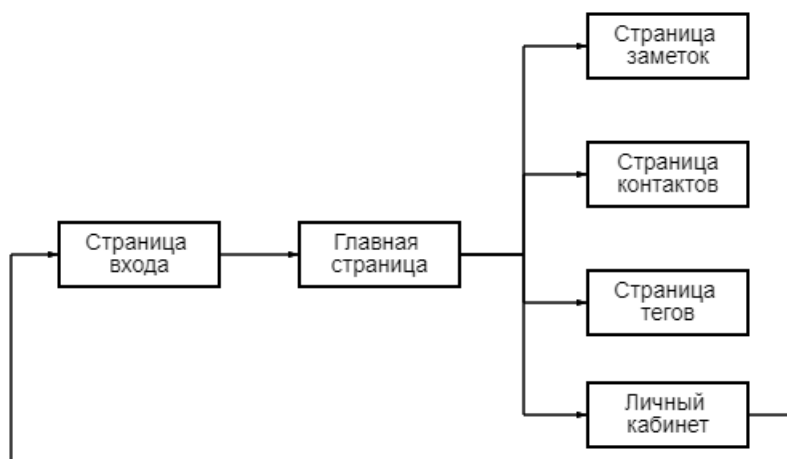


Рисунок 22 – Схема переходов между страницами

В соответствии со схемой после успешного входа пользователь перенаправляется на главную страницу приложения и имеет доступ ко всем оставшимся страницам. Выход из приложения возможен только при переходе пользователем в личный кабинет.

Прототипы интерфейсов и схема связи страниц были использованы при разработке клиентского приложения электронной контактной книги.

2.4 Требования к аппаратно-программному обеспечению клиентского приложения

Для разработки клиентского приложения электронной контактной книги предъявляются следующие минимальные требования к аппаратному обеспечению:

- компьютер с процессором Intel Core i3 и выше;
- оперативная память не менее 4 Гб;
- операционная система Windows 10;
- свободное место на жестком диске не менее 15 Гб.

На компьютере необходимо наличие установленного редактора кода Visual Studio Code, а также плагина Prettier для облегчения процесса форматирования кода.

В качестве фреймворка для адаптивной верстки интерфейса приложения используется Bootstrap 5.2.3, подключаемый к HTML-страницам с помощью CDN ссылок.

Для отслеживания изменений в файлах необходимо наличие системы управления версиями Git, а именно приложения для работы командной строки Git – Git Bash.

В связи с разработкой веб-приложения доступ к реализуемой системе должен обеспечиваться с помощью браузера. В связи с набором используемых технологий, корректное отображение контента обеспечивают последние версии следующих браузеров:

- Google Chrome;
- Firefox;
- Microsoft Edge;
- Opera;
- Safari.

Помимо этого, при разработке необходимо стабильное Интернет-соединение для доступа к ресурсам сети.

2.5 Обоснование выбора технологий и средств разработки приложения

В качестве редактора кода для реализации клиентского приложения был выбран Visual Studio Code.

Visual Studio Code – это редактор кода с функциями IDE для кроссплатформенной разработки веб-приложений. Редактор поддерживает технологию автодополнения кода IntelliSense, подсветку синтаксиса и интегрируется с системой контроля версий Git [13]. Среда разработки является легко настраиваемой и расширяемой за счет установки дополнительных плагинов для добавления новых языков, отладчиков и подключения дополнительных сервисов.

Одним из используемых плагинов Visual Studio Code является Prettier – Code Formatter. Данное расширение позволяет настроить автоматическое форматирование кода при сохранении изменений в файле, включая настройку отступов и вид специальных символов.

Live Server также является плагином VS Code и позволяет увидеть разработчику изменения, внесенные в код, без предварительного перезапуска сервера.

Для разработки графического интерфейса приложения используется HTML. HTML – язык гипертекстовой разметки, позволяющий структурировать информацию для отображения в веб-браузере. Для создания визуально привлекательного интерфейса с помощью стилизации html-элементов используется каскадная таблица стилей CSS.

Для упрощения процесса создания веб-страниц было решено использовать фреймворк Bootstrap 5.2.3. Он представляет собой набор готовых стилей и компонентов, необходимых для ускорения верстки адаптивных страниц.

В качестве браузера для просмотра веб-страниц был выбран Google Chrome, так как он содержит широкий набор инструментов для разработки и отладки сайтов.

Для взаимодействия с сервером, а также обработки событий пользователя и создания интерактивных элементов интерфейса используется скриптовый язык JavaScript.

Серверная часть приложения была реализована на языке программирования C# с помощью фреймворка ASP.NET Core, для хранения данных была выбрана СУБД PostgreSQL.

3 Разработка клиентского приложения

3.1 Разработка структуры клиентского приложения

Клиентское приложение представляет собой многостраничное приложение (MPA). Multi Page Application – веб-приложение, состоящее из нескольких статичных страниц. При этом любое изменение в данных приводит к перезагрузке страницы. Структура MPA обычно включает в себя HTML-страницы, каждая из которых может содержать собственные CSS и JavaScript файлы. Такой подход позволяет разделить функциональность приложения на отдельные модули, что упрощает разработку и поддержку [14].

Структурные компоненты разработанного приложения и связи между ними представлены на диаграмме компонентов на рисунке 23.

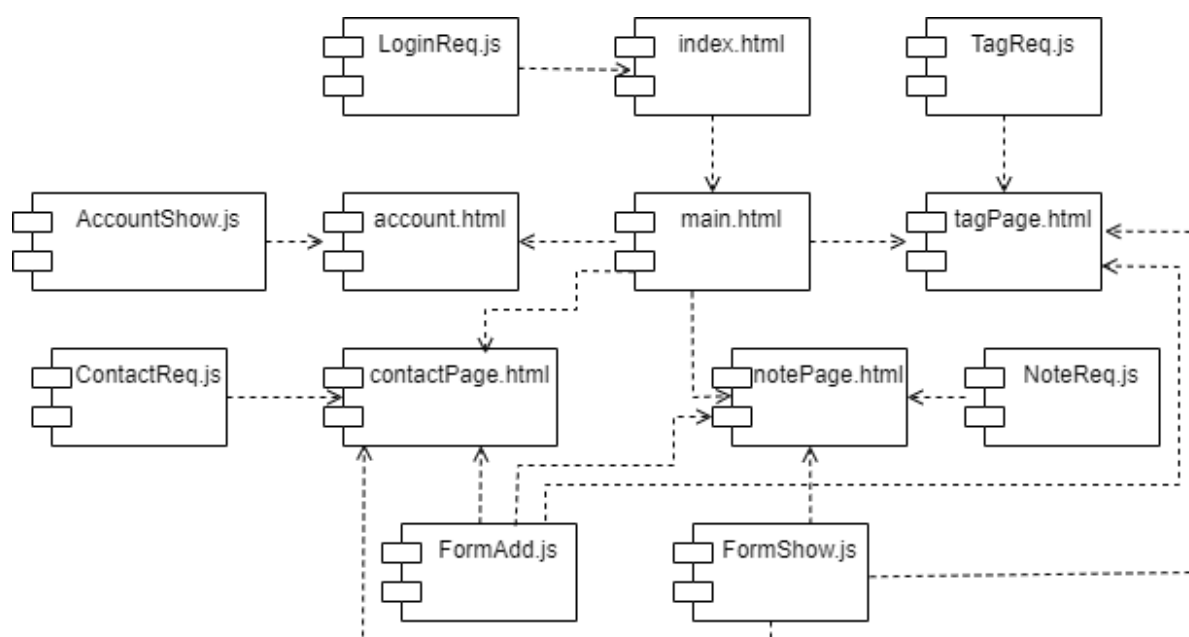


Рисунок 23 – Диаграмма компонентов приложения

Описание компонентов приведено в таблице 5.

Таблица 5 – Описание компонентов клиентского приложения

Компонент	Назначение
index.html	Содержит код реализации отображения начальной страницы приложения
main.html	Содержит код реализации отображения главной страницы приложения

Продолжение таблицы 5

Компонент	Назначение
LoginReq.js	Содержит запросы для отправки данных авторизации и регистрации на сервер
account.html	Содержит код реализации отображения личного кабинета пользователя
AccountShow.js	Содержит функции для отображения и работы с данными личного кабинета и запросы для обмена данными с сервером
contactPage.html	Содержит код реализации отображения страницы контактов
ContactReq.js	Содержит функции для отображения и работы с данными контактов и запросы для обмена данными с сервером
notePage.html	Содержит код реализации отображения страницы заметок
NoteReq.js	Содержит функции для отображения и работы с данными заметок и запросы для обмена данными с сервером
tagPage.html	Содержит код реализации отображения страницы тегов
TagReq.js	Содержит функции для отображения и работы с данными тегов и запросы для обмена данными с сервером
FormAdd.js	Содержит функции и обработчики событий для реализации форм добавления данных
FormShow.js	Содержит функции и обработчики событий для реализации форм отображения данных

HTML-документы имеют определенную структуру, представленную в листинге 1. Основным компонентом является тег, заключаемый в угловые скобки.

Листинг 1 – Структура HTML-документа

```
<!DOCTYPE html>
<html>
<head>
    <title> Заголовок страницы </title>
</head>
<body>
    Содержимое страницы
</body>
</html>
```

Для описания скриптов в заголовке или теге может использоваться тег `<script>`, содержащий сам код или ссылку на расположение программы.

Код функций, выполняемых на веб-страницах располагается в файлах с расширением `.js`.

Листинги кода реализации статических страниц, а также JavaScript файлов представлены в приложении Б.

3.2 Разработка пользовательского интерфейса клиентского приложения

На основе описанных выше требований был разработан пользовательский интерфейс, содержащий следующие страницы:

- начальная страница, предоставляющая возможность регистрации и авторизации пользователя;
- страница для работы с контактами;
- страница для работы с заметками;
- страница для работы с тегами;
- страница личного кабинета пользователя.

Помимо страниц были разработаны формы для отображения данных, а также их редактирования или удаления.

На рисунке 24 представлен интерфейс начальной страницы приложения, которая содержит форму для входа.

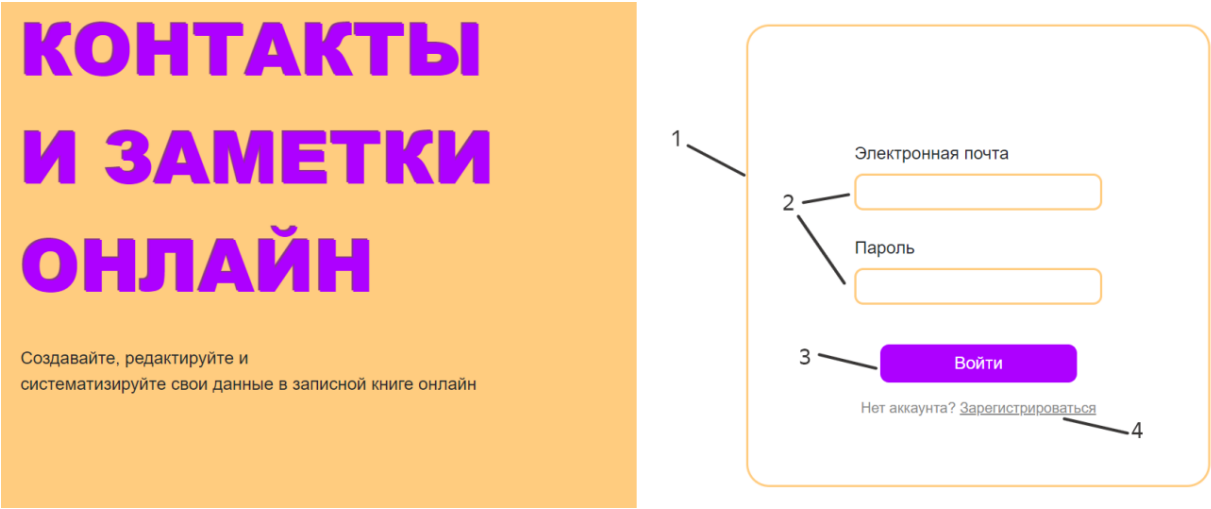


Рисунок 24 – Форма авторизации

Элементы формы, обозначенные цифрами на рисунке 24 описаны в таблице 6.

Таблица 6 – Описание элементов формы авторизации

№ п/п	Имя элемента	Тип элемента	Назначение
1	form	Форма	Отображение формы авторизации

Продолжение таблицы 6

№ п/п	Имя элемента	Тип элемента	Назначение
2	input	Текстовое поле	Поля для ввода электронной почты и пароля
3	input	Кнопка	Кнопка входа в приложение
4	a	Ссылка	Ссылка для перехода на форму регистрации

В случае отсутствия аккаунта, пользователь может зарегистрироваться в приложении при помощи формы, интерфейс которой приведен на рисунке 25.

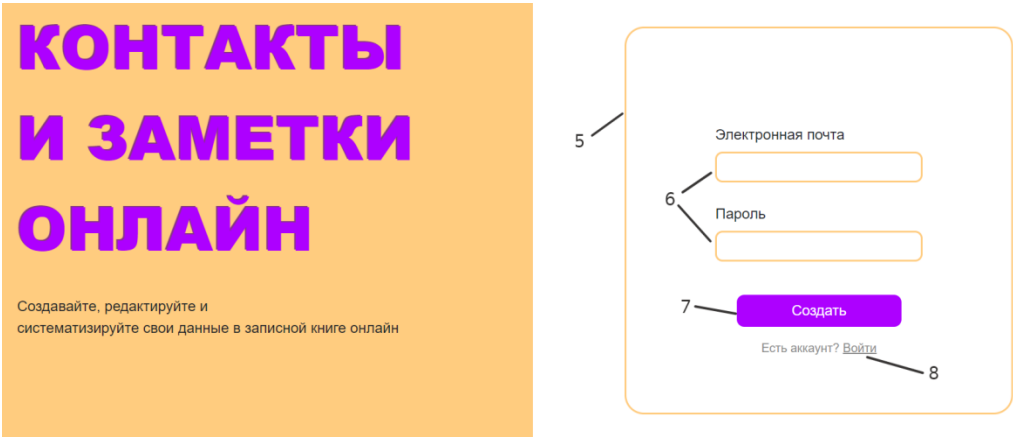


Рисунок 25 – Форма регистрации

Элементы формы, обозначенные цифрами на рисунке 25, описаны в таблице 7.

Таблица 7 – Описание элементов формы регистрации

№ п/п	Имя элемента	Тип элемента	Назначение
5	form	Форма	Отображение формы регистрации
6	input	Текстовое поле	Поля для ввода электронной почты и пароля
7	input	Кнопка	Кнопка регистрации в приложении
8	a	Ссылка	Ссылка для перехода на форму авторизации

Страницы для работы с заметками, контактами и тегами имеют схожую структуру и расположение компонентов интерфейса. В верхней части страницы располагается навигационная панель для обеспечения доступа к разделам приложения. Основное содержание страницы представляют отображаемые информационные объекты. Также каждая страница имеет кнопку для добавления объекта и возможность поиска.

Интерфейс страницы для работы с контактами представлен на рисунке 26.

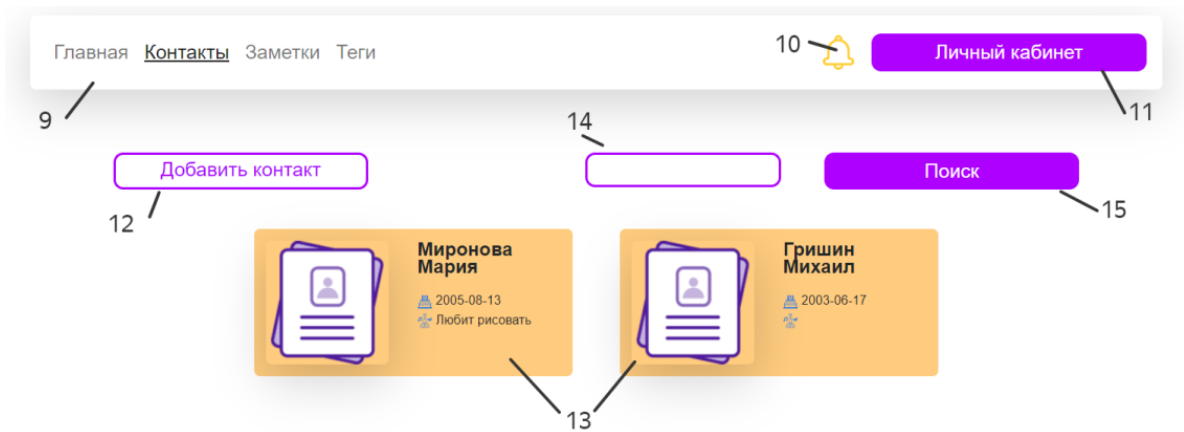


Рисунок 26 – Страница контактов

Элементы интерфейса, обозначенные цифрами на рисунке 26, описаны в таблице 8.

Таблица 8 – Описание элементов страницы контактов

№ п/п	Имя элемента	Тип элемента	Назначение
9	nav	Навигационная панель	Навигация по страницам приложения
10	button	Кнопка	Кнопка для отображения напоминаний приложения
11	input	Кнопка	Кнопка для перехода в личный кабинет
12	input	Кнопка	Добавление нового контакта
13	div	Блок	Отображение краткой информации о контакте
14	input	Текстовое поле	Поле ввода имени контакта для поиска
15	input	Кнопка	Кнопка для поиска контактов

На всех страницах приложения, за исключением начальной, обязательным элементом является навигационная панель. Код навигационной панели, написанный с использованием фреймворка Bootstrap 5, приведен в листинге 2.

Листинг 2 – Код навигационной панели

```
<nav class="navbar navbar-expand-lg bg-white shadow-lg p-1 mb-5 mt-3 bg-body rounded w-75 font-family-sans-serif fs-5">
  <div class="container-fluid">
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link" href="/Main/">Главная</a>
```

```

        </li>
        <li class="nav-item">
          <a class="nav-link"
href="/ContactPage/">Контакты</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/NotePage/">Заметки</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/TagPage/">Теги</a>
        </li>
      </ul>
    </div>
    <button class="time m-3"></button>
    <div class="lk_btn text-center p-1 w-25 h-25">
      <input id="lk_button" class="lk w-75 h-75" style="color:
white; background-color: #ad00ff" value="Личный кабинет"
/>
    </div>
  </div>
</nav>

```

Как можно видеть в листинге 2, навигационная панель состоит из списка ссылок, представляющих собой переходы между страницами приложения, кнопки для показа актуальных уведомлений и кнопки для перехода в личный кабинет пользователя.

Интерфейс личного кабинета пользователя с возможностью редактирования данных представлен на рисунке 27. Поля формы с данными пользователя являются редактируемыми, а для сохранения изменений нужно нажать соответствующую кнопку. Помимо этого пользователь может выйти из приложения при нажатии на кнопку «Выход».

Рисунок 27 – Личный кабинет пользователя

Элементы интерфейса, обозначенные цифрами на рисунке 27, описаны в таблице 9.

Таблица 9 – Описание элементов личного кабинета

№ п/п	Имя элемента	Тип элемента	Назначение
16	form	Форма	Отображение данных о пользователе
17	input	Кнопка	Кнопка для редактирования данных
18	input	Кнопка	Кнопка для выхода из приложения

Для просмотра и редактирования существующих контактов была разработана форма, приведенная на рисунке 28. Открыть ее пользователь может с помощью двойного нажатия на любой контакт, расположенный на странице. Поля формы, являются доступными для изменений.

При необходимости удаления контакта пользователь может нажать на соответствующую кнопку, а для закрытия формы на крестик, расположенный в верхнем правом углу.

Миронова Мария

22 — X

19

Дата рождения
2005-08-13

Адрес проживания
Пенза, ул.Гагарина, 47

Место работы

Другое
Любит рисовать

20 Редактировать

21 Удалить

Рисунок 28 – Форма с подробной информацией о контакте

Элементы формы, обозначенные цифрами на рисунке 28, описаны в таблице 10.

Таблица 10 – Описание элементов формы с информацией о контакте

№ п/п	Имя элемента	Тип элемента	Назначение
19	textarea	Текстовая область	Отображение информации конкретного контакта с возможностью редактирования
20	input	Кнопка	Кнопка для сохранения изменений
21	input	Кнопка	Кнопка для удаления контакта
22	button	Кнопка	Кнопка для закрытия формы

На рисунке 29 представлена форма для работы с существующей заметкой. Пользователю доступны функции редактирования данных заметки или ее удаления.

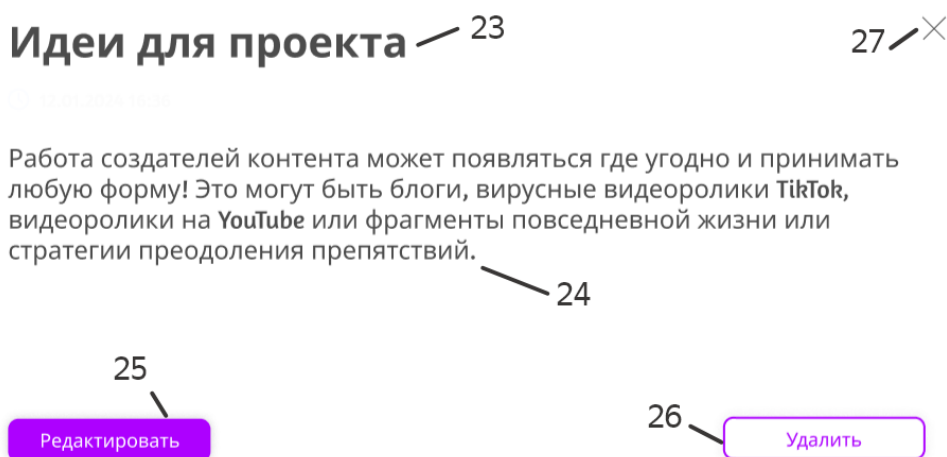


Рисунок 29 – Форма с подробной информацией о заметке

Элементы формы, обозначенные цифрами на рисунке 29, описаны в таблице 11.

Таблица 11 – Описание элементов формы с информацией о заметке

№ п/п	Имя элемента	Тип элемента	Назначение
23	textarea	Текстовая область	Отображение названия заметки с возможностью редактирования
24	textarea	Текстовая область	Отображение описания заметки с возможностью редактирования
25	input	Кнопка	Кнопка для сохранения изменений данных
26	input	Кнопка	Кнопка для удаления заметки
27	button	Кнопка	Кнопка для закрытия формы

Разработка пользовательского интерфейса клиентского приложения электронной контактной книги производилась с использованием языка разметки HTML, каскадной таблицы стилей CSS и языка программирования JavaScript.

3.3 Разработка средств взаимодействия с серверным приложением

Для реализации взаимодействия клиентского приложения с серверным был выбран архитектурный подход REST API.

Функции для отправки запросов и обработки ответов сервера, представлены в таблицах 12 – 15.

Описание функций, реализующих регистрацию и авторизацию пользователя, а также управление данными личного кабинета, представлено в таблице 12.

Таблица 12 – Описание функций для работы с данными пользователя

Функция	URL запроса	HTTP метод	Описание
PostUserLogin()	/User/Login	POST	Авторизация пользователя
PostUserSignup()	/User/Signup	POST	Регистрация пользователя
GetUser()	/User/GetUserData	GET	Получение данных пользователя
EditUser()	/User/ChangeUserData	PUT	Изменение данных пользователя
DeleteUser()	/User/DeleteAccount	DELETE	Удаление аккаунта пользователя

Описание функций, реализующих возможности для работы с контактами пользователя, представлено в таблице 13.

Таблица 13 – Описание функций для работы с контактами

Функция	URL запроса	HTTP метод	Описание
GetContacts()	/Contact/GetAllContacts	GET	Получение всех контактов пользователя
GetContactInfo(id)	/Contact/GetContact/\${id}	GET	Получение данных конкретного контакта
FindByName(word)	/Contact/FindContactByFIO?request=\${word}	POST	Получение контактов, ФИО которых совпадает с word
CreateNewContact()	/Contact/CreateContact	POST	Добавление нового контакта

Продолжение таблицы 13

Функция	URL запроса	HTTP метод	Описание
EditContactInfo(id)	/Contact/UpdateContact/\${id}	PUT	Изменение информации о контакте
DeleteContactInfo(id)	/Contact/DeleteContact/\${id}	DELETE	Удаление контакта

Описание функций, реализующих возможности для работы с заметками пользователя, представлено в таблице 14.

Таблица 14 – Описание функций для работы с заметками

Функция	URL запроса	HTTP метод	Описание
GetNotes()	/Note/GetNotes	GET	Получение всех заметок пользователя
GetNoteInfo(id)	/Note/GetNote/\${id}	GET	Получение данных конкретной заметки
FindByName(word)	/Note/FindNoteByName?request=\${word}	POST	Получение заметок, название которых совпадает с word
CreateNewNote()	/Note/CreateNote	POST	Добавление новой заметки
EditNoteInfo(id)	/Note/UpdateNote/\${id}	PUT	Изменение информации о заметке
DeleteNoteInfo(id)	/Note/DeleteNote/\${id}	DELETE	Удаление заметки

Описание функций, реализующих возможности для работы с тегами пользователя, представлено в таблице 15. Теги бывают двух видов: теги для заметок и теги для контактов.

Таблица 15 – Описание функций для работы с тегами

Функция	URL запроса	HTTP метод	Описание
GetTagsContact()	/Tag/GetTagsWithContacts	GET	Получение всех тегов контактов
GetTagsNote()	/Tag/GetTagsWithNotes	GET	Получение всех тегов заметок
GetTagForContact(id)	/Tag/GetOneTagForContact/\${id}	GET	Получение тегов конкретного контакта
GetTagForNote(id)	/Tag/GetOneTagForNote/\${id}	GET	Получение тегов конкретной заметки
CreateTagContacts()	/Tag/CreateTagForContacts	POST	Добавление нового тега для контактов
CreateTagNotes()	/Tag/CreateTagForNotes	POST	Добавление нового тега для заметок

Продолжение таблицы 15

Функция	URL запроса	HTTP метод	Описание
EditTagForContact(id)	/Tag/UpdateTagForContacts/{id}	PUT	Изменение информации о теге контактов
EditTagForNote(id)	/Tag/UpdateTagForNotes/{id}	PUT	Изменение информации о теге заметок
DeleteTagForContact(id)	/Tag/DeleteTagForContacts/{id}	DELETE	Удаление тега контактов
DeleteTagForNote(id)	/Tag/DeleteTagForNotes/{id}	DELETE	Удаление тега заметок

Для наглядного представления взаимодействия клиентской и серверной частей используется диаграмма последовательности. Объекты взаимодействия обозначаются прямоугольниками с соответствующими названиями. Вертикальные прямоугольники отражают деятельность объекта, а стрелки показывают обмен сообщениями между объектами. На рисунке 30 приведен порядок обмена сообщениями для процесса авторизации пользователя в приложении.

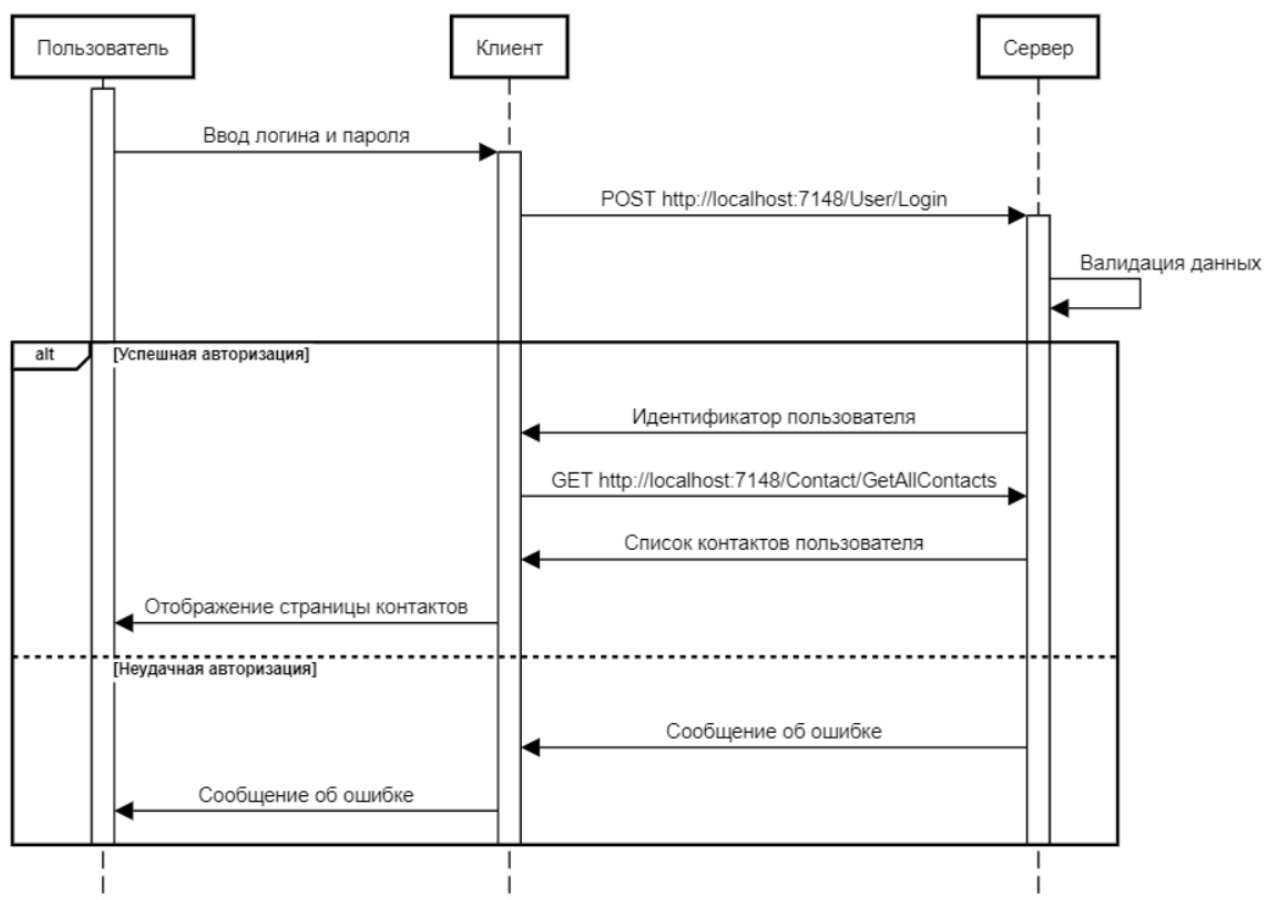


Рисунок 30 – Диаграмма последовательности процесса авторизации

В случае успешной авторизации пользователя происходит переход на страницу контактов, где отображаются существующие контакты. Если авторизация неуспешна, то на экране отображается сообщение об ошибке.

Для получения информации с сервера в JavaScript используется метод `fetch()`. Он поддерживается всеми современными браузерами.

При нажатии на блок с отображением контакта пользователю открывается форма с подробной информацией о контакте. Функция `GetContactInfo(id)` получает информацию с сервера с помощью метода `fetch()`. Код данной функции приведен в листинге 3.

Листинг 3 – Код функции `GetContactInfo(id)`

```
async function GetContactInfo(id) {
    const response = await fetch(`/Contact/GetContact/${id}`, {
        method: "GET",
        headers: { "Accept": "application/json" }
    });
    if (response.ok) {
        const contact = await response.json();
        console.log(contact);
        const form = document.forms["contactInfo"];
        form.elements["fio"].value = contact.contact.fio;
        form.elements["cont_birth"].value = con-
        tact.contact.cont_birth;
        form.elements["address"].value = contact.contact.address;
        form.elements["cont_work"].value = contact.contact.cont_work;
        form.elements["other"].value = contact.contact.other;
    }
}
```

Метод `Fetch()` имеет два параметра: URL для отправки запроса и дополнительные параметры: метод, заголовки и т. д. Если запрос прошел успешно, то в свойстве ответа `response.ok` можно получить тело ответа [15]. В случае с функцией `GetContactInfo(id)` телом является подробная информация о контакте.

Листинги кода реализации функций для отправки запросов и получения ответов сервера приведены в приложении Б.

4 Контроль качества программного обеспечения клиентского приложения

4.1 Планирование процесса контроля качества программного обеспечения

4.1.1 Обоснование выбора модели тестирования

Тестирование – важная часть процесса разработки, направленная на обеспечение качества, надежности и производительности разработанного программного обеспечения.

Основными задачами тестирования являются:

- выявление и устранение несоответствий, ошибок и потенциальных проблем;
- проверка на соответствие программного обеспечения требованиям, предъявляемым к нему;
- подтверждение совместимости приложения с различными устройствами и браузерами.

Выделяют два основных вида тестирования: функциональное и нефункциональное.

Функциональное тестирование направлено на проверку возможностей приложения и соответствие его поведения заданным требованиям. Основной целью функционального тестирования является подтверждение корректной работы приложения [16].

Функциональное тестирование подразделяется на следующие виды:

- тестирование черного ящика;
- системное тестирование;
- интеграционное тестирование – проверка работоспособности интегрированных, то есть уже соединенных в целое приложение модулей;
- модульное тестирование;
- приемочное тестирование.

Тестирование черного ящика предполагает проверку программного обеспечения без знания ее внутренней структуры. Этот метод позволяет тестирующему сконцентрироваться на внешнем поведении системы, взаимодействуя с ней как пользователь [17].

При системном тестировании все компоненты приложения тестируются как единое программное средство. Основной целью является проверка корректности работы приложения при обработке различных ситуаций. Данный вид тестирования позволяет выявить и решить проблемы с совместимостью и стабильностью системы [18].

Модульное тестирование предполагает тестирование отдельных компонентов или блоков программного средства. Данный вид тестирования проводится на ранних этапах разработки приложения и позволяет сократить общие затраты и время разработки [19].

Приемочное тестирование является подтверждением того, что система готова к использованию. Данный вид тестирования является заключительным этапом функционального тестирования и обычно проводится конечными пользователями.

Нефункциональное тестирование позволяет оценить – насколько хорошо система работает. Целью является подтверждение того, что приложение соответствует требованиям к производительности, безопасности, удобству использования и качеству.

Нефункциональное тестирование делится на следующие виды:

- тестирование производительности – проверка работоспособности системы при различных нагрузках;
- тестирование надежности – проверка устойчивости системы к сбоям;
- тестирование удобства использования подразумевает оценку пользовательского интерфейса на доступность и простоту использования;
- тестирование адаптивности – проверка способности приложения подстраиваться под экраны с различными разрешениями [20].

Для тестирования клиентского приложения электронной контактной книги было проведено как функциональное, так и нефункциональное тестирование.

В качестве вида функционального тестирования был выбран метод черного ящика, так как необходимо проверить корректно ли клиентское приложение выполняет свои функциональные возможности с точки зрения пользователя.

В качестве вида нефункционального тестирования было выбрано тестирование адаптивности web-приложения. Для разрабатываемого приложения электронной контактной книги важно, чтобы на экране устройства помещались все элементы страниц и сохранялись функциональные возможности вне зависимости от разрешения экрана.

4.1.2 Обоснование выбора метрик программного кода

Метрики программного кода представляют собой набор оценок, необходимый для формирования представления о коде разрабатываемого приложения.

Метрики кода принято разделять на три группы:

- количественные метрики – оценки, связанные с размером программы;
- метрики сложности потока управления программы – оценки основываются на анализе управляющего графа программы, одной из таких метрик является цикломатическая сложность;
- метрики сложности потока данных – оценки использования и размещения данных в программе [21].

Для оценки качества программного кода были выбраны метрика Холстеда и метрика SLOC.

Метрика Холстеда является количественной метрикой и вычисляется на основании анализа числа строк и синтаксических конструкций исходного программного кода. Основу данной метрики составляют следующие характеристики:

- n_1 – количество уникальных операторов программы;
- n_2 – количество уникальных операндов программы;

- N1 – общее количество операторов в программе;
- N2 – общее количество операндов в программе [22].

На основе этих характеристик рассчитывается длина, словарь и объем программы.

Метрика SLOC используется для подсчета количества строк в тексте программного кода и помогает оценить трудозатраты на разработку. Различают физические и логические строки кода. К физическим строкам относятся все строки текстового файла без учета пустых строк. Логические строки кода представляют собой количество команд (операторов) программы.

4.2 Тестирование программных средств

4.2.1 Функциональное тестирование клиентского приложения

С целью проверки корректности работы функциональных возможностей системы было проведено функциональное тестирование приложения методом черного ящика.

План и результаты тестирования основных функциональных возможностей клиентского приложения приведены в таблице 16 и приложении В соответственно.

Таблица 16 – План тестирования клиентского приложения

№ п/п	Состав теста	Результат
1	Проверка регистрации с некорректным адресом электронной почты	Вывод сообщения об ошибке в адресе электронной почты (рисунок В.1)
2	Проверка регистрации пользователя с существующим аккаунтом	Вывод сообщения о существовании аккаунта (рисунок В.2)
3	Проверка авторизации с неверным паролем	Вывод сообщения об ошибке в пароле (рисунок В.3)
4	Проверка авторизации пользователя с существующим аккаунтом	Отображение страницы контактов пользователя (рисунок В.4)
5	Осуществление выхода из приложения	Отображение начальной страницы приложения (рисунок В.5)
6	Редактирование данных в личном кабинете пользователя	Отображение измененных данных личного кабинета (рисунки В.6 и В.7)

Продолжение таблицы 16

№ п/п	Состав теста	Результат
7	Добавление нового контакта	Отображение нового контакта на странице контактов (рисунок В.8)
8	Добавление нового тега	Отображение нового тега на странице тегов (рисунки В.9 и В.10)
9	Добавление новой заметки	Отображение новой заметки на странице заметок (рисунки В.11 и В.12)
10	Поиск контакта по названию	Отображение найденных контактов (рисунок В.13)
11	Изменение данных контакта	Отображение измененных данных на странице контактов (рисунки В.14 и В.15)
12	Удаление контакта	Удаленный контакт не отображается на странице контактов (рисунок В.16)
13	Удаление заметки	Удаленная заметка не отображается на странице заметок (рисунок В.17)

По результатам выполнения тестирования приложения электронной контактной книги, несоответствий и ошибок не обнаружено, соответственно, приложение работает корректно.

4.2.2 Тестирование адаптивности интерфейса

В качестве нефункционального тестирования клиентского приложения была проведена проверка адаптивности.

Адаптивность web-приложения – способность приложения, обеспечивающая правильное отображение и функционирование страниц сайта на различных типах устройств. Для адаптации под разрешения экранов используются специальные точки – брейк-поинты, которые определяют поведение макета страниц в зависимости от размеров устройств.

Для тестирования были выбраны следующие разрешения устройств:

- 375*667 (смартфон);
- 768*1024 (планшет);
- 1440*1024 (ноутбук).

Результаты тестирования адаптивности страниц клиентского приложения приведены в таблице 17. Страницы приложения, выбранные для тестирования, приведены в приложении Г.

Таблица 17 – Результаты тестирования адаптивности

Критерий	Тип устройства		
	Смартфон (375*667)	Планшет (768*1024)	Ноутбук (1440*1024)
Стиль контента	Контент отображается последовательно, стиль и размер текста соответствует типам устройств.		
Выравнивание	Выравнивание основного контента по центру на всех типах устройств.		
Согласованность изображений	Изображения хорошо видны, друг на друга не накладываются, текстом не перекрываются.		
Меню	Разворачивается и сворачивается на всех типах устройств без потери своей функциональности и визуальных качеств.		
Ориентация устройства	Дизайн сайта адаптирован с учетом ориентации устройства.		Для данного устройства данный критерий не важен.

Результаты тестирования адаптивности показали правильное функционирование и отображение интерфейса приложения на различных типах устройств.

4.2.3 Оценка времени выполнения задач пользователем

Для оценки скорости работы пользователя с системой используется метод GOMS (Goals, Operators, Methods, Selection Rules – цели, операторы, методы и правила их выбора). Данный метод предполагает, что все действия пользователя можно разложить на составляющие – более простые действия. Получив список этих простых действий, можно замерить время их выполнения. Благодаря статистически полученным данным о среднем времени выполнения задач, оценивается эффективность выполнения действий пользователей в конкретном интерфейсе [23].

Разбиение задач на составляющие и длительность каждой подзадачи представлены в таблице 18.

Таблица 18 – Характеристики подзадач пользователя

Обозначение	Описание действия	Время
К	Нажатие на клавишу клавиатуры	0,28 сек
М	Нажатие на кнопку мыши	0,1 сек
П	Перемещение курсора мыши	1,1 сек
В	Взятие или бросание мыши	0,4 сек
Д	Продолжительность выбора действия	1,2 сек
Р	Время реакции системы	от 0,1 сек

Общее время выполнения задачи представляет собой сумму времени выполнения всех подзадач.

Метод GOMS был применен для оценки времени, затраченного пользователем для добавления новой заметки. Составляющие этой задачи приведены в таблице 19.

Таблица 19 – Действия пользователя для добавления новой заметки

Тип действия	Продолжительность	Описание
Д	1,2	Пользователь анализирует задачу и создает алгоритм ее решения
П	1,1	Перемещение курсора на кнопку добавления заметки
М	0,1	Нажатие на кнопку добавления
Р	0,1	Открытие формы для добавления заметки
Д	1,2	Открылась форма и пользователю необходимо ее изучить
П	1,1	Перемещение курсора к полю для ввода названия заметки
М	0,1	Выбор поля для ввода
Д	1,2	Пользователь придумывает название заметки
В	0,4	Перенос руки с мыши на клавиатуру
К * 14	3,92	Ввод названия заметки «Купить подарок»
В	0,4	Перенос руки с клавиатуры на мышь
П	1,1	Перемещение курсора к полю для ввода описания заметки
М	0,1	Выбор поля для ввода
Д	1,2	Пользователь придумывает описание заметки
В	0,4	Перенос руки с мыши на клавиатуру
К*17	4,76	Ввод описания заметки «блэкнот и шоколад»
В	0,4	Перенос руки с клавиатуры на мышь
П	1,1	Перемещение курсора к кнопке сохранения
М	0,1	Нажатие на кнопку сохранения
Р	0,3	Закрытие формы добавления, отображение созданной заметки на странице

Общее время выполнения задачи определяется по формуле:

$$T = T_k + T_m + T_n + T_v + T_d + T_p,$$

где T_k – общее время, затраченное на нажатие клавиши клавиатуры;

T_m – общее время, затраченное на нажатие кнопки мыши;

T_n – общее время, затраченное на перемещение курсора мыши;

T_v – общее время, затраченное на взятие или бросание мыши;

T_d – общее время, затраченное пользователем на выбор действий;

T_p – общее время реакций системы.

В итоге, для добавления новой заметки пользователю потребуется 20,5 секунд.

Как можно видеть в таблице 19 пользователь при заполнении полей несколько раз переносит руку с мыши на клавиатуру и обратно, что увеличивает время выполнения операции. Для увеличения скорости заполнения полей можно задуматься над возможностью переключения между полями формы с помощью клавиатуры.

Таким образом, метод GOMS позволяет не только произвести оценку времени выполнения конкретной задачи, но и проанализировать порядок действий пользователя. Результаты данного анализа могут быть применены для дальнейшего развития и расширения функциональных возможностей приложения.

4.3 Оценка качества программного кода на основе метрик

4.3.1 Метрика SLOC

Для расчета данной метрики рассматривается исходный код функций `PostUserLogin()` и `PostUserSignup()`, реализующих регистрацию и авторизацию пользователя соответственно. Код данных функций представлен в листинге 4.

Листинг 4 – Код функций `PostUserLogin()` и `PostUserSignup()`

```
async function PostUserLogin() {
  const response = await fetch("/User/Login", {
    method: "POST",
    body: new FormData(document.forms.loginForm), });
  console.log(response1.status);
  if (response.ok) {
    const user = await response.json();
    window.location.href = `https://localhost:7148/Main`;
  } else {
    const errorData = await response.json();
    if (errorData) {
      // Выводим ошибки пользователю
      document.getElementById("message_log").innerHTML = errorData + "<br/>";
      console.error("Ошибки от сервера:", errorData);
    }
  }
}
```

```

        } else {
            alert("Ошибка сервера. Попробуйте позже.");
        }
    }
}
}
async function PostUserSignup() {
    const responsel = await fetch("/User/Signup", {
        method: "POST",
        body: new FormData(document.forms.registerForm),
    });
    console.log(responsel.status);
    if (responsel.ok) {
        const user = await responsel.json();
        window.location.href = `https://localhost:7148/Main`;
    } else {
        const errorData = await responsel.json();
        if (errorData) {
            console.error("Ошибки от сервера:", errorData);
            // Выводим ошибки пользователю
            document.getElementById("message_reg").innerHTML = errorData + "<br/>";
        } else {
            alert("Ошибка сервера. Попробуйте позже.");
        }
    }
}
}

```

Основой метрики SLOC является количество функциональных и логических строк кода. Данный код содержит 42 физические строки и 23 логические строки, а также 2 комментария, что является нормой для функций.

4.3.2 Метрика Холстеда

Для оценки сложности исходного кода была использована метрика Холстеда. Она позволяет рассчитать длину, словарь, объем программы и ее сложность.

В данном приложении расчет метрики Холстеда приведен для функции получения данных личного кабинета пользователя. Код данной функции приведен в листинге 5.

Листинг 5 – Код функции получения данных пользователя

```

async function GetUser() {
    const response = await fetch('/User/GetUserData', {
        method: "GET",
        headers: { "Accept": "application/json" }
    });
    if (response.ok) {
        const user = await response.json();
        console.log(user);
    }
}

```

```

const form = document.forms["accountForm"];
form.elements["user_name"].value = user.user_name;
form.elements["user_birth"].value = user.user_birth;
form.elements["user_pn"].value = user.user_pn;
form.elements["user_email"].value = user.user_email;
form.elements["user_password"].value = user.user_password;
}
}

```

Для расчета метрики необходимо узнать количество операторов и операндов исходного кода.

Операторами являются знаки операций, имена функций, а также символы-разделители. Операторы функции и число их вхождений приведены в таблице 20.

Таблица 20 – Операторы функции получения данных

Операторы	Число вхождений
[]	5
{}	3
:	2
.	18
;	9
=	8
if	1
ok	1
value	5
GetUser()	1
console.log()	1
json()	1
fetch	1
await	2
document	1
elements	5
forms	1
n1 = 17	N1 = 65

Операндами являются объекты программы, над которыми выполняются действия.

Операнды программы и число их вхождений приведены в таблице 21.

Таблица 21 – Операнды функции получения данных

Операнды	Число вхождений
response	3

Продолжение таблицы 21

Операнды	Число вхождений
'/User/GetUserData'	1
form	6
user	7
user_name	2
user_birth	2
user_pn	2
user_email	2
user_password	2
n2 = 9	N2 = 27

Длина программы определяется по формуле:

$$N1 + N2 = 65 + 27 = 92,$$

где $N1$ – общее число операторов программы;

$N2$ – общее число операндов.

Словарь программы рассчитывается по формуле:

$$n = n1 + n2 = 17 + 9 = 26,$$

где $n1$ – число уникальных операторов программы;

$n2$ – число уникальных операндов программы.

Сложность программы рассчитывается по формуле:

$$HD = \left(\frac{n1}{2}\right) * \left(\frac{N2}{n2}\right) = \left(\frac{17}{2}\right) * \left(\frac{27}{9}\right) = 25.5.$$

С помощью длины и словаря программы можно рассчитать объем программы по формуле:

$$V = N * \log_2(n) = 92 * 4.7 = 432.4.$$

В результате подстановки значений в формулу получаем объем программы $V = 432.4$

Результаты расчета показателей соответствуют допустимым значениям по данным из книги Холстеда [24].

Заключение

В ходе выполнения выпускной квалификационной работы было разработано клиентское приложение электронной контактной книги с возможностью ведения заметок.

Проведен анализ предметной области и сравнительный анализ аналогичных приложений. На основе анализа были разработаны требования к реализуемому приложению. Для описания функциональных возможностей приложения была составлена диаграмма вариантов использования.

С помощью программного средства для управления проектами Microsoft Project 2007 было проведено планирование и расчет бюджета проекта. В итоге финансовые затраты составили 74 395,00 руб.

На этапе проектирования были разработаны прототипы интерфейса основных страниц приложений с использованием графического редактора Figma.

Разработка приложения проводилась в редакторе программного кода Visual Studio Code на языке JavaScript. В качестве операционной системы использовалась Windows 10 Pro.

Заключительным этапом разработки стало проведение тестирования. В результате тестирования ошибок выявлено не было, все тесты пройдены успешно.

На данный момент реализован прототип приложения, реализующий основные функциональные возможности, включающие работу с контактами, заметками и тегами. В дальнейшем планируется добавление возможности загрузки изображений для контактов, экспорта данных в текстовый файл, а также усовершенствование панели напоминаний.

Список использованных источников

1. Балашова, И. Ю. Современные информационные технологии в проектировании программных систем и комплексов : учеб. пособие / И. Ю. Балашова, Д. В. Такташкин; под ред. П. П. Макарычева. – Пенза : Изд-во ПГУ, 2019. – 106 с.
2. Google Keep – ваш универсальный помощник:[электронный ресурс]. URL: <https://www.google.ru/keep/> (дата обращения: 15.05.2024)
3. Яндекс Заметки: [электронный ресурс]. URL: https://disk.yandex.ru/notes/345924994_1_312 (дата обращения: 15.05.2024)
4. Simplenote [электронный ресурс]. URL: <https://simplenote.com/> (дата обращения: 15.05.2024)
5. Google Контакты – обзор сервиса: [электронный ресурс]. URL: <https://startpack.ru/application/google-contacts> (дата обращения: 16.05.2024)
6. Simple & Free Cross-Platform Contact Manager App for Business: [электронный ресурс]. URL: <https://www.contactbook.app/> (дата обращения: 16.05.2024)
7. Диаграмма Ганта: для чего нужна, как построить: [электронный ресурс]. URL: <https://digital-academy.ru/blog/gantt-chart> (дата обращения: 19.05.2024)
8. Шкрыль А.А. MS Project 2007: современное управление проектами – СПб.: БХВ-Петербург, 2007. – 256 с.
9. НОУ ИНТУИТ | Нотация и семантика языка UML: [электронный ресурс]. URL: <https://intuit.ru/studies/courses/32/32/lecture/1004?page=2> (дата обращения 19.05.2024)
10. Архитектура «клиент-сервер»: суть, виды, преимущества [электронный ресурс]. URL: <https://itspectr.ru/arhitektura-klient-server-osnovy-i-princzipy/> (дата обращение 19.05.2024)

11. Понимание REST API: основные понятия и принципы работы: [электронный ресурс]. URL: <https://yandex.cloud/ru/docs/glossary/rest-api> (дата обращения: 21.05.2024)
12. Простое руководство к диаграммам развертывания UML: [электронный ресурс]. URL: <https://creately.com/blog/ru/uncategorized-ru/учебное-пособие-по-диаграмме-развёрт/> (дата обращения: 22.05.2024)
13. Что такое Visual Studio Code и зачем он нужен: [электронный ресурс]. URL: <https://workspace.ru/tools/environment/visual-studio-code/> (дата обращения: 23.05.2024)
14. Разработка многостраничных сайтов на JavaScript: [электронный ресурс]. URL: <https://sky.pro/wiki/javascript/razrabotka-mnogostranichnyh-sajtov-na-javascript/> (дата обращения 23.05.2024)
15. Fetch: [электронный ресурс]. URL: <https://learn.javascript.ru/fetch> (дата обращения: 28.05.2024)
16. Различные виды тестирования программного обеспечения: [электронный ресурс]. URL: <https://appmaster.io/ru/blog/vidy-testirovaniia-programmnogo-obespecheniia> (дата обращения: 29.05.2024)
17. Виды функционального и нефункционального программирования: [электронный ресурс]. URL: <https://sky.pro/wiki/profession/vidy-funkcionalnogo-i-nefunkcionalnogo-testirovaniya/> (дата обращения: 29.05.2024)
18. Какие бывают этапы и виды тестирования: подробный разбор: [электронный ресурс]. URL: <https://ru.hexlet.io/blog/posts/vidy-testirovaniya> (дата обращения: 29.05.2024)
19. Виды и типы тестирования: [электронный ресурс]. URL: <https://testengineer.ru/vidy-tipy-testirovaniya/> (дата обращения: 29.05.2024)
20. Нефункциональное тестирование Frontend части: [электронный ресурс]. URL: <https://byndyusoft.com/blogs/nonfunctionalfrontendtesting> (дата обращения: 29.05.2024)
21. Новичков А., Шамрай А., Черников А. Метрики кода и их практическая реализация в Subversion и ClearCase. Часть 1 – метрики: [Электронный

ресурс]. URL:

<https://web.archive.org/web/20110913071425/http://www.rusarticles.com/programmy-statya/metriki-koda-i-ix-prakticheskaya-realizaciya-v-subversion-i-clearcase-chast-1-metriki-603328.html> (дата обращения: 02.06.2024)

22. НОУ ИНТУИТ | Лекция 1: Общие сведения о программном обеспечении: [электронный ресурс]. URL:

https://intuit.ru/studies/professional_skill_improvements/17031/courses/874/lecture/14289?page=2 (дата обращения: 02.06.2024)

23. Головач В.В. Дизайн пользовательского интерфейса. [Электронный ресурс]. URL: https://studylib.ru/doc/6221558/dizajn-pol._zovatel._skogo-interfejsa.-golovach-v (дата обращения: 03.06.2024)

24. Холстед, М. Начала науки о программах / Холстед М – М.: Финансы и статистика, 1981. – 128 с.

Приложение А
(справочное)
Глоссарий

Клиентская часть приложения – часть web-приложения, с которой взаимодействует пользователь.

API (Application Programming Interface) – набор протоколов и правил, описывающих способы взаимодействия клиента и сервера.

REST (Representational State Transfer) – архитектурный стиль, устанавливающий ограничения для API. REST API поддерживает стандартные операции для работы с данными: GET, POST, PUT и DELETE.

URL (Uniform Resource Locator) – адрес ресурса в сети Интернет.

Web-приложение – программа, для запуска которой необходим браузер.

Информационный объект – отображение некоторой сущности, характеризующейся совокупностью специфических информационных свойств. В случае электронной контактной книги информационными объектами являются контакты, заметки, теги и напоминания.

Заметка – информационный объект приложения, представляющий собой краткое сообщение о факте или событии.

Контакт – информационный объект приложения, представляющий собой структуру для хранения контактных данных людей.

Тег – информационный объект приложения, позволяющий выделять категории заметок и контактов.

Напоминание – информационный объект приложения, который напоминает о событии, хранящемся в электронной контактной книге.

Markdown – язык разметки текстовых документов.

Фреймворк – набор готовых инструментов, необходимый для ускорения процесса разработки.

UML (Unified Modeling Language) – графический язык для моделирования в области разработки программного обеспечения.

MPA (Multi Page Application) – тип структуры веб-приложения, при котором само приложения состоит из нескольких страниц.

Приложение Б
(рекомендуемое)
Код приложения

Листинг Б.1 – Файл index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstr
ap.min.css"
    rel="stylesheet"
    integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOMLASjC"
    crossorigin="anonymous"
  />
  <link rel="stylesheet" href="style.css" />
  <title>Контакты и заметки онлайн</title>
</head>
<body>
  <div class="container-fluid">
    <div class="row">
      <div class="col-12 col-sm-12 col-xl-6 title">
        <div class="name">контакты и заметки онлайн</div>
        <div class="text">
          Создавайте, редактируйте и <br />
          систематизируйте свои данные в записной книге онлайн
        </div>
      </div>
      <div class="col-12 col-sm-12 col-xl-6">
        <!--
Login form----->
        <form class="login" name="loginForm" id="login_form" method="post">
          <p id="message_log" class="text-center" style="color: fire-
brick"></p>
          <div class="inputs">
            <label class="form-label" for="l_email">Электронная почта</label>
            <input type="email"
              name="user_email"
              id="l_email"
              class="form-control"
              required />
          </div>
          <div class="inputs">
            <label class="form-label" for="l_password">Пароль</label>
            <input type="password"
              name="user_password"
              id="l_password"
              class="form-control"
              required />
          </div>
          <div class="btn mb-3 mt-3">
            <input id="login_button"
              class="enter_btn w-100 h-100"
              type="submit"
              style="color: white; background-color: #ad00ff"
              value="Войти" />
          </div>
        </form>
      </div>
    </div>
  </div>

```

```

<!-- Register buttons -->
<div class="text-center">
<div id="p_text" style="color: #888888">
Нет аккаунта?
<a id="reg_link"
href="#"
onclick="reloadRegister(); docu-
ment.getElementById('register_form').style.display=''; docu-
ment.getElementById('login_form').style.display='none';"
style="color: #888888">Зарегистрироваться</a>
</div>
</div>
</form>
<!------->
Register form----->
<form class="register"
name="registerForm"
style="display: none"
id="register_form"
method="post">
<p id="message_reg" class="text-center" style="color: fire-
brick"></p>
<div class="inputs">
<label class="form-label" for="email">Электронная почта</label>
<input type="email"
name="user_email"
id="r_email"
class="form-control"
required />
</div>
<div class="inputs">
<label class="form-label" for="password">Пароль</label>
<input type="password"
name="user_password"
id="r_password"зхъ\

class="form-control"
required />
</div>
<div class="btn mb-3 mt-3">
<input id="register_button"
class="enter_btn"
type="submit"
value="Создать"
style="color: white; background-color: #ad00ff" />
</div>
<!-- Login buttons -->
<div class="text-center">
<div id="p_text" style="color: #888888">
Есть аккаунт?
<a id="reg_link"
href="#"
onclick="reloadLogin(); docu-
ment.getElementById('register_form').style.display='none'; docu-
ment.getElementById('login_form').style.display='';"
style="color: #888888">Войти</a>

```

```

</div>
</div>
</form>
</div>
</div>
<h1 style="padding: 2%; padding-left: 5%">Создавайте элементы</h1>
<div class="row m-3 justify-content-around text-left">
<div class="elems m-2 col-12 col-sm-4 col-xl-3">

<div>
<p>Заметки</p>
Храните информацию о встречах, рабочих моментах и других важных
деталях.
</div>
</div>
<div class="elems m-2 col-12 col-sm-4 col-xl-3">

<div>
<p>Контакты</p>
Больше не придется искать информацию в разных источниках - все
важные сведения хранятся в одном месте.
</div>
</div>
<div class="elems m-2 col-12 col-sm-4 col-xl-3">

<div>
<p>Теги</p>
Благодаря систематизации данных Вы с легкостью можете получить
доступ к информации по интересующей Вас теме.
</div>
</div>
</div>
<h1 style="padding: 2% 4% 2% 2%; text-align: right">
Множество полезных функций
</h1>
<div class="row justify-content-around text-left">
<div class="parts p-0 col-12 col-sm-6 col-xl-5">

<div class="text_part">
<p>Просмотр дней рождений</p>
Больше не стоит беспокоиться, что Вы забудете о важной дате. Все дни ро-
ждения Ваших контактов собраны в одном месте.
</div>
</div>
<div class="parts p-0 col-12 col-sm-6 col-xl-5">

<div class="text_part">
<p>Поиск по вашим записям</p>

Даже при огромном количестве данных легко найти нужную запись. Для
этого нужно просто воспользоваться поиском.
</div>
</div>
<div class="parts p-0 col-12 col-sm-6 col-xl-5">

<div class="text_part">

```

<p>Уведомления о событиях</p>Разгрузите голову! Вы можете записать все свои планы в заметки, а мы своевременно напомним Вам о них.

</div>

</div>

<div class="parts p-0 col-12 col-sm-6 col-xl-5">

<div class="text_part">

<p>Цветовая палитра тегов</p>

Наглядное представление того, по каким категориям распределены данные. Все для быстрого поиска.

</div>

</div>

</div>

<div class="row final col-12 col-sm-12 col-xl-12 text-center">

<div class="text_fin"

style="

color: #ad00ff;

text-transform: uppercase;

font-size: 200%;

font-weight: 400;

">

Начните вести свою электронную книгу прямо сейчас!

</div>

<div class="btn return m-3">

<a style="text-decoration: none;

color: white;

background-color: #ad00ff;

border-radius: 10px;

font-family: sans-serif;

"

href="#top">Начать

</div>

</div>

</div>

<script>

const loginForm = document.querySelector("form.login");

const registerForm = document.querySelector("form.register");

document

.getElementById("login_button")

.addEventListener("click", function (event) {

event.preventDefault();

var l_email = document.getElementById('l_email').value;

var l_password = document.getElementById('l_password').value;

if (l_email == "" || l_password == "") {

document.getElementById('message_log').innerHTML = "Поля обязательны для заполнения!";

return;

}

PostUserLogin();

});

document

.getElementById("register_button")

.addEventListener("click", function (event) {

event.preventDefault();

var r_email = document.getElementById('r_email').value;

var r_password = document.getElementById('r_password').value;

if (r_email == "" || r_password == "") {

```

        document.getElementById('message_reg').innerHTML = "Поля
        обязательны для заполнения!";
        return;
    }
    console.log(r_email, r_password);
    PostUserSignup();
    });

    function reloadLogin() {
        document.getElementById('message_log').innerHTML = "";
        document.getElementById('l_email').value = '';
        document.getElementById('l_password').value = '';
    }
    function reloadRegister() {
        document.getElementById('message_reg').innerHTML = "";
        document.getElementById('r_email').value = '';
        document.getElementById('r_password').value = '';
    }
    </script>
    <script src="/js/LoginReq.js"></script>
</body>
</html>

```

Листинг Б.2 – Файл LoginReq.js

```

async function PostUserLogin() {
    const response = await fetch("/User/Login", {
        method: "POST",
        body: new FormData(document.forms.loginForm),
    });
    console.log(response.status);
    if (response.ok) {
        const user = await response.json();
        console.log(user);
        window.location.href = `https://localhost:7148/Main`;
    } else {
        const errorData = await response.json();
        if (errorData) {
            // Выводим ошибки пользователю
            document.getElementById("message_log").innerHTML = er-
            rorData + "<br/>";
            // Выводим ошибки в консоль
            console.error("Ошибки от сервера:", errorData);
        } else {
            alert("Ошибка сервера. Попробуйте позже.");
        }
    }
}

async function PostUserSignup() {
    const responsel = await fetch("/User/Signup", {
        method: "POST",
        body: new FormData(document.forms.registerForm),
    });
    console.log(responsel.status);
    if (responsel.ok) {
        const user = await responsel.json();
        console.log(user);
        window.location.href = `https://localhost:7148/Main`;
    }
}

```

```

    } else {
        const errorData = await response1.json();
        if (errorData) {
            // Выводим ошибки в консоль
            console.error("Ошибки от сервера:", errorData);
            // Выводим ошибки пользователю
            document.getElementById("message_reg").innerHTML = errorData + "<br/>";
        } else {
            alert("Ошибка сервера. Попробуйте позже.");
        }
    }
}

```

Листинг Б.3 – Файл contact.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="/style.css" />
    <title>Контакты</title>
</head>
<body>
    <div class="all d-flex justify-content-center">
        <nav class="navbar navbar-expand-lg bg-white shadow-lg p-1 mb-5 mt-3 bg-body rounded w-75 font-family-sans-serif fs-5">
            <div class="container-fluid">
                <button class="navbar-toggler" data-bs-toggle="collapse" data-bs-target="#navbarNav">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarNav">
                    <ul class="navbar-nav mb-2 mb-lg-0">
                        <li class="nav-item">
                            <a class="nav-link" href="/Main/">Главная</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link active text-decoration-underline" href="/ContactPage/">Контакты</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="note.html">Заметки</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="tag.html">Теги</a>
                        </li>
                    </ul>
                </div>
                <span class="m-3"></span>
                <div class="lk btn text-center p-1 w-25 h-25">
                    <input id="lk_button" class="lk w-75 h-75" type="submit" style="color: white; background-color: #ad00ff"

```

```

        value="Личный кабинет" />
    </div>
    </div>
</nav>
</div>
<div class="container">
<div class="row d-flex flex-row justify-content-center m-auto">
<p id="message_find" class="text-center" style="color: fire-
brick"></p>
<div class="col-12 col-sm-6 col-xl-4 add_btn text-center fs-5 m-3">
<input id="btn_addCont" class="h-75" type="submit"
    style="color: #ad00ff; background-color: white"
    value="Добавить контакт" />
</div>

<div class="col-6 col-sm-4 col-xl-4 d-flex justify-content-end">
<input type="text" id="text_f" class="inp_find m-3 h-50" />
</div>
<div class="col-6 col-sm-3 col-xl-4 lk_btn text-center p-1 fs-5 m-
3">
    <input id="btn_find" class="h-75" type="button" style="color:
white; background-color: #ad00ff" value="Поиск" />
</div>
</div>
<div class="row dinamicBlocks justify-content-center m-auto"
id="dinamicBlocks">
</div>
</div>

<div class="modal" tabindex="-1" style="display: none" id="modalF">
<div class="modal-dialog modal-dialog-centered">
<form class="modal-content" name="contactInfo">
<div class="modal-header border-0">
<button type="button"
id="btn_Show_close"
class="btn-close"
data-bs-dismiss="modal"
aria-label="Закреть"></button>
</div>
<div class="modal-body pe-5 ps-5">
<div class="container pb-3">
<div class="row">
<textarea class="col form_textar fw-bold fs-4"
name="fio"
id="placeNameS"
rows="3"
style="resize: none; overflow: hidden"></textarea>

<div class="col contact_img rounded d-flex justify-content-end">

</div>
</div>
</div>
<div class="container"
    style="border: 3px solid #ffcc7f; border-radius: 10px">
<div class="row">

```

```

<div class="show_block" style="border: 1px solid #ffcc7f">
<div class="show_tit fw-bold pb-2 pt-2"
  style="border-bottom: 1px solid #ffcc7f"
  id="show_data">
Дата рождения
</div>
<div class="show_info pb-2 pt-2" id="sinp_data">
  <textarea class="form_textar" name="cont_birth"
id="placeDataS" cols="45" rows="2"></textarea>
</div>
</div>
</div>
<div class="row">
<div class="show_block" style="border: 1px solid #ffcc7f">
<div class="show_tit fw-bold pb-2 pt-2"
  style="border-bottom: 1px solid #ffcc7f"
  id="show_home">
Адрес проживания
</div>
<div class="show_info pb-2 pt-2" id="sinp_home">
<textarea class="form_textar" name="address" id="placeHomeS"
cols="45" rows="2"></textarea>
</div>
</div>
</div>
<div class="row">
<div class="show_block" style="border: 1px solid #ffcc7f">
<div class="show_tit fw-bold pb-2 pt-2"
  style="border-bottom: 1px solid #ffcc7f"
  id="show_work">
Место работы
</div>
<div class="show_info pb-2 pt-2" id="sinp_work">
<textarea class="form_textar"
name="cont_work"
id="placeWorks"
cols="45"
rows="2"></textarea>
</div>
</div>
</div>
<div class="row">
<div class="show_block" style="border: 1px solid #ffcc7f">
<div class="show_tit fw-bold pb-2 pt-2"
  style="border-bottom: 1px solid #ffcc7f"
  id="show_any">
Другое
</div>
<div class="show_info pb-2 pt-2" id="sinp_any">
<textarea class="form_textar" name="other" id="placeAnyS" cols="45"
rows="2"></textarea>
</div>
</div>
</div>
</div>
</div>
<div class="modal-footer border-0 justify-content-around pb-5">

```

```

<div class="lk_btn text-center p-2 w-50 h-25">
    <input id="lk_button" type="submit" style="color: white; back-
ground-color: #ad00ff" value="Редактировать" />
</div>
<div class="add_btn text-center p-1 w-25 h-25">
    <input id="lk_button" type="submit" style="color: #ad00ff;
background-color: white;" value="Удалить" />
</div>
</div>
</form>
</div>
</div>
<script src="/js/ContactReq.js"></script>
<script src="/js/AccountShow.js"></script>
<script src="/js/FormShow.js"></script>
</body>
</html>

```

Листинг Б.4 – Файл ContactReq.js

```

function changeButtonValue() {
    const lk_button = document.getElementById("lk_button");
    const add_button = document.getElementById("btn_addCont");
    if (window.innerWidth < 1140) {
        lk_button.value = "ЛК";
        add_button.value = "+";
    } else {
        lk_button.value = "Личный кабинет";
        add_button.value = "Добавить контакт";
    }
}
window.addEventListener("resize", changeButtonValue);
changeButtonValue();

document.getElementById("btn_find").addEventListener("click", function ()
{
    const find = document.getElementById("text_f").value;
    if (find == "") {
        document.getElementById("dinamicBlocks").innerHTML = "";
        GetContacts();
    } else {
        console.log("click_find");
        FindByName(find);
    }
});

async function FindByName(word) {
    const response = await
fetch(`/Contact/FindContactByFIO?request=${word}`, {
    method: "POST",
});
    if (response.ok) {
        const contacts = await response.json();
        contacts.forEach((c) => {
            document.getElementById("dinamicBlocks").innerHTML = "";
            // выводим контакты на страницу
            CreateContactHtml(c.contact);
        });
    } else {

```

```

        const errorData = await response.json();
        if (errorData) {
            document.getElementById(
                "dinamicBlocks"
            ).innerHTML = `
```

```

        "</div>";
        div.innerHTML = elemBlock;
        parent.appendChild(div);
        div.addEventListener("click", () => {
            ShowContact(div.id);
        });
    }
    function ShowContact(id) {
        document.getElementById("modalF").style.display = "block";
        GetContactInfo(id);
    }
    // Получение полной информации контакта
    async function GetContactInfo(id) {
        const response = await fetch(`/Contact/GetContact/${id}`, {
            method: "GET",
            headers: { Accept: "application/json" },
        });
        if (response.ok) {
            const contact = await response.json();
            console.log(contact);
            const form = document.forms["contactInfo"];
            form.elements["fio"].value = contact.contact.fio;
            form.elements["cont_birth"].value = con-
            tact.contact.cont_birth;
            form.elements["address"].value = contact.contact.address;
            form.elements["cont_work"].value = contact.contact.cont_work;
            form.elements["other"].value = contact.contact.other;
        }
    }
    GetContacts();

```

Листинг Б.5 – Файл account.html

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<link rel="stylesheet" href="../style.css" />
<title>Личный кабинет</title>
</head>
<body>
<div class="all d-flex justify-content-center">
<nav class="navbar navbar-expand-lg bg-white shadow-lg p-1 mb-5 mt-3 bg-
body rounded w-75 font-family-sans-serif fs-5">
<div class="container-fluid">
<button class="navbar-toggler"
data-bs-toggle="collapse"
data-bs-target="#navbarNav">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
<ul class="navbar-nav mb-2 mb-lg-0">
<li class="nav-item">
<a class="nav-link" href="/Main/">Главная</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/ContactPage/">Контакты</a>

```

```

</li>
<li class="nav-item">
<a class="nav-link" href="note.html">Заметки</a>
</li>
<li class="nav-item">
<a class="nav-link" href="tag.html">Теги</a>
</li>
</ul>
</div>
<span class="m-3"></span>
<div class="lk_btn text-center p-1 w-25 h-25">
<input id="lk_button" class="lk w-75 h-75" type="submit" style="color:
white; background-color: #ad00ff" value="Личный кабинет" />
</div>
</div>
</nav>
</div>
<div class="container-fluid">
<div class="row d-flex justify-content-center align-items-start">
<div class="col-12 col-sm-10 col-xl-5">
<div class="d-flex align-items-center justify-content-center m-3">

</div>
</div>
<div class="col-12 col-sm-10 col-xl-5 pt-3">
<form class="account pt-0 mt-0" name="accountForm" id="account_form"
method="post">
<p id="message_lk" class="text-center" style="color: firebrick"></p>
<div class="container rounded" style="border: 2px solid #ffcc7f">
<div class="row">
<div class="show_block" style="border: 1px solid #ffcc7f">
<div class="show_tit fw-bold pb-2 pt-2" style="border-bottom: 1px solid
#ffcc7f" id="lk_fio"> ФИО
</div>
<div class="show_info pb-2 pt-2" id="linp_fio">
<textarea class="form_textar" name="user_name" id="placeFio" cols="45"
rows="1"></textarea>
</div>
</div>
</div>
<div class="row">
<div class="show_block" style="border: 1px solid #ffcc7f">
<div class="show_tit fw-bold pb-2 pt-2" style="border-bottom: 1px solid
#ffcc7f" id="lk_data"> Дата рождения
</div>
<div class="show_info pb-2 pt-2" id="linp_data">
<textarea class="form_textar" name="user_birth" id="placeData" cols="45"
rows="1"></textarea>
</div>
</div>
</div>
<div class="row">
<div class="show_block" style="border: 1px solid #ffcc7f">
<div class="show_tit fw-bold pb-2 pt-2" style="border-bottom: 1px solid
#ffcc7f" id="lk_phone"> Номер телефона
</div>
<div class="show_info pb-2 pt-2" id="linp_phone">

```

```

<textarea class="form_textar" name="user_pn" id="placePhone" cols="45"
rows="1"></textarea>
</div>
</div>
</div>
<div class="row">
<div class="show_block" style="border: 1px solid #ffcc7f">
<div class="show_tit fw-bold pb-2 pt-2" style="border-bottom: 1px solid
#ffcc7f" id="lk_email"> Адрес электронной почты
</div>
<div class="show_info pb-2 pt-2" id="linp_email">
<textarea class="form_textar" name="user_email" id="placeEmail" cols="45"
rows="1"></textarea>
</div>
</div>
</div>
<div class="row">
<div class="show_block" style="border: 1px solid #ffcc7f">
<div class="show_tit fw-bold pb-2 pt-2" style="border-bottom: 1px solid
#ffcc7f" id="show_pass"> Пароль
</div>
<div class="show_info pb-2 pt-2" id="linp_pass">
<textarea class="form_textar" name="user_password" id="placePass"
cols="45" rows="1"></textarea>
</div>
</div>
</div>
</div>
</div>
<div class="row justify-content-around pt-5 pb-3">
<div class="lk_btn text-center m-2 p-2 w-50 h-25">
<input id="change_user_btn" type="button" style="color: white; back-
ground-color: #ad00ff" value="Редактировать" />
</div>
<div class="text-center lk_btn_form m-2 p-1 w-25 h-25" style="border: 3px
solid #ad00ff; border-radius: 10px; cursor: pointer;">
<input id="exit_lk_btn" type="button" style="color: #ad00ff; background-
color: white;" value="Выход" />
</div>
</div>
</form>
</div>
</div>
</div>
</div>
<script src="/js/AccountReq.js"></script>
<script>
const accountForm = document.querySelector("form.accountForm");
function changeButtonValue() {
    const lk_button = document.getElementById("lk_button");
    if (window.innerWidth < 1140) {
        lk_button.value = "ЛК";
    } else {
        lk_button.value = "Личный кабинет";
    }
}
window.addEventListener("resize", changeButtonValue);
changeButtonValue();

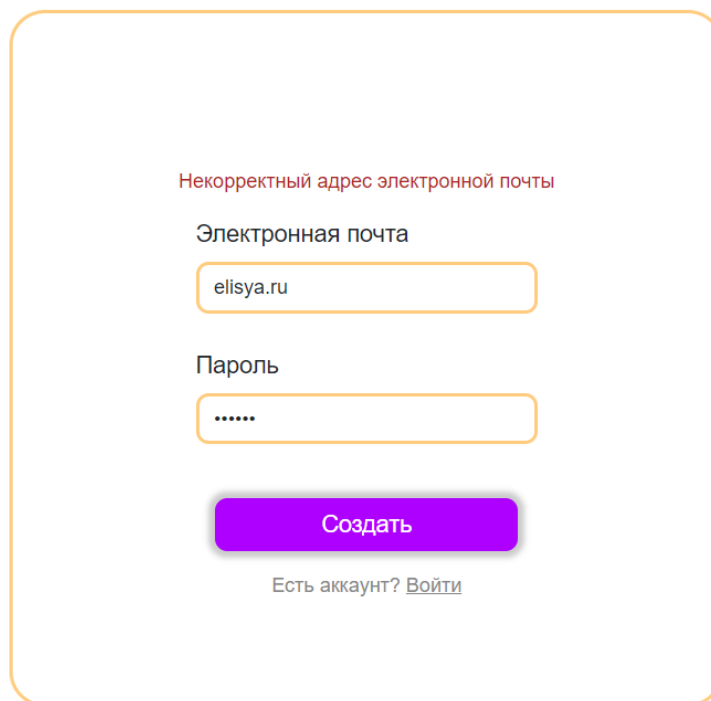
```

```

GetUser();
document.getElementById("change_user_btn").addEventListener("click",
function () {
    EditUser();
    document.getElementById("message_lk").innerHTML = '';
});
document.getElementById("exit_lk_btn").addEventListener("click", function
() {
    if (window.confirm("Вы точно хотите выйти?")) {
        window.location.href = `https://localhost:7148`;
    }
});
// Получение данных пользователя
async function GetUser() {
    const response = await fetch('/User/GetUserData', {
        method: "GET",
        headers: { "Accept": "application/json" }
    });
    if (response.ok) {
        const user = await response.json();
        console.log(user);
        const form = document.forms["accountForm"];
        form.elements["user_name"].value = user.user_name;
        form.elements["user_birth"].value = user.user_birth;
        form.elements["user_pn"].value = user.user_pn;
        form.elements["user_email"].value = user.user_email;
        form.elements["user_password"].value = user.user_password;
    }
}
//Изменение данных пользователя
async function EditUser() {
    const response = await fetch("/User/ChangeUserData", {
        method: "PUT",
        body: new FormData(document.forms.accountForm),
    });
    console.log(response.body);
    if (response.ok) {
        const user = await response.json();
        console.log(user);
    } else {
        const errorData = await response.json();
        if (errorData) {
            // Выводим ошибки в консоль
            console.error("Ошибки от сервера:", errorData);
            // Выводим ошибки пользователю
            document.getElementById("message_lk").innerHTML = error-
Data + "<br/>";
        } else {
            alert("Ошибка сервера. Попробуйте позже.");
        }
    }
}
</script>
<script src="/js/AccountShow.js"></script>
</body>
</html>

```

Приложение В
(рекомендуемое)
Результаты функционального тестирования



Некорректный адрес электронной почты

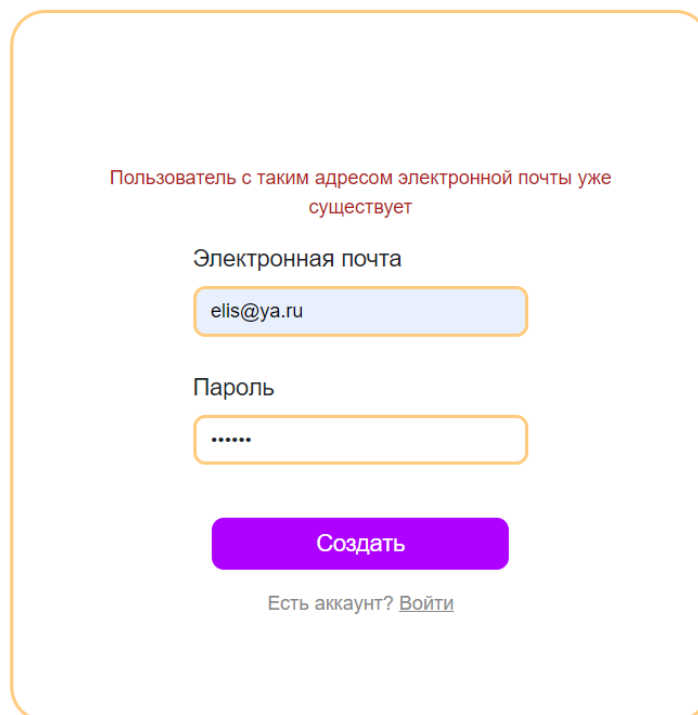
Электронная почта

Пароль

[Создать](#)

Есть аккаунт? [Войти](#)

Рисунок В.1 – Результат проверки регистрации с некорректным адресом почты



Пользователь с таким адресом электронной почты уже существует

Электронная почта

Пароль

[Создать](#)

Есть аккаунт? [Войти](#)

Рисунок В.2 – Результат проверки регистрации пользователя с существующим аккаунтом

Неверный пароль

Электронная почта

elis@ya.ru

Пароль

.....

Войти

Нет аккаунта? [Зарегистрироваться](#)

Рисунок В.3 – Результат проверки авторизации с неверным паролем

Главная Контакты Заметки Теги

Личный кабинет

Добавить контакт

Поиск

Миронова Мария
2005-08-13
Любит рисовать

Гришин Михаил
2003-06-17

Рисунок В.4 – Результат авторизации с существующим аккаунтом

КОНТАКТЫ И ЗАМЕТКИ ОНЛАЙН

Создавайте, редактируйте и систематизируйте свои данные в записной книге онлайн

Электронная почта

Пароль

Войти

Нет аккаунта? [Зарегистрироваться](#)

Рисунок В.5 – Результат выхода из приложения

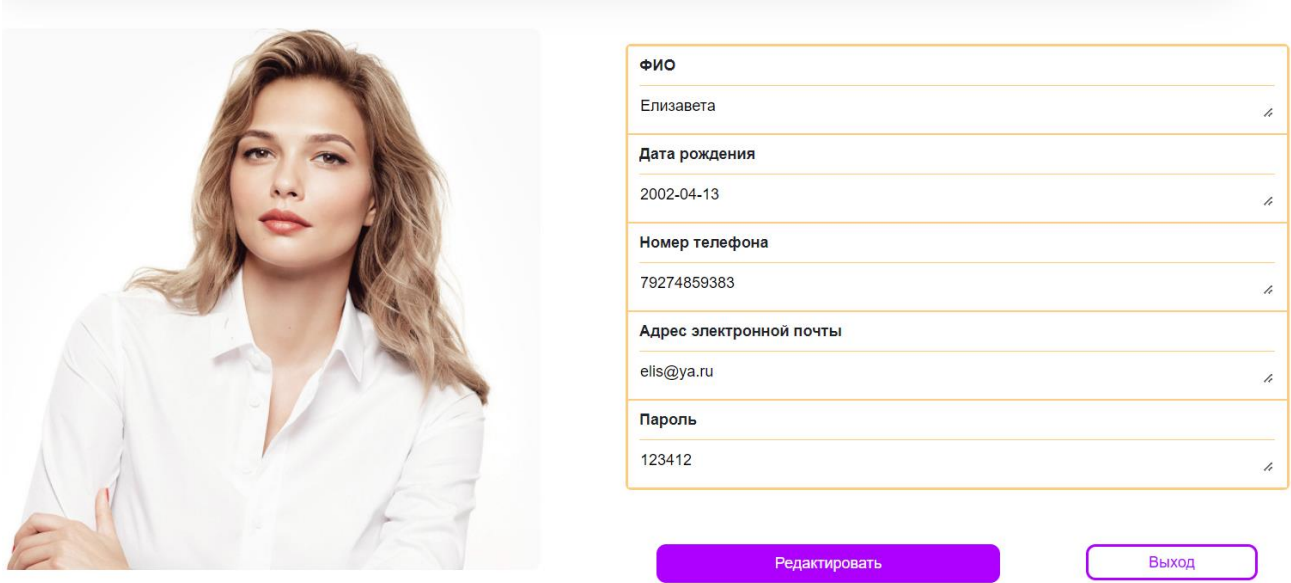


Рисунок В.6 – Данные пользователя до редактирования

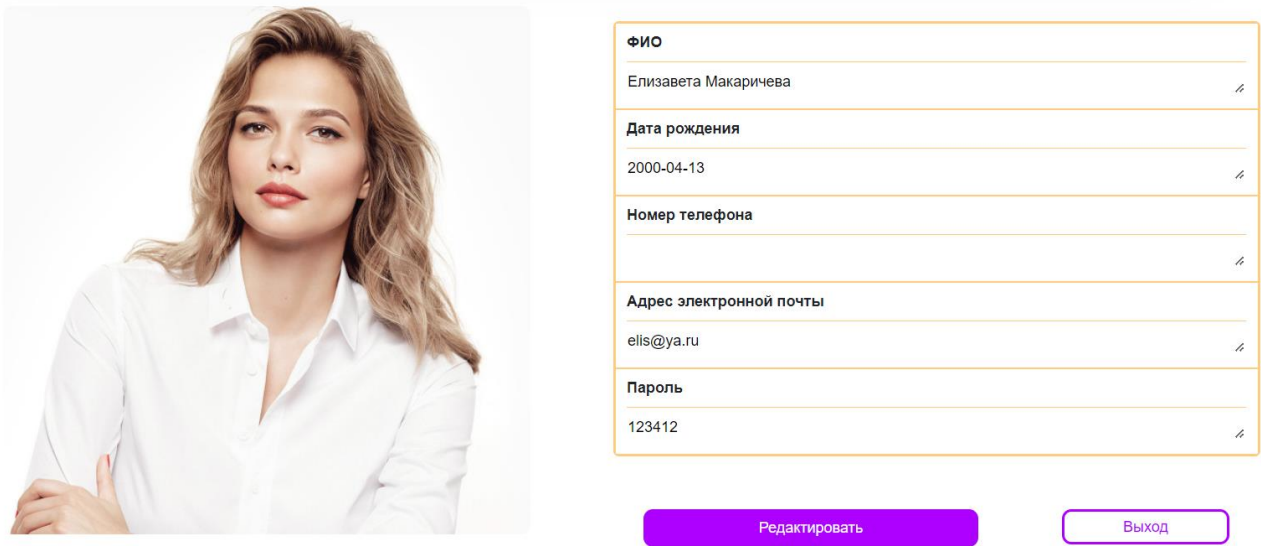


Рисунок В.7 – Данные пользователя после редактирования

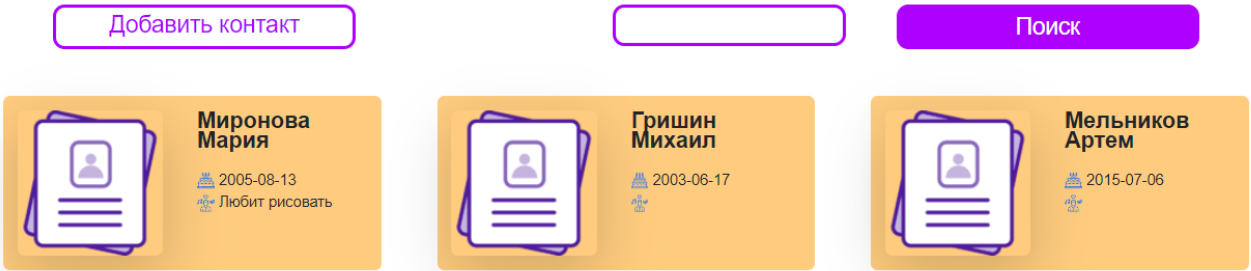


Рисунок В.8 – Результат добавления нового контакта «Мельников Артем»

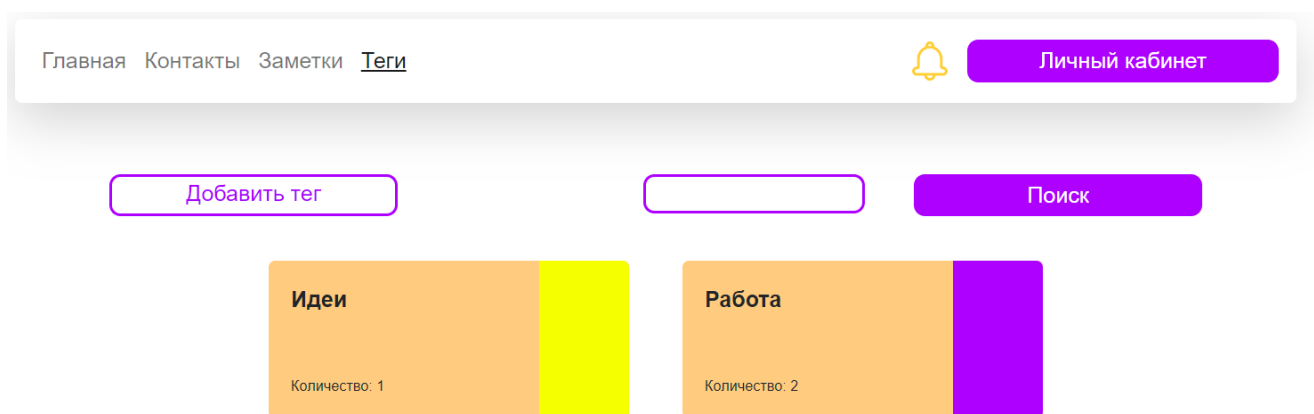


Рисунок В.9 – Страница тегов перед добавлением нового тега

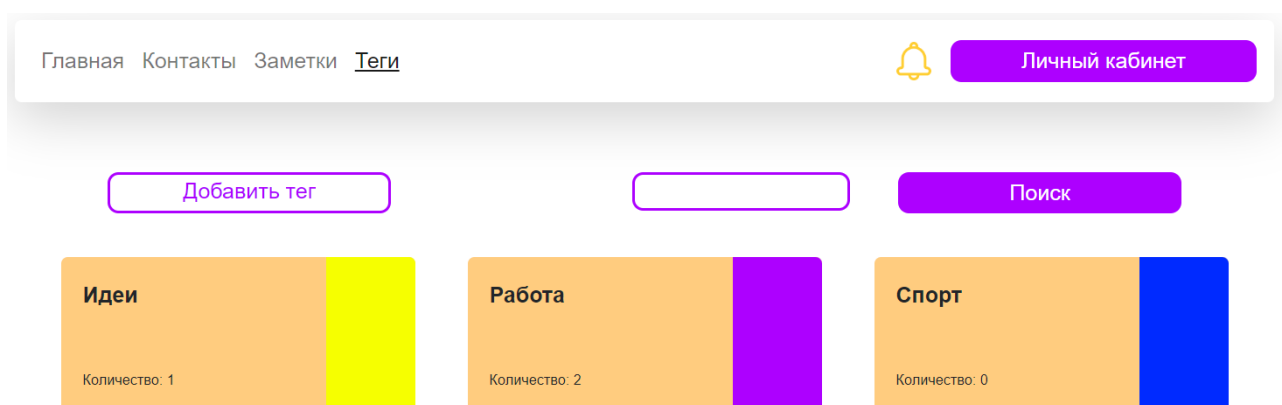


Рисунок В.10 – Страница тегов после добавления нового тега

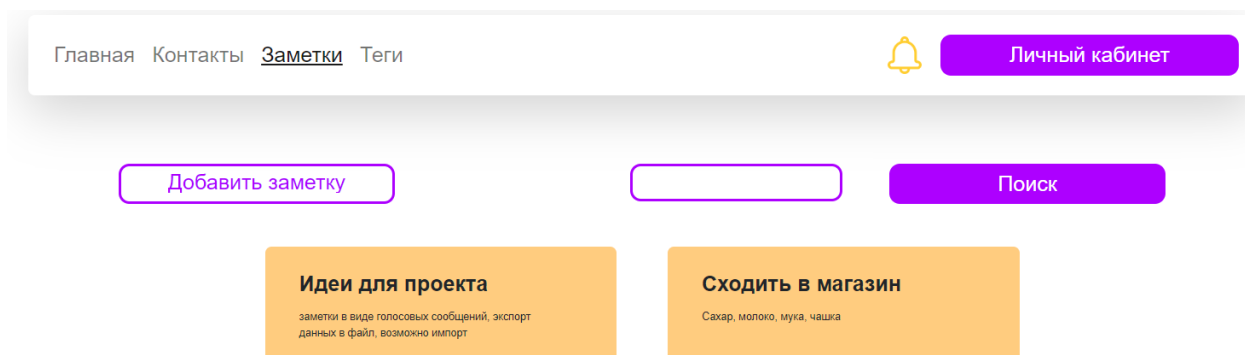


Рисунок В.11 – Страница заметок перед добавлением новой заметки

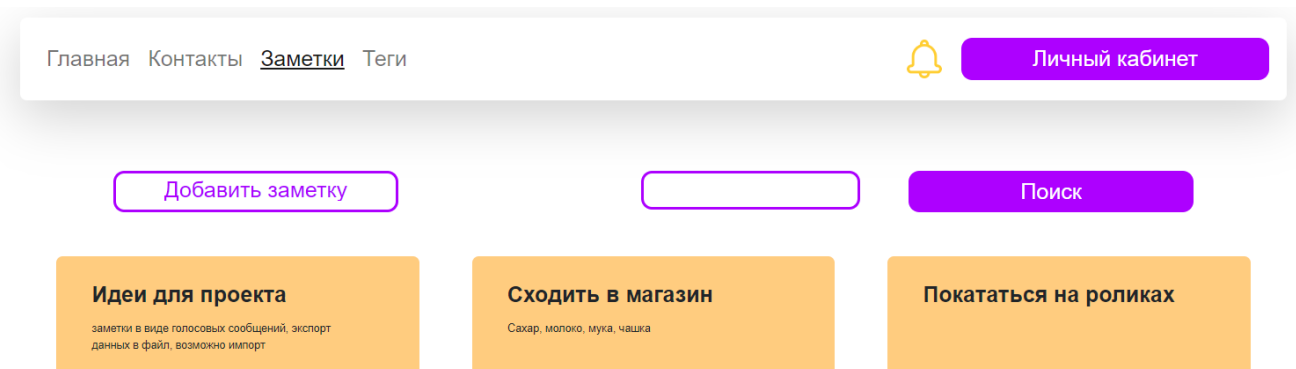


Рисунок В.12 – Страница заметок после добавления новой заметки

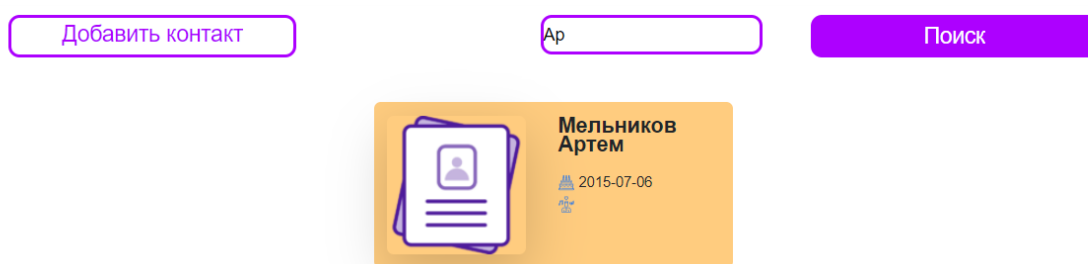


Рисунок В.13 – Результат поиска контакта, содержащего «Ар» в названии

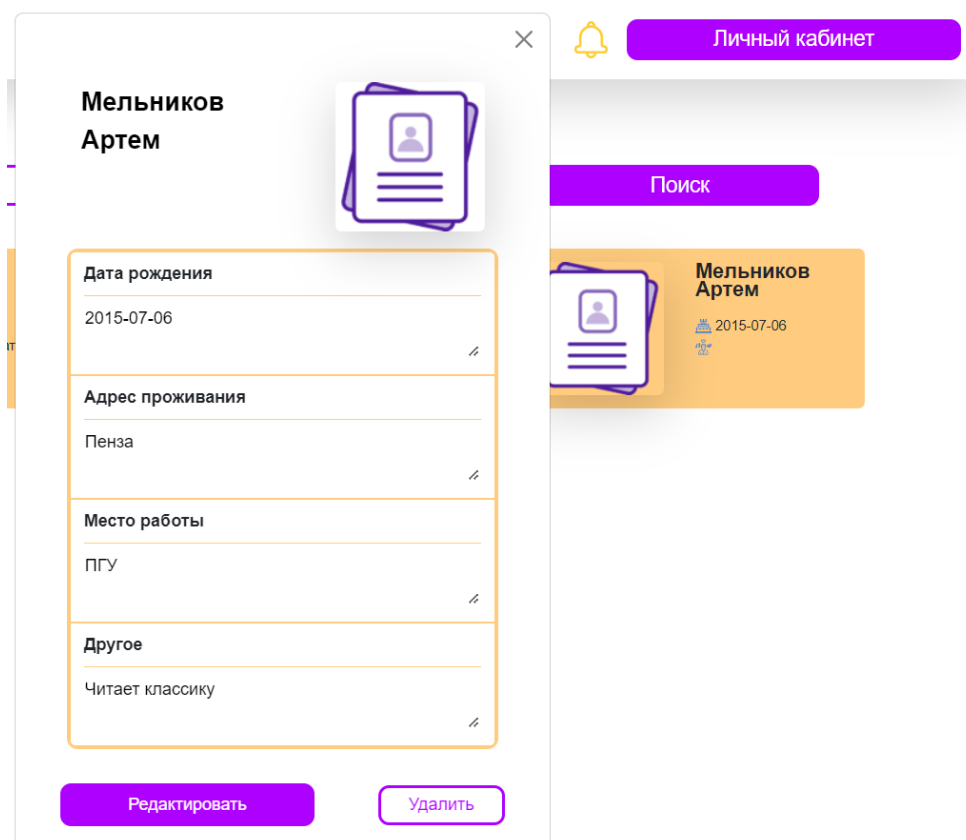


Рисунок В.14 – Изменение данных контакта «Мельников Артем»

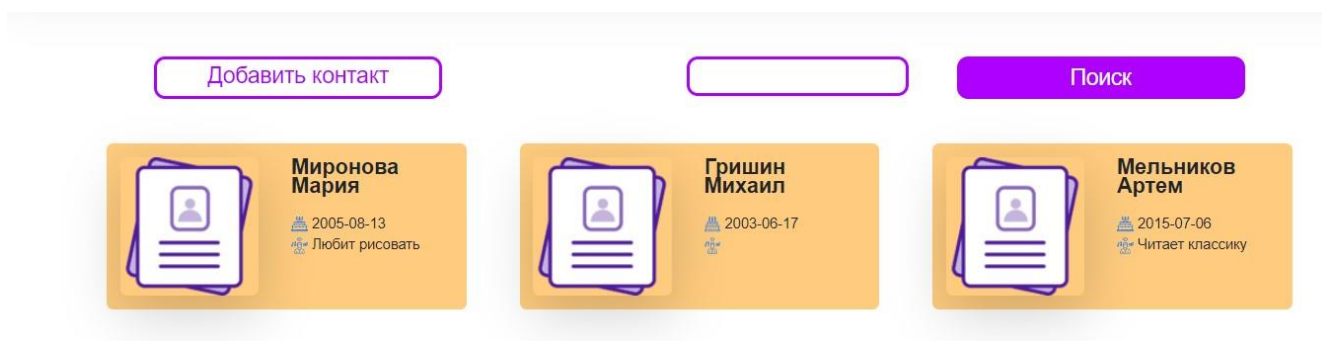


Рисунок В.15 – Результат отображения измененных данных

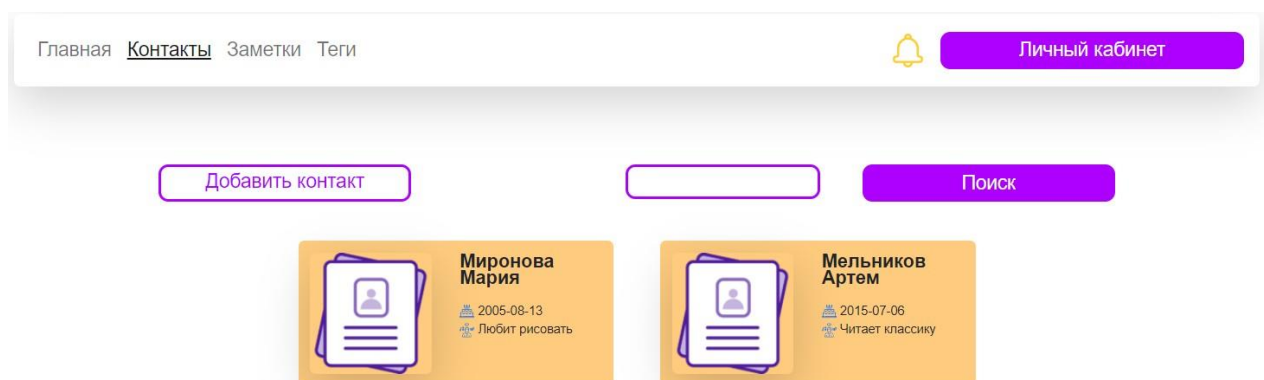


Рисунок В.16 – Результат удаления контакта «Миронова Мария»

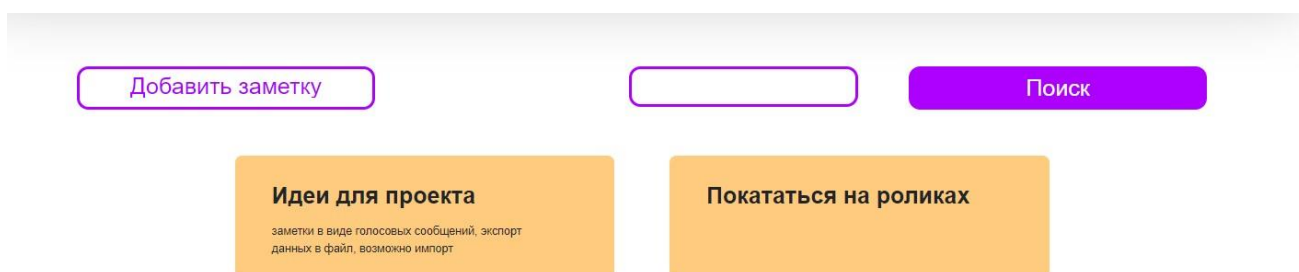


Рисунок В.17 – Результат удаления заметки «Сходить в магазин»

Приложение Г
(рекомендуемое)

Результаты тестирования адаптивности интерфейса

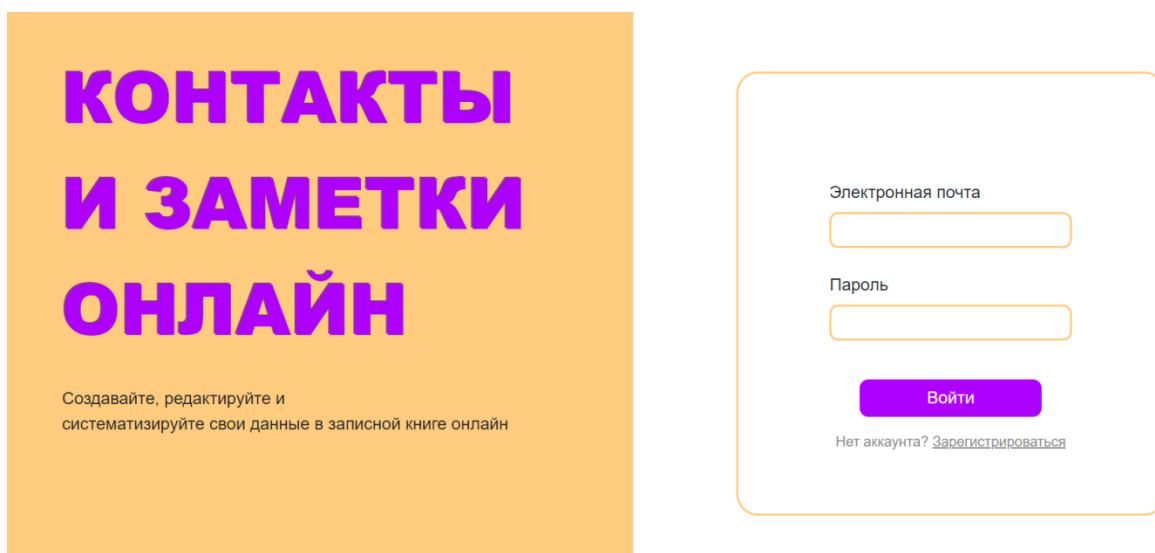


Рисунок Г.1 – Интерфейс начальной страницы при разрешении 1440 рх

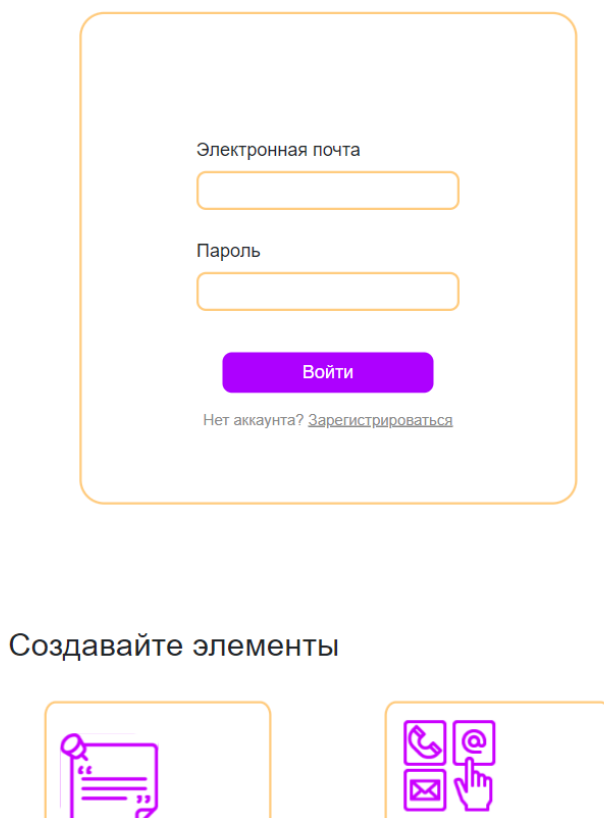
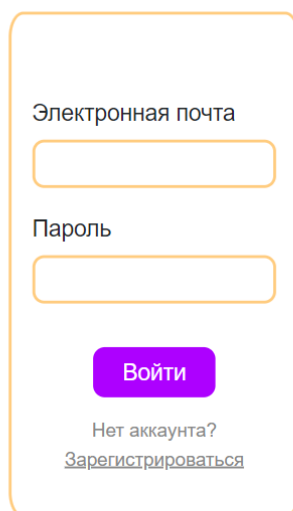


Рисунок Г.2 – Интерфейс начальной страницы при разрешении 768 рх



Электронная почта

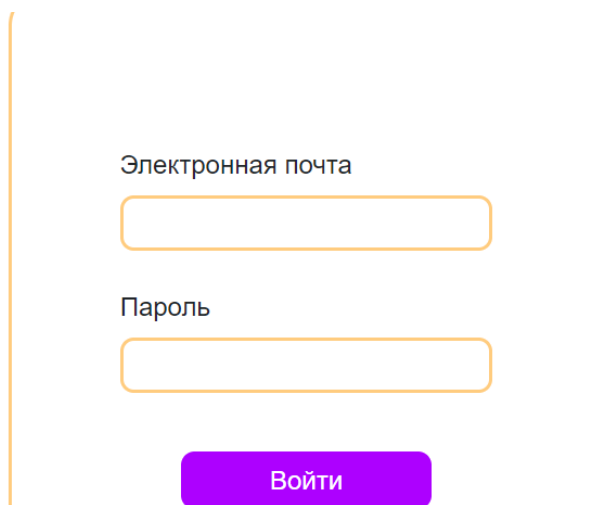
Пароль

Войти

Нет аккаунта?
[Зарегистрироваться](#)

Создавайте элементы

Рисунок Г.3 – Интерфейс начальной страницы при разрешении 375 px
(книжная ориентация)



Электронная почта

Пароль

Войти

Рисунок Г.4 – Интерфейс начальной страницы при разрешении 375 px
(альбомная ориентация)

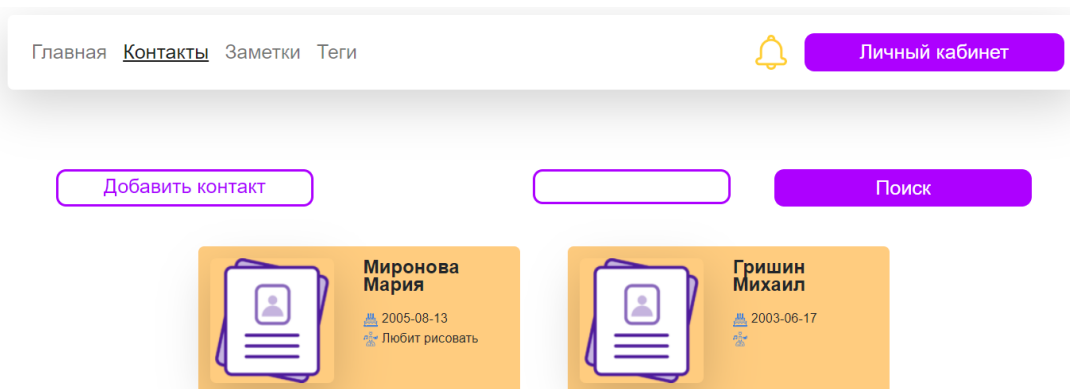


Рисунок Г.5 – Интерфейс страницы контактов при разрешении 1440 рх

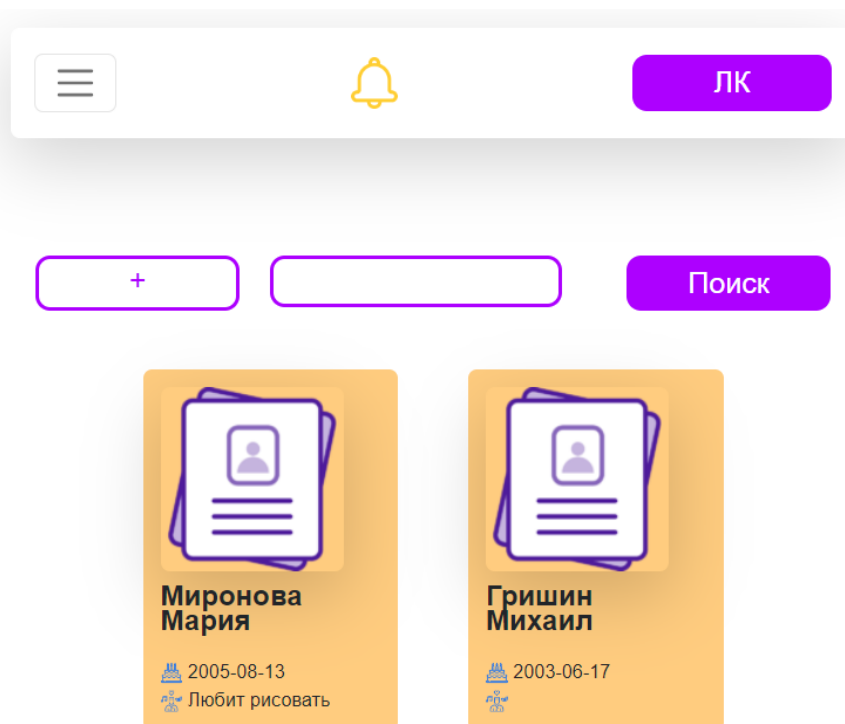


Рисунок Г.6 – Интерфейс страницы контактов при разрешении 768 рх

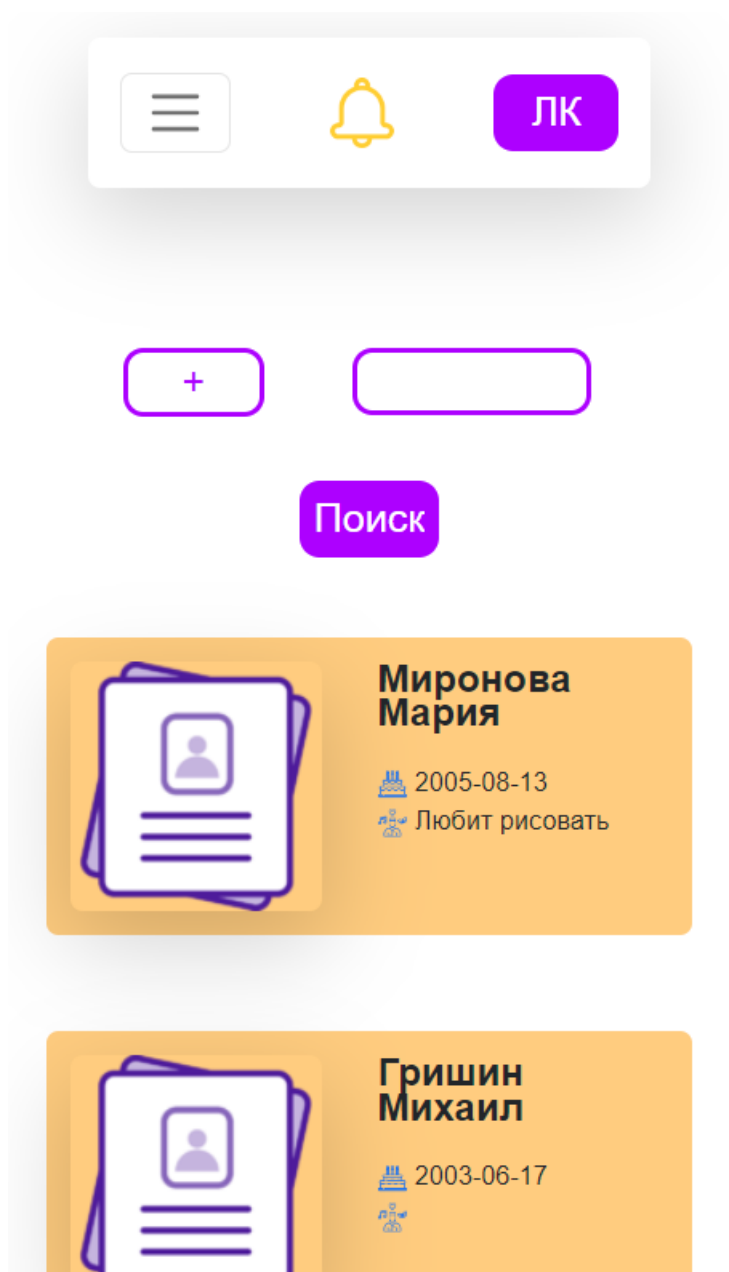
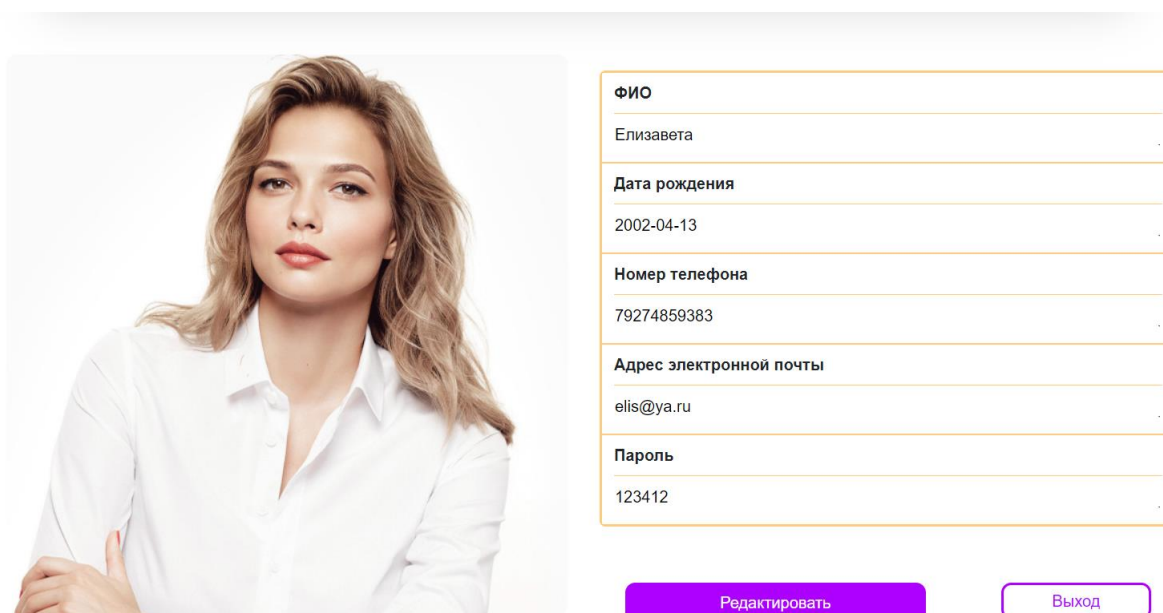


Рисунок Г.7 – Интерфейс страницы контактов при разрешении 375 рх
(книжная ориентация)

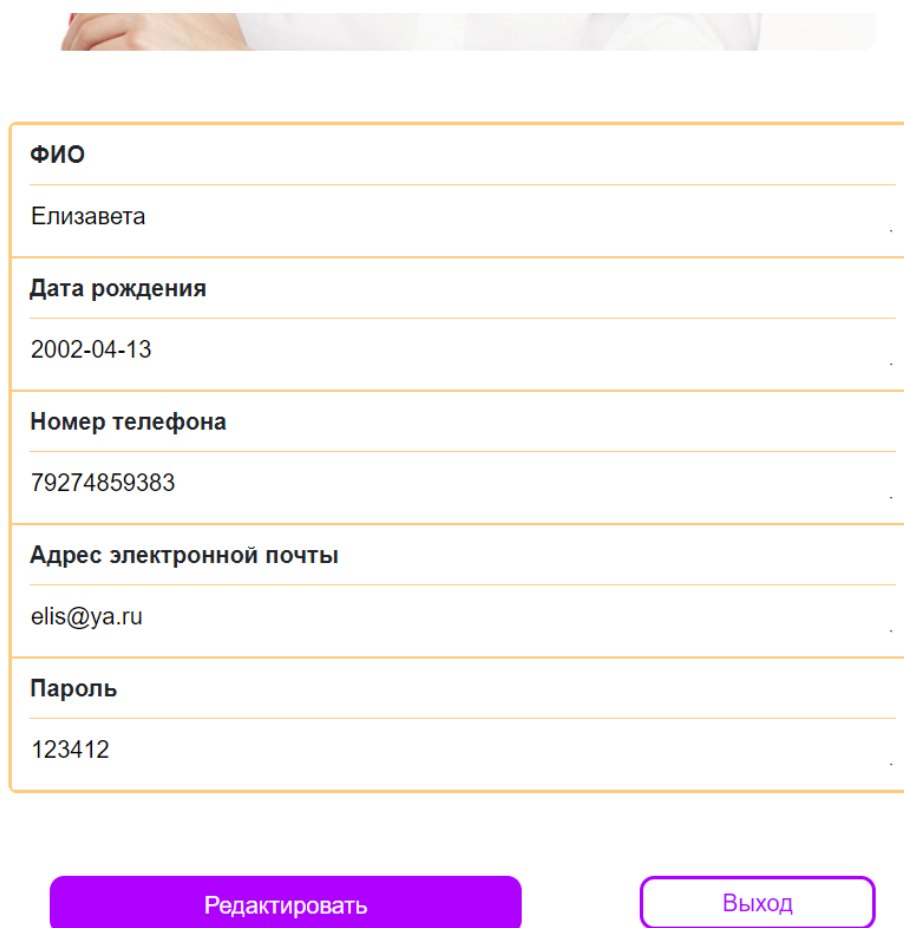


Формат: 1440 px

| | |
|---|-----------------------|
|  | ФИО |
| | Елизавета |
| | Дата рождения |
| | 2002-04-13 |
| | Номер телефона |
| | 79274859383 |
| Адрес электронной почты | |
| elis@ya.ru | |
| Пароль | |
| 123412 | |

[Редактировать](#) [Выход](#)

Рисунок Г.8 – Интерфейс личного кабинета при разрешении 1440 px



Формат: 768 px

| | |
|--|-----------------------|
|  | ФИО |
| | Елизавета |
| | Дата рождения |
| | 2002-04-13 |
| | Номер телефона |
| | 79274859383 |
| Адрес электронной почты | |
| elis@ya.ru | |
| Пароль | |
| 123412 | |

[Редактировать](#) [Выход](#)

Рисунок Г.9 – Интерфейс личного кабинета при разрешении 768 px

| |
|--------------------------------|
| ФИО |
| Елизавета |
| Дата рождения |
| 2002-04-13 |
| Номер телефона |
| 79274859383 |
| Адрес электронной почты |
| elis@ya.ru |
| Пароль |
| 123412 |

Редактировать

Выход

Рисунок Г.10 – Интерфейс страницы контактов при разрешении 375 рх
(книжная ориентация)

| |
|-----------------------|
| ФИО |
| Елизавета |
| Дата рождения |
| 2002-04-13 |
| Номер телефона |
| 79274859383 |

Рисунок Г.11 – Интерфейс страницы контактов при разрешении 375 рх
(альбомная ориентация)