

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Пензенский государственный университет»

Кафедра «Математическое обеспечение и применение ЭВМ»

ОТЧЕТ

по учебной (проектно-технологической) практике

Выполнил: Макаричева Е. М.

Направление: 09.03.04 «Программная инженерия»

Группа: 20ВП1

Руководитель практики: Михалев А.Г.

Работа защищена

Оценка

Пенза, 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра “Математическое обеспечение и применение ЭВМ”

“УТВЕРЖДАЮ”

Зав. кафедрой _____
“ _____ ” _____ 20 ____ г.

ЗАДАНИЕ

на учебную (проектно-технологическую) практику

Студенту _____ Макаричевой Е.М. _____ Группа 20ВП1
Тема задания _____ Разработка веб-приложения «Ежедневник»

Исходные данные (технические требования) на проектирование

1. Изучить методологию проектирования UML 2.0. Для этого изучить бесплатный курс в системе ИНТУИТ. Адрес курса: <https://intuit.ru/studies/courses/480/336/info>

2. Реализовать программный проект с использованием UML 2.0. Реализация включает в себя выполнение этапов: анализ требований и постановка задачи; проектирование; кодирование; тестирование; оформление отчета

3. Задание: Разработать приложение, позволяющее хранить информацию о событиях, привязанных к определенному времени (встречи, мероприятия, дни рождения). Необходимо предусмотреть возможности добавления, удаления и редактирования записи. Приложение должно иметь интуитивно понятный пользовательский интерфейс.

Объем работы по курсу

1. Расчетная часть

1. Изучение методологии проектирования UML 2.0

2. Анализ требований и предметной области

3. Проектирование приложения или бизнес-процесса

4. Кодирование

2. Графическая часть

Отсутствует

3. Экспериментальная часть

Тестирование и верификация проекта

Срок выполнения проекта по разделам

| | | |
|--|---|------------|
| 1. Изучение методологии проектирования UML 2.0 | к | 04.07.2022 |
|--|---|------------|

| | | |
|---|---|------------|
| 2. Анализ требований и предметной области | к | 06.07.2022 |
|---|---|------------|

| | | |
|-------------------|---|------------|
| 3. Проектирование | к | 07.07.2022 |
|-------------------|---|------------|

| | | |
|----------------|---|------------|
| 4. Кодирование | к | 08.07.2022 |
|----------------|---|------------|

| | | |
|---------------------------------------|---|------------|
| 5. Тестирование и верификация проекта | к | 09.07.2022 |
|---------------------------------------|---|------------|

| | | |
|----------------------|---|------------|
| 6. Оформление отчета | к | 11.07.2022 |
|----------------------|---|------------|

Дата выдачи задания “28” июня 2022 г.

Дата защиты отчета “ ____ ” _____

Руководитель _____ Михалев А.Г.

Задание получил “28” июня 2022 г.

Студент _____ Макаричева Е. М.

Реферат

Отчет по практике содержит 40 листов, 8 рисунков, 5 таблиц, 5 использованных источников, 3 приложения.

ВЕБ-ПРИЛОЖЕНИЕ, КАЛЕНДАРЬ, ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС, ЯЗЫК МОДЕЛИРОВАНИЯ UML, ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ, ДИАГРАММА ДЕЯТЕЛЬНОСТИ, ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ, ТЕСТИРОВАНИЕ.

Целью практики является разработка приложения для хранения различных событий, привязанных ко времени.

Разработка проводилась на языке программирования JavaScript в редакторе кода Visual Studio Code.

Осуществлено функциональное тестирование разработанного программного обеспечения, которое показало корректность его работы.

| | | | | | | | | |
|-----------|------|-----------------|-------|------|---|--------------|------|--------|
| | | | | | ПГУ 09.03.04 – 04УП201.18 | | | |
| | | | | | | | | |
| Из | Лист | № докум. | Подп. | Дата | | | | |
| Разраб. | | Макаричева Е.М. | | | Разработка веб-приложения «Ежедневник» | Лит. | Лист | Листов |
| Пров. | | Михалев А.Г. | | | | | 4 | 40 |
| | | | | | Отчет. | Группа 20ВП1 | | |
| Н. контр. | | | | | | | | |
| Утв. | | | | | | | | |

Содержание

| | |
|--|----|
| Введение | 6 |
| 1. Постановка задачи и анализ требований | 7 |
| 1.1. Постановка задачи..... | 7 |
| 1.2 Формулировка и анализ требований к разрабатываемому программному обеспечению | 7 |
| 1.2.1 Требования к интерфейсу пользователя | 7 |
| 1.2.2 Требования к структуре данных | 8 |
| 1.2.3 Требования к программным средствам | 9 |
| 2. Проектирование программы | 11 |
| 2.1 Модель интерфейса..... | 11 |
| 2.2 Проектирование структур данных | 13 |
| 3. Реализация программы | 15 |
| 3.1 Кодирование | 15 |
| 3.2 Диаграмма деятельности | 16 |
| 4. Тестирование | 18 |
| Заключение | 20 |
| Список использованных источников | 21 |
| Приложение А. Текст программы | 22 |
| Приложение Б. Скриншоты работы программы..... | 34 |
| Приложение В. Сертификат открытого университета | 39 |

Введение

Актуальность темы заключается в том, что в современном мире человеку приходится работать с большими объемами информации, поэтому удобнее и эффективнее использовать специальные базы данных и информационные системы, которые позволяют хранить необходимую информацию, схожую по тематике.

Понятие базы данных (БД) можно определить следующим образом – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. Базы данных находят применение в самых различных сферах деятельности человека: индивидуальное предприятие, склад, автосервис, институт, больница, школа и т.д.

Базы данных позволяют эффективнее организовать деятельность какой-либо организации или человека.

Целью разрабатываемого приложения является создание базы данных, которая будет хранить данные о мероприятиях, встречах и других событиях.

Для достижения поставленной цели нужно решить следующие задачи:

- изучить особенности работы с базами данных и основные операции для работы с ними;
- разработать модель пользовательского интерфейса программного средства;
- спроектировать диаграммы: вариантов использования, деятельности и компонентов с помощью нотации UML.
- разработать основной функционал приложения;
- провести тестирование разработанной программы;

Разработка программы осуществлялась под управлением операционной системы Windows 10 Pro в среде программирования Visual Studio Code на JavaScript.

1. Постановка задачи и анализ требований

1.1. Постановка задачи

На учебную (проектно-технологическую) практику была поставлена задача: Разработать приложение, позволяющее хранить информацию о событиях, привязанных к определенному времени (встречи, мероприятия, дни рождения). Необходимо предусмотреть возможности добавления, удаления и редактирования записи. Приложение должно иметь интуитивно понятный пользовательский интерфейс.

Для достижения поставленной задачи необходимо:

- исследовать современные технологии программирования на языке JavaScript в среде программирования Visual Studio Code;
- провести анализ требований в соответствии с поставленной задачей;
- разработать алгоритм решения задачи;
- выполнить тестирование разработанного приложения. Результаты представить в виде таблицы и скриншотов;
- представить сертификат о прохождении курса в ИНТУИТ;
- оформить отчет по практике в соответствии с ГОСТ.

1.2 Формулировка и анализ требований к разрабатываемому программному обеспечению

Перед разработкой приложения необходимо сформулировать и проанализировать требования, предъявляемые к программному продукту в соответствии с заданием.

1.2.1 Требования к интерфейсу пользователя

Веб-приложение должно предоставить пользователю удобный и интуитивно понятный интерфейс, позволяющий:

- выбирать дату для добавления события;

- добавлять необходимую информацию о событии с помощью заполнения полей;
- удалять события;
- редактировать информацию о событии;
- просматривать события на день.

1.2.2 Требования к структуре данных

В практической работе для хранения информации необходимо использовать базу данных. При работе с БД любая операция над данными включает в себя селекцию данных, то есть выделение из всей совокупности именно тех данных, над которыми должна быть выполнена требуемая операция. Для однозначной идентификации записи в базе данных используются уникальные значения - ключи.

Необходимо предусмотреть следующие операции для корректной работы базы данных:

- идентификация данных и нахождение их позиции в БД;
- выборка (чтение) данных из БД;
- включение (запись) данных в БД;
- удаление данных из БД;
- модификация (изменение) данных БД.

Обработка данных в БД осуществляется с помощью процедур базы данных – транзакций. Транзакцией называют упорядоченное множество операций, переводящих БД из одного согласованного состояния в другое. Транзакция либо выполняется полностью, т.е. выполняются все входящие в неё операции, либо не выполняется совсем, если в процессе её выполнения возникает ошибка [1].

Одна запись в базе данных должна содержать следующие поля: название события, место проведения, время проведения, участники и дополнительная информация. Название события является ключом для записи,

соответственно оно не должно быть пустым. Если же какое-нибудь поле не заполнено (кроме названия), то ему присваивается значение по умолчанию.

1.2.3 Требования к программным средствам

Разработанное приложение должно предоставить пользователю следующие функции:

- просмотр всех событий на день;
- добавить запись;
- удалить запись;
- редактировать запись.

В процессе анализа требований к программным средствам была составлена диаграмма вариантов использования.

Диаграмма вариантов использования, разработанная с помощью средства моделирования – AgroUML, приведена на рисунке 1. [2]

На диаграмме вариант использования «Добавить запись» связан отношением включения с такими вариантами, как «Заполнить поле 'Название'», «Заполнить поле 'Место'», «Заполнить поле 'Время'», «Заполнить поле 'Участники'» и «Заполнить поле 'Дополнительная информация'».

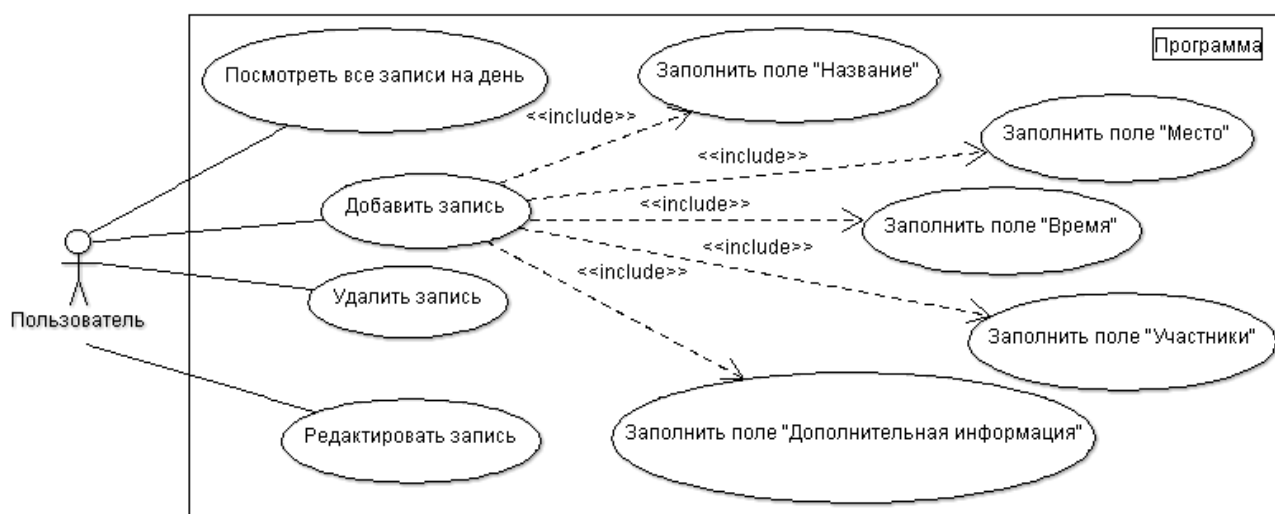


Рисунок 1 – Диаграмма вариантов использования

Описания некоторых спецификация прецедентов представлены в таблицах 1-3.

Таблица 1 – Спецификация прецедента «Добавить запись»

| |
|---|
| Наименование: Добавить запись |
| ID: 1 |
| Краткое описание: на основе ввода пользователя программа добавляет запись в базу данных и выводит ее на экран |
| Действующие лица: программа, пользователь |
| Основной поток: 1. Пользователь заносит информацию в специальные поля и нажимает кнопку «Добавить» 2. Программа получает данные пользователя и формирует на их основе запись, добавляет ее в БД и выводит на экран |
| Постусловие: Запись добавлена |

Таблица 2 – Спецификация прецедента «Удалить запись»

| |
|---|
| Наименование: Удалить запись |
| ID: 2 |
| Краткое описание: на основе действия пользователя программа удаляет конкретную запись |
| Действующие лица: программа, пользователь |
| Основной поток: 1. Пользователь нажимает на кнопку удаления для конкретной записи 2. Программа находит по ключу запись, удаляет ее из базы данных и с экрана |
| Постусловие: Запись удалена |

Таблица 3 – Спецификация прецедента «Просмотреть все записи за день»

| |
|--|
| Наименование: Просмотреть все записи за день |
| ID: 3 |
| Краткое описание: на экран выведены записи на конкретный день |
| Действующие лица: программа, пользователь |
| Основной поток: 1. Пользователь нажимает на ячейку календаря с датой 2. Программа ищет в базе данных записи, связанные с этой датой и выводит их на экран |
| Постусловие: Записи видны пользователю |

2. Проектирование программы

2.1 Модель интерфейса

В результате проектирования была разработана модель интерфейса, представленная на рисунках 2-4.

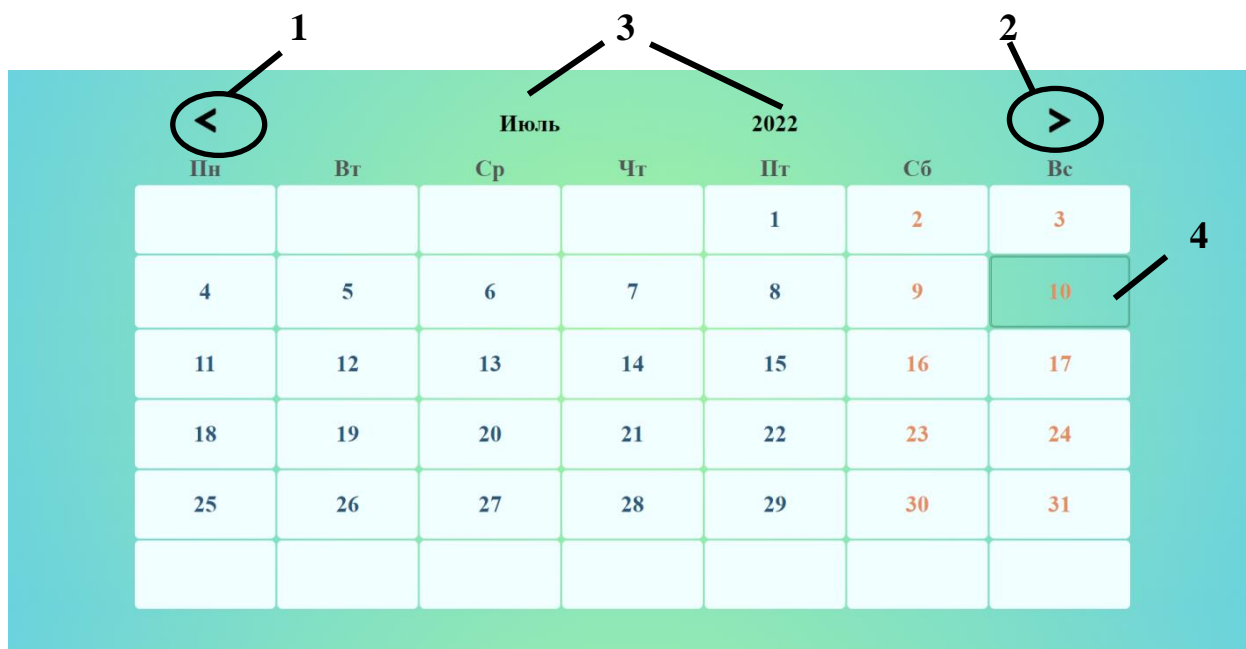


Рисунок 2 – Форма основного экрана приложения

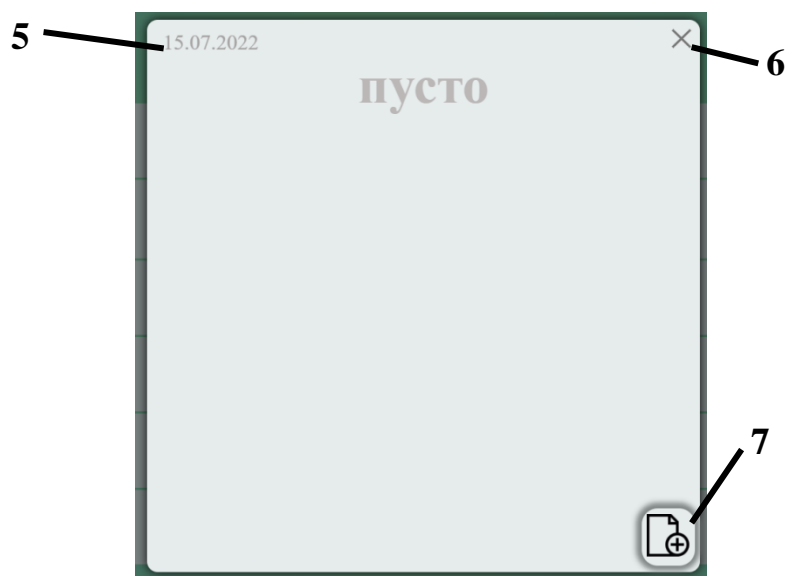


Рисунок 3 – Форма просмотра записей на день

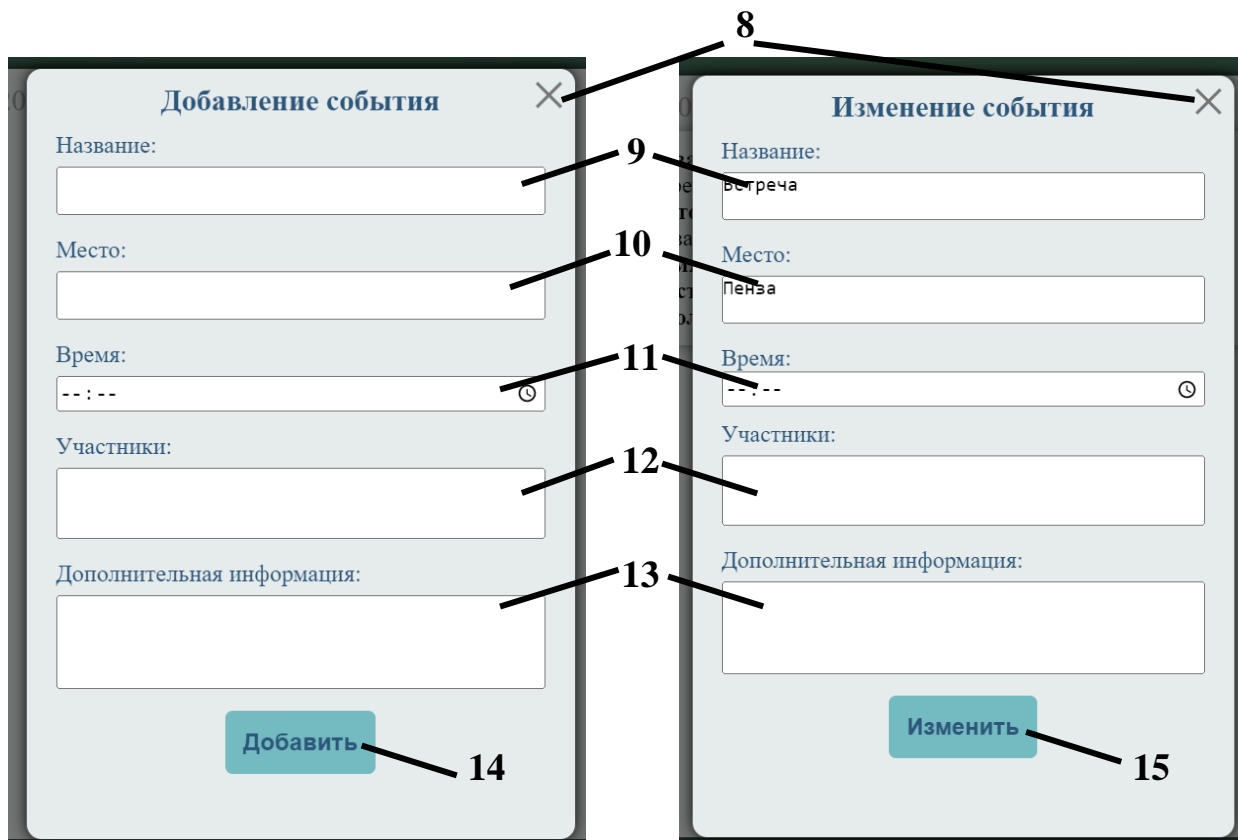


Рисунок 4 – Формы добавления события и изменения события

Интерфейсы разработанных форм содержат следующие компоненты:

- 1 – кнопка для перехода к предыдущему месяцу;
- 2 – кнопка перехода к следующему месяцу;
- 3 – текущий месяц и год;
- 4 – текущая дата;
- 5 – дата, для которой открыта форма просмотра записей;
- 6 – кнопка закрытия формы просмотра записей;
- 7 – кнопка для открытия формы добавления записи;
- 8 – кнопки закрытия форм;
- 9 – поле для ввода названия события;
- 10 – поле для ввода места проведения мероприятия;
- 11 – поле для ввода времени проведения;
- 12 – поле для ввода участников мероприятия;
- 13 – поле для ввода дополнительной информации о событии;
- 14 – кнопка добавления записи;
- 15 – кнопка изменения записи.

2.2 Проектирование структур данных

В процессе проектирования программного продукта было принято решение о работе с IndexedDB – базой данных, встроенной в браузер. Она позволяет хранить практически любые значения по ключам и содержать в одной БД несколько типов ключей.[3][4]

Для начала работы с IndexedDB, нужно подключиться к базе данных с помощью метода `open`. Ниже представлен код реализации:

```
let dbReq = indexedDB.open('myNotes', 1); //название бд и версия
//срабатывает, если база устарела, инициализация
dbReq.onupgradeneeded = (event) => {
    // Зададим переменной db ссылку на базу данных
    db = event.target.result;
    // Создадим хранилище объектов с именем notes.
    notes = db.createObjectStore('notes', { keyPath: 'title' });
    index = notes.createIndex('day_idx', 'day');
};
```

Вызов возвращает `dbReq` объект, на котором нужно прослушать события:

- `success`: база данных готова к использованию, в ней есть «объект базы данных», который необходимо использовать для дальнейших вызовов.
- `error`: не удалось открыть.
- `onupgradeneeded`: база данных готова, но ее версия устарела.

Переменная `notes` содержит хранилище объектов с названием `notes`, и путем к свойству объекта - ключа. Соответственно, в разрабатываемой базе данных ключом является название события (записи).

На рисунке 5 отображены записи в хранилище `notes`.

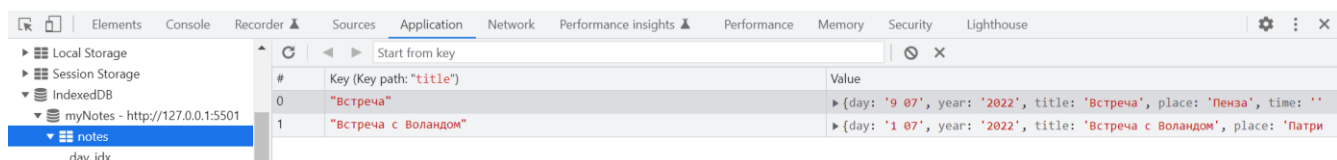
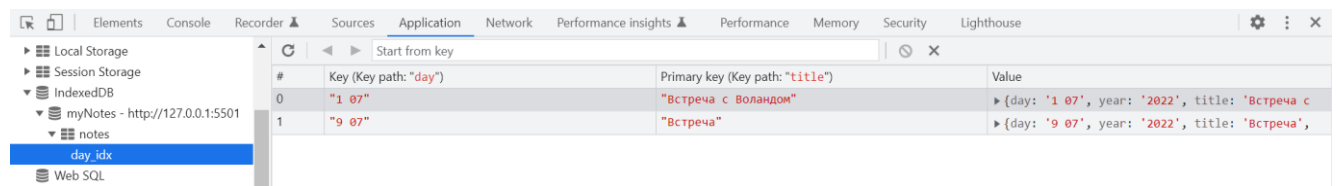


Рисунок 5 – Отображение информации в `notes`

Индекс - это “дополнение” к хранилищу, которое отслеживает данное поле объекта. Для каждого значения этого поля в нем хранится список ключей для объектов, имеющих это значение.

Переменная `index` содержит индекс для поиска событий в БД по полю даты.

На рисунке 6 отображены данные индекса с названием `day_idx`. Для упрощения операции поиска записей данные в IndexedDB заранее упорядочиваются.



| # | Key (Key path: "day") | Primary key (Key path: "title") | Value |
|---|-----------------------|---------------------------------|---|
| 0 | "1 07" | "Встреча с Воландом" | {day: '1 07', year: '2022', title: 'Встреча с |
| 1 | "9 07" | "Встреча" | {day: '9 07', year: '2022', title: 'Встреча', |

Рисунок 6 – Отображение информации в `day_idx`

Одна запись в БД представляет собой объект с полями. Структура объекта представлена ниже:

```
note = {  
  day: "",  
  year: "",  
  title: "",  
  place: "",  
  time: "",  
  people: "",  
  info: "",  
};
```

Описание полей объекта:

`day` – день и месяц события;

`year` – год события;

`title` – название события;

`place` – место проведения;

`time` – время проведения;

`people` – участники мероприятия;

`info` - дополнительная информация о событии.

Если значение поля объекта не задано пользователем, то в базу данных заносится значение по умолчанию, а именно пустая строка.

3. Реализация программы

3.1 Кодирование

В ходе выполнения практической работы было разработано веб-приложение, код которого приведен в приложении А.

В процессе реализации была составлена диаграмма компонентов, которая отображает разбиение программной системы на структурные компоненты и связи между компонентами (Рисунок 7).

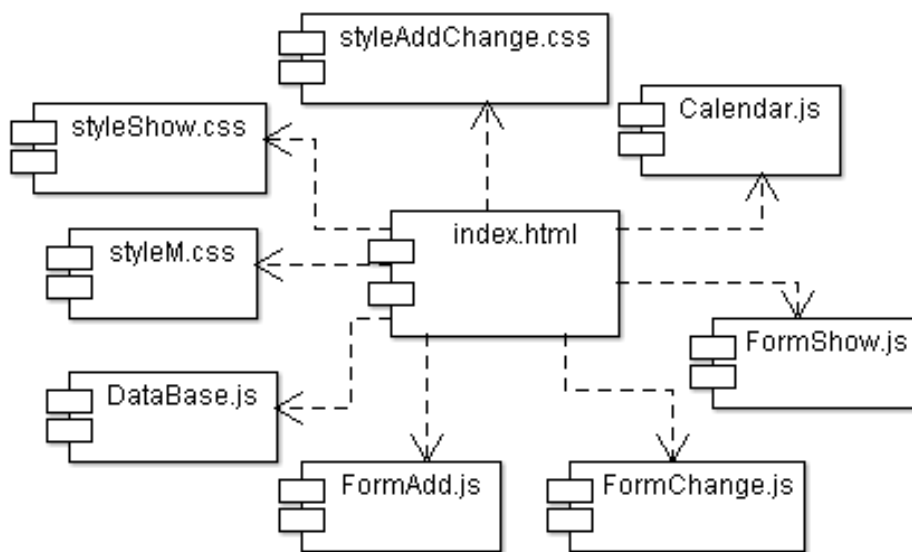


Рисунок 7 – Диаграмма компонентов

Описание компонентов приведено в таблице 4.

Таблица 4 – Описание структурных компонентов программы

| Компонент | Описание |
|--------------------|---|
| index.html | Файл, содержащий разметку веб-страницы |
| styleShow.css | Файл, содержащий стили формы просмотра всех записей |
| styleM.css | Файл, содержащий стили основного окна приложения |
| styleAddChange.css | Файл, содержащий стили форм добавления и изменения |
| Calendar.js | Файл с реализацией динамического календаря |

| | |
|---------------|--|
| DataBase.js | Файл, содержащий информацию о базе данных |
| FormAdd.js | Файл с функционалом формы добавления записи |
| FormChange.js | Файл с функционалом формы изменения записи |
| FormShow.js | Файл с функционалом формы просмотра всех записей |

3.2 Диаграмма деятельности

В ходе выполнения курсовой работы было разработано приложение «Daily planner», код которого приведен в приложении А.

Для алгоритма добавления записи была разработана диаграмма деятельности, изображенная на рисунке 8. Диаграмма деятельности создана с помощью средства UML моделирования – AgroUML.

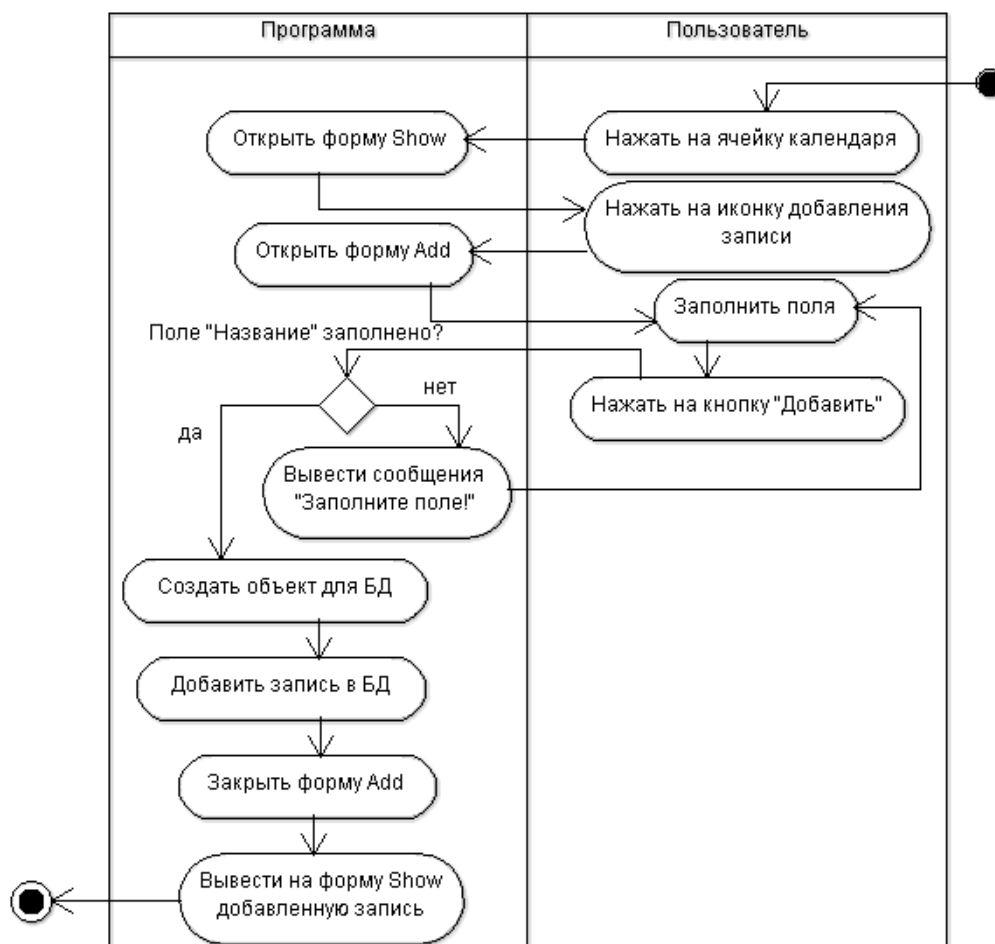


Рисунок 8 – Диаграмма деятельности

На диаграмме деятельности представлены две дорожки: Пользователь и Программа. Пока пользователь не совершает никаких действий, программа работать не будет. Пользователь должен выбрать дату – ячейку из календаря и нажать на нее. Откроется форма просмотра всех записей Show. Для добавления записи необходимо нажать соответствующую иконку, после чего откроется соответствующая форма - Add. Так как название события является обязательным полем, то при заполнении этого поля добавление записи будет успешным, в противном случае, запись добавлена не будет и появится соответствующее сообщение с просьбой заполнения данного поля.

4. Тестирование

Для тестирования разработанного приложения было выбрано функциональное тестирование. Этот выбор связан с необходимостью проверить работоспособность разработанной ПС и соответствие заданным требованиям.

Функциональное тестирование является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения (ПО) исходным функциональным требованиям клиента. То есть проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, нужные пользователям.

Преимущества функционального тестирования:

- Функциональное тестирование ПО полностью имитирует фактическое использование системы.
- Позволяет своевременно выявить системные ошибки ПО и, тем самым, избежать множества проблем при работе с ним в дальнейшем.
- Экономия за счет исправления ошибок на более раннем этапе жизненного цикла ПО.[5]

Результаты тестирования приведены в таблице 5.

Таблица 5 - Результаты тестирования

| № | Состав теста | Ожидаемый результат | Наблюдаемый результат |
|---|-------------------------------------|--|---|
| 1 | Нажатие на ячейку календаря | Появление формы с событиями на день (выбранную ячейку) | Появление формы с сообщением «пусто» (Рисунок Б.1) |
| 2 | Нажатие на кнопку добавления записи | Появление формы с полями для заполнения | Появление формы «Добавление события» с пустыми полями (Рисунок Б.2) |
| 3 | Добавление события без названия | Сообщение с просьбой заполнения поля | Сообщение «Введите название события!» (Рисунок Б.3) |
| 4 | Добавление события | После заполнения полей формы | После добавления запись отобразилась на |

| | | | |
|---|---|---|--|
| | | добавления и нажатия на кнопку «Добавить», запись должна отобразиться на экране | форме событий выбранного дня (Рисунки Б.4 и Б.5) |
| 5 | Проверка добавления событий в базу данных | События должны отображаться в БД как записи (объекты) | События занесены в базу данных (Рисунки Б.6 и Б.7) |
| 6 | Изменение существующей записи | На форме после редактирования полей в форме изменения данные должны измениться | Данные события изменены (Рисунки Б.8 и Б.9) |
| 7 | Удаление записи | При нажатии на значок удаления, запись с названием «Концерт» удалена. | Запись удалена (Рисунки Б.10 и Б.11) |

В ходе выполнения тестирования несовпадения ожидаемого и наблюдаемого результата не выявлены. Следовательно, можно сделать вывод, что программа работает корректно. Скриншоты с результатами тестирования приведены в Приложении Б.

Заключение

В процессе прохождения учебной (проектно-технологической) практики были выполнены следующие задачи:

- проанализированы требования к разрабатываемому программному продукту;
- рассмотрено функциональное назначение программы с помощью построения и анализа диаграммы вариантов использования UML;
- разработан алгоритм решения поставленной задачи;
- написан и протестирован код разработанного приложения.

В процессе анализа требований были проанализированы и сформулированы требования к интерфейсу пользователя, к структуре данных и к программным средствам.

При выполнении практической работы была разработана модель пользовательского интерфейса. В ходе реализации программы была создана диаграмма деятельности, которая наглядно передает работу фрагмента программы.

В ходе функционального тестирования, была проверена корректная работа разработанного приложения при осуществлении различных действий пользователя.

Также в период прохождения практики был получен сертификат Национального Открытого Университета «ИНТУИТ» по курсу «Язык UML 2 в анализе и проектировании программных систем и бизнес-процессов». Сертификат приведен в Приложении В.

Список использованных источников

1. Карпова И.П. Базы данных. Учебное пособие. – Московский государственный Институт электроники и математики (Технический университет). – М., 2009
2. 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.
3. IndexedDB: [сайт]. URL: <https://javascript.info/indexeddb> (дата обращения: 10.07.22)
4. Использование IndexedDb – Интерфейсы веб API | MDN:[сайт]. URL:https://developer.mozilla.org/ru/docs/Web/API/IndexedDB_API/Using_IndexedDB#getting_data_from_the_database
5. Функциональное тестирование: [сайт]. URL: <https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/> (дата обращения: 09.07.22)

Приложение А.
ТЕКСТ ПРОГРАММЫ

Файл index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="icon" href="icons/calendar.ico" type="image/x-icon"/>
  <link rel="stylesheet" href="css/styleM.css">
  <link rel="stylesheet" href="css/styleShow.css">
  <link rel="stylesheet" href="css/styleAddChange.css">
  <title>Daily planner</title>
</head>
<body>
  <div class="mainTable">
    <table id="calendar2">
      <thead>
        <tr id="headTr"><td>
          
        </td>
        <div class="d">
          <td colspan="2" style="text-align:right"></td>
          <td colspan="3" style="text-align:center"></td>
        </div>
        <td>
          
        </td>
      </tr>
      <div class="head">
        <tr>
          <td>Пн</td>
          <td>Вт</td>
          <td>Ср</td>
          <td>Чт</td>
          <td>Пт</td>
          <td>Сб</td>
          <td>Вс</td>
        </tr>
      </div>
    </thead>
    <tbody>
    </tbody>
  </table>
</div>
<div class="formAdd" id="formAdd" style="display:none">
  <div class="formAddAll">
    <div class="close">
      <div class="closeBut" id="closeAdd">
        
      </div>
    </div>
    <div class="FormTitle">Добавление события</div>
    <div class="FormEvent">
      <div class="InfoPAdd" id="title">
        <div>Название:</div>
        <textarea name="Title" id="titleText" cols="40" rows="2"></textarea>
        <div id="errorTitle" style="display:none">Введите название события!</div>
      </div>
      <div class="InfoPAdd" id="place">
        <div>Место:</div>
        <textarea name="Place" id="placeText" cols="40" rows="2"></textarea>
      </div>
      <div class="InfoPAdd" id="time">
        <div>Время:</div>
        <input type="time" id="timeText"/>
      </div>
      <div class="InfoPAdd" id="people">
        <div>Участники:</div>
        <textarea name="People" id="peopleText" cols="40" rows="3"></textarea>
      </div>
      <div class="InfoPAdd" id="info">
        <div>Дополнительная информация:</div>
        <textarea name="Info" id="infoText" cols="40" rows="4"></textarea>
      </div>
      <div class="addInfo">
```

```

        <button type="button" id="addInfo" onclick="Add()">Добавить</button>
    </div>
</div>
</div>
</div>
<div class="formChange" id="formChange" style="display:none">
    <div class="formChangeAll">
        <div class="close">
            <div class="closeBut" id="closeChange">
                
            </div>
        </div>
        <div class="FormTitle">Изменение события</div>
        <div class="FormEvent">
            <div class="InfoPChange" id="titleC">
                <div>Название:</div>
                <textarea name="Title" id="titleTextC" cols="40" rows="2"></textarea>
                <div id="errorTitleC" style="display:none">Введите название события!</div>
            </div>
            <div class="InfoPChange" id="placeC">
                <div>Место:</div>
                <textarea name="Place" id="placeTextC" cols="40" rows="2"></textarea>
            </div>
            <div class="InfoPChange" id="timeC">
                <div>Время:</div>
                <input type="time" id="timeTextC"/>
            </div>
            <div class="InfoPChange" id="peopleC">
                <div>Участники:</div>
                <textarea name="People" id="peopleTextC" cols="40" rows="3"></textarea>
            </div>
            <div class="InfoPChange" id="infoC">
                <div>Дополнительная информация:</div>
                <textarea name="Info" id="infoTextC" cols="40" rows="4"></textarea>
            </div>
            <div class="changeInfo">
                <button type="button" id="changeInfo" onclick="Change()">Изменить</button>
            </div>
        </div>
    </div>
</div>
</div>
<div class="formShow" id="formShow" style="display:none">
    <div class="formShowAll">
        <div class="closeBut" id="closeShow">
            
        </div>
        <div class="DateNow" id="DateNow"></div>
        <div id="text">пусто</div>
        <div class="dinamicNotes" id="dinamicNotes"></div>
        <div class="addBut" id="addBut">
            
        </div>
    </div>
</div>
</div>

<script src="js/DataBase.js"></script>
<script src="js/Calendar.js"></script>
<script src="js/FormAdd.js"></script>
<script src="js/FormChange.js"></script>
<script src="js/FormShow.js"></script>
</body>
</html>

```

Файл Calendar.js

```

var dateDay; // дата-подпись календаря
var dateYear; // год-подпись календаря
function Calendar(id, year, month) {
    var Dlast = new Date(year, month + 1, 0).getDate(),
        D = new Date(year, month, Dlast),
        DNlast = new Date(D.getFullYear(), D.getMonth(), Dlast).getDay(),
        DNfirst = new Date(D.getFullYear(), D.getMonth(), 1).getDay(),
        calendar = '<tr>',
        month = {
            1: 'Январь',
            2: 'Февраль',
            3: 'Март',

```



```

        ) - 1
    );
};
// переключатель плюс месяц
document.querySelector(
    '#calendar2 thead tr:nth-child(1) td:nth-child(4)'
).onclick = function () {
    Calendar(
        'calendar2',
        document.querySelector('#calendar2 thead td:nth-child(3)').dataset.year,
        parseFloat(
            document.querySelector('#calendar2 thead td:nth-child(2)').dataset
                .month
        ) + 1
    );
};
//преобразование ключа для вывода на экран
function getKeyByValue(object, value) {
    var num = Object.keys(object).find((key) => object[key] === value);
    if (num.length < 2) {
        num = '0' + num;
    }
    return num;
}
}

```

Файл DataBase.js

```

let db;
let index; //индекс для поиска по дате
let notes;
let note;
let dbReq = indexedDB.open('myNotes', 1); //название бд и версия
//срабатывает, если нет базы данных, инициализация
dbReq.onupgradeneeded = (event) => {
    // Зададим переменной db ссылку на базу данных
    db = event.target.result;
    // Создадим хранилище объектов с именем notes.
    notes = db.createObjectStore('notes', { keyPath: 'title' });
    index = notes.createIndex('day_idx', 'day');
};
dbReq.onsuccess = (event) => {
    db = event.target.result;
};
dbReq.onerror = (event) => {
    alert('error opening database ' + event.target.errorCode);
    //indexedDB.deleteDatabase('myNotes');
};

```

Файл FormAdd.js

```

let fl = false; //флаг для отслеживания добавления записи
document.querySelector('#closeAdd').onclick = function () {
    document.getElementById('titleText').value = '';
    document.getElementById('placeText').value = '';
    document.getElementById('timeText').value = '';
    document.getElementById('peopleText').value = '';
    document.getElementById('infoText').value = '';
    document.getElementById('formAdd').style.display = 'none';
};
function Add() {
    let titleT = document.getElementById('titleText').value;
    if (titleT == '') {
        document.getElementById('errorTitle').style.display = 'block';
        return;
    } else document.getElementById('errorTitle').style.display = 'none';
    let placeT =
        document.getElementById('placeText').value == ''
        ? ''
        : document.getElementById('placeText').value;
    let timeT =
        document.getElementById('timeText').value == ''
        ? ''
        : document.getElementById('timeText').value;
    let peopleT =
        document.getElementById('peopleText').value == ''
        ? ''
        : document.getElementById('peopleText').value;
}

```

```

let infoT =
    document.getElementById('infoText').value == ''
    ? ''
    : document.getElementById('infoText').value;
let transact = db.transaction('notes', 'readwrite');
let store = transact.objectStore('notes');
note = {
    day: dateDay,
    year: dateYear,
    title: titleT,
    place: placeT,
    time: timeT,
    people: peopleT,
    info: infoT,
};
console.log(note);
store.add(note);
transact.oncomplete = () => {
    console.log('stored note!');
    CreateNoteHtml(note);
    document.getElementById('formAdd').style.display = 'none';
};
transact.onerror = (event) => {
    if (
        !confirm(
            `Событие ${note.title} уже есть` +
            'error storing note ' +
            event.target.errorCode
        )
    ) {
        document.getElementById('formAdd').style.display = 'none';
    }
};
console.log(note);
if (!fl) document.getElementById('text').style.display = 'none';
document.getElementById('titleText').value = '';
document.getElementById('placeText').value = '';
document.getElementById('timeText').value = '';
document.getElementById('peopleText').value = '';
document.getElementById('infoText').value = '';
}
function CreateNoteHtml(part) {
    let parent = document.querySelector('#dinamicNotes');
    let div = document.createElement('div');
    div.className = 'note';
    div.id = 'note_' + part.title;
    let partIcons =
        '<div class="Icons">' +
        '<div class="EditBut">' +
        '' +
        '</div>' +
        '<div class="DeleteBut" >' +
        '' +
        '</div>' +
        '</div>';
    let partInfo =
        '<div class="Info" id="Info">' +
        '<div class="InnerShow" id="TitleShow">' +
        '<div class="Pole">Название:</div><div class="Res" id="${part.title}">${part.title}</div>' +
        '</div>' +
        '<div class="InnerShow" id="PlaceShow">' +
        '<div class="Pole">Место:</div><div class="Res">${part.place}</div>' +
        '</div>' +
        '<div class="InnerShow" id="TimeShow">' +
        '<div class="Pole">Время:</div><div class="Res">${part.time}</div>' +
        '</div>' +
        '<div class="InnerShow" id="PeopleShow">' +
        '<div class="Pole">Участники:</div><div class="Res">${part.people}</div>' +
        '</div>' +
        '<div class="InnerShow" id="InfoShow">' +
        '<div class="Pole">Дополнительная информация:</div><div class="Res">${part.info}</div>' +
        '</div>' +
        '</div>';
    div.innerHTML = partInfo + partIcons;
    parent.appendChild(div);
}

```

```

    AddOnClickEvents();
}

```

Файл FormChange.js

```

document.querySelector('#closeChange').onclick = function () {
    document.getElementById('titleTextC').value = '';
    document.getElementById('placeTextC').value = '';
    document.getElementById('timeTextC').value = '';
    document.getElementById('peopleTextC').value = '';
    document.getElementById('infoTextC').value = '';
    document.getElementById('formChange').style.display = 'none';
};
function Change() {
    let titleT = document.getElementById('titleTextC').value;
    if (titleT == '') {
        document.getElementById('errorTitleC').style.display = 'block';
        return;
    } else document.getElementById('errorTitleC').style.display = 'none';
    let placeT =
        document.getElementById('placeTextC').value == ''
        ? ''
        : document.getElementById('placeTextC').value;
    let timeT =
        document.getElementById('timeTextC').value == ''
        ? ''
        : document.getElementById('timeTextC').value;
    let peopleT =
        document.getElementById('peopleTextC').value == ''
        ? ''
        : document.getElementById('peopleTextC').value;
    let infoT =
        document.getElementById('infoTextC').value == ''
        ? ''
        : document.getElementById('infoTextC').value;
    let transact = db.transaction('notes', 'readwrite');
    let store = transact.objectStore('notes');
    note = {
        day: dateDay,
        year: dateYear,
        title: titleT,
        place: placeT,
        time: timeT,
        people: peopleT,
        info: infoT,
    };
    console.log(note);
    store.add(note);
    transact.oncomplete = () => {
        console.log('stored note!');
        RewriteNoteHtml(note);
        console.log(document.getElementById('formChange').name);
        Delete(document.getElementById('formChange').name);
        document.getElementById('formChange').style.display = 'none';
    };
    transact.onerror = (event) => {
        if (
            !confirm(
                `Событие ${note.title} уже есть` +
                'error storing note ' +
                event.target.errorCode
            )
        ) {
            document.getElementById('formChange').style.display = 'none';
        }
    };
    console.log(note);
    if (!fl) document.getElementById('text').style.display = 'none';
}
function RewriteNoteHtml(part) {
    lastTitle = document.getElementById('formChange').name;
    let parent = document.querySelector('#dynamicNotes');
    let div = document.getElementById(
        `note_${document.getElementById('formChange').name}`
    );
    console.log(`note_${document.getElementById('formChange').name}`);
}

```

```

div.innerHTML = '';
let partIcons =
  '<div class="Icons">' +
  '<div class="EditBut">' +
  `` +
  '</div>' +
  '<div class="DeleteBut" >' +
  `` +
  '</div>' +
  '</div>';
let partInfo =
  '<div class="Info" id="Info">' +
  '<div class="InnerShow" id="TitleShow">' +
  `<div class="Pole">Название:</div><div class="Res" id="${part.title}">${part.title}</div>` +
  '</div>' +
  '<div class="InnerShow" id="PlaceShow">' +
  `<div class="Pole">Место:</div><div class="Res">${part.place}</div>` +
  '</div>' +
  '<div class="InnerShow" id="TimeShow">' +
  `<div class="Pole">Время:</div><div class="Res">${part.time}</div>` +
  '</div>' +
  '<div class="InnerShow" id="PeopleShow">' +
  `<div class="Pole">Участники:</div><div class="Res">${part.people}</div>` +
  '</div>' +
  '<div class="InnerShow" id="InfoShow">' +
  `<div class="Pole">Дополнительная информация:</div><div class="Res">${part.info}</div>` +
  '</div>' +
  '</div>';
div.innerHTML = partInfo + partIcons;
div.id = 'note_' + part.title;
AddOnClickEvents();
}
function Delete(title) {
  console.log(title);
  console.log('find delete');
  let tr = db.transaction('notes', 'readwrite');
  let store = tr.objectStore('notes');
  let request = store.delete(title);
  request.onsuccess = function () {
    console.log('Удалено ' + request.result);
  };
};
}

```

Файл FormShow.js

```

document.querySelector('#closeShow').onclick = function () {
  document.getElementById('formShow').style.display = 'none';
  document.querySelector('#dinamicNotes').innerHTML = '';
  document.getElementById('text').style.display = '';
};

document.querySelector('#addBut').onclick = function () {
  document.getElementById('formAdd').style.display = 'block';
  document.getElementById('formAdd').style.zIndex = '2';
  document.getElementById('formShow').style.zIndex = '1';

  console.log('add');
};

function EditNote(title) {
  document.getElementById('formChange').style.display = 'block';
  console.log(title);
  let tr = db.transaction('notes', 'readwrite');
  let store = tr.objectStore('notes');
  let request = store.openCursor(title);
  request.onsuccess = function () {
    let cursor = request.result;
    if (cursor) {
      let key = cursor.key;
      let val = cursor.value;
      console.log(key, val);
      document.getElementById('titleTextC').value = val.title;
      document.getElementById('placeTextC').value = val.place;
      document.getElementById('timeTextC').value = val.time;
      document.getElementById('peopleTextC').value = val.people;
      document.getElementById('infoTextC').value = val.info;
    }
  };
}

```

```

        document.getElementById('formChange').name = val.title;
        console.log(document.getElementById('formChange').name);
    }
    console.log('Получено ' + request.result);
};
}
function DeleteNote(title) {
    console.log(title);
    console.log('find delete');
    let tr = db.transaction('notes', 'readwrite');
    let store = tr.objectStore('notes');
    let request = store.delete(title);
    request.onsuccess = function () {
        console.log('Удалено ' + request.result);
    };
    document.getElementById(`note_${title}`).remove();
}

function GetNote(key) {
    //key - день
    console.log('shhhooooooooow' + ' ' + typeof key);
    let tr = db.transaction('notes', 'readwrite');
    let store = tr.objectStore('notes');
    let dayInd = store.index('day_idx');
    let find = dayInd.getAll(key);
    find.onsuccess = function () {
        if (find.result.length !== 0) {
            console.log('Notes ', find.result);
            document.getElementById('text').style.display = 'none';
            for (let i = 0; i < find.result.length; i++) {
                CreateNoteHtml(find.result[i]);
            }
        } else {
            console.log('No notes');
        }
    };
    tr.oncomplete = function () {
        console.log('Transaction is complete');
    };
}

function AddOnClickEvents() {
    var deletes = document.getElementsByClassName('imgDelete');
    console.log(deletes);
    for (let del of deletes) {
        del.onclick = function () {
            DeleteNote(del.alt);
        };
    }
    var edits = document.getElementsByClassName('imgEdit');
    console.log(edits);
    for (let ed of edits) {
        ed.onclick = function () {
            EditNote(ed.alt);
        };
    }
}
}

```

Файл styleAddChange.css

```

.formAdd, .formChange {
    width:100%;
    min-height:100%;
    background-color: rgba(0,0,0,0.5);
    overflow:hidden;
    position:fixed;
    z-index: 2;
    top:0px;
}
.formAddAll, .formChangeAll{
    margin:40px auto 0px auto;
    width:350px;
    height: 500px;
    padding:10px;
    background-color: #e6ebeb;
    border-radius:10px;
    box-shadow: 0px 0px 10px #000;
}

```

```

}
.FormTitle{
    text-align: center;
    font-size: 20px;
    font-weight: bold;
    color: rgb(44, 86, 122);
}
.addInfo, .changeInfo{
    margin: 10px;
    text-align: center;
}
#addInfo, #changeInfo{
    padding: 10px;
    font-size: 16px;
    font-weight: bold;
    border-radius: 5px;
    background-color:rgb(116, 186, 193);
    border: 2px solid rgb(116, 186, 193);
    color: rgb(44, 86, 122);
}
#addInfo:hover, #changeInfo:hover{
    background: #bfdada;
}
.InfoPAdd{
    margin: 10px;
    display: flex;
    flex-direction: column;
    justify-content: center;
}
.InfoPChange{
    margin: 10px;
    display: flex;
    flex-direction: column;
    justify-content: center;
}
.InfoPAdd div{
    color: rgb(44, 86, 122);
}
.InfoPChange div{
    color: rgb(44, 86, 122);
}
.InfoPAdd input, .InfoPChange input{
    margin: 5px 0;
    outline: none;
}
.InfoPAdd input:focus, .InfoPChange input:focus{
    border: 1px solid rgb(116, 186, 193);
}
#errorTitle{
    color:rgb(231, 140, 92);
    font-size: 16px;
}
#errorTitleC{
    color:rgb(231, 140, 92);
    font-size: 16px;
}

```

Файл styleM.css

```

* {
    margin: 0;
    padding: 0;
}
body{
    background: radial-gradient(circle, rgba(165,245,157,1) 0%, rgba(107,210,222,1) 100%);
}
#calendar2 {
    width: 80%;
    line-height: 30px;
    font-size: 24px;
    table-layout: fixed;
    font-weight: bold;
    margin-top: 25px;
}
#calendar2 thead tr:last-child {
    font-size: 24px;
    color: rgb(85, 85, 85);
}

```

```

}
#calendar2 thead tr:nth-child(1) td:nth-child(2) td:nth-child(3) {
    color: rgb(50, 50, 50);
}
#calendar2 thead tr:nth-child(1) td:nth-child(1):hover, #calendar2 thead tr:nth-child(1) td:nth-
child(4):hover {
    cursor: pointer;
}
#calendar2 tbody td {
    color: rgb(44, 86, 122);
}
#calendar2 tbody td:hover{
    background: #bfdada;
}
#calendar2 tbody td:nth-child(n+6), #calendar2 .holiday {
    color: rgb(231, 140, 92);
}
#calendar2 tbody td.today {
    border:2px double #449576;
    background: none;
}
#calendar2 tbody td.today:hover {
    background: #449576;
}
tbody td{
    background: #F0FFFF;
    border-radius: 5px;
    height: 70px;
}
.mainTable{
    display: flex;
    text-align:center;
    justify-content: center;
    align-items: center;
}
.d{
    text-align: center;
}
textarea{
    margin: 5px 0;
    resize: none;
    outline: none;
    -moz-appearance: none;
}
textarea:focus{
    border: 1px solid rgb(116, 186, 193);
}

```

Файл styleShow.css

```

.formShow{
    width:100%;
    height:100%;
    background-color: rgba(0,0,0,0.5);
    overflow:hidden;
    position:fixed;
    z-index: 1;
    top:0px;
}
.formShowAll{
    margin:40px auto 0px auto;
    width:500px;
    height: 500px;
    padding:10px;
    background-color: #e6ebeb;
    border-radius:10px;
    box-shadow: 0px 0px 10px #000;
    overflow: auto;
}
.closeBut{
    position: fixed;
    top: 45px;
}
#closeShow{
    margin-left:480px;
}
#closeAdd, #closeChange{

```



```

        margin-left: 330px;
    }
    .closeBut:hover{
        cursor: pointer;
    }
    #text{
        color:rgb(187, 183, 183);
        font-size: 50px;
        font-weight: bold;
        display: flex;
        justify-content: center;
    }
    .addBut{
        position: fixed;
        top: 500px;
        margin-left:455px;
        box-shadow: 0 0 5px 5px rgb(128, 134, 134);
        border-radius: 15px;
        width: 40px;
        padding: 5px;
    }
    .addBut:hover{
        background-color: rgb(116, 186, 193);
    }

    ::-webkit-scrollbar {
        width: 0;
    }
    .dinamicNotes{
        display: flex;
        flex-direction: column;
    }
    .note{
        display: flex;
        border-radius:10px;
        box-shadow: 0px 0px 10px rgb(128, 134, 134);
        width: 405px;
        padding: 10px 20px;
        margin: 5px;
    }
    .Icons{
        display: flex;
        align-items: flex-start;
        align-self: stretch;
    }
    .Info{
        width: 380px;
    }
    .Pole{
        color:black;
        font-weight: bold;
    }
    #EditBut{
        margin-top:-5px;
    }
    #DateNow{
        color:rgb(165, 162, 162);
        font-size: 20px;
        margin-left: 5px;
    }
    .EditBut:hover{
        cursor: pointer;
    }
    .DeleteBut:hover{
        cursor: pointer;
    }

```

Приложение Б.

СКРИНШОТЫ РАБОТЫ ПРОГРАММЫ

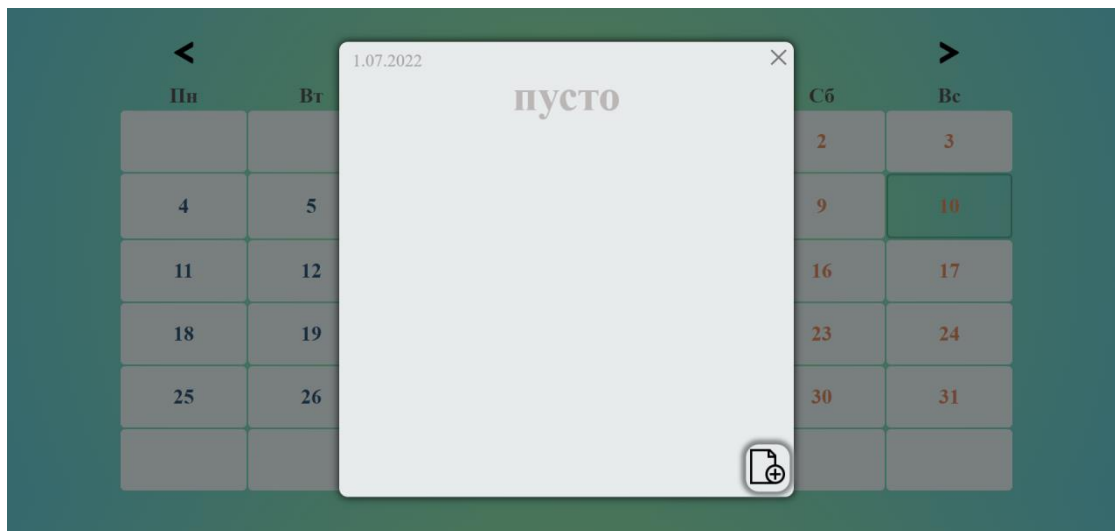


Рисунок Б.1 – Тест 1

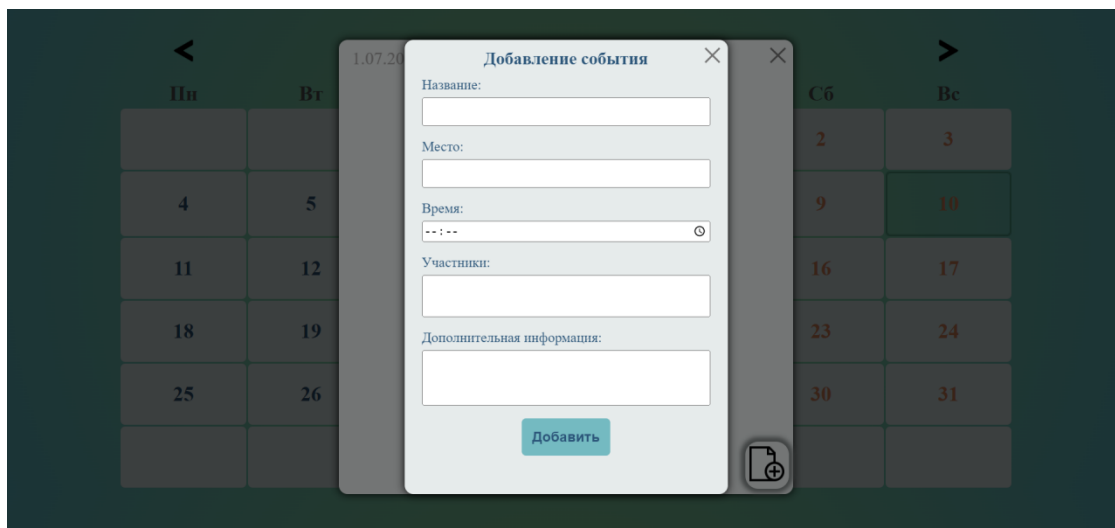


Рисунок Б.2 – Тест 2

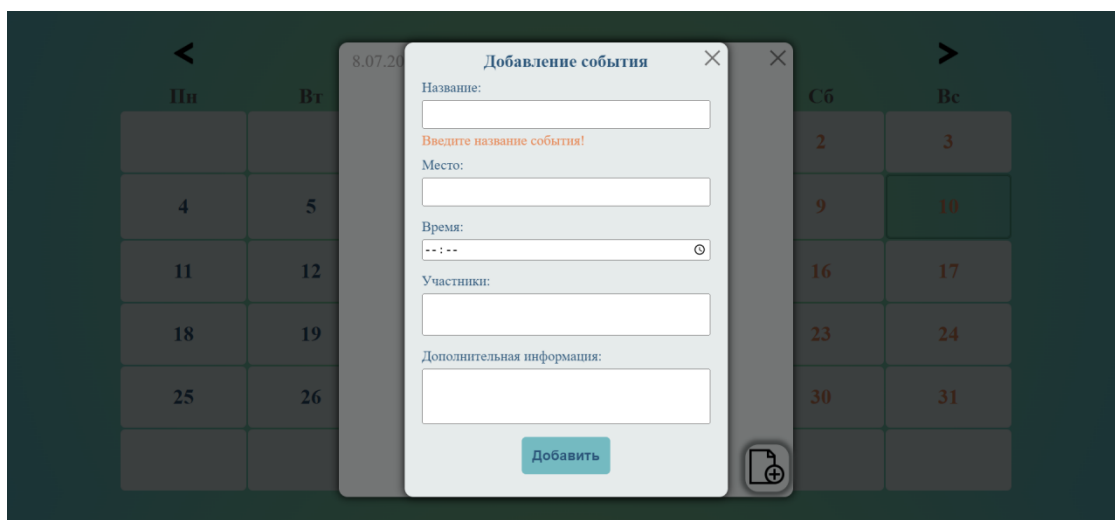


Рисунок Б.3 – Тест 3

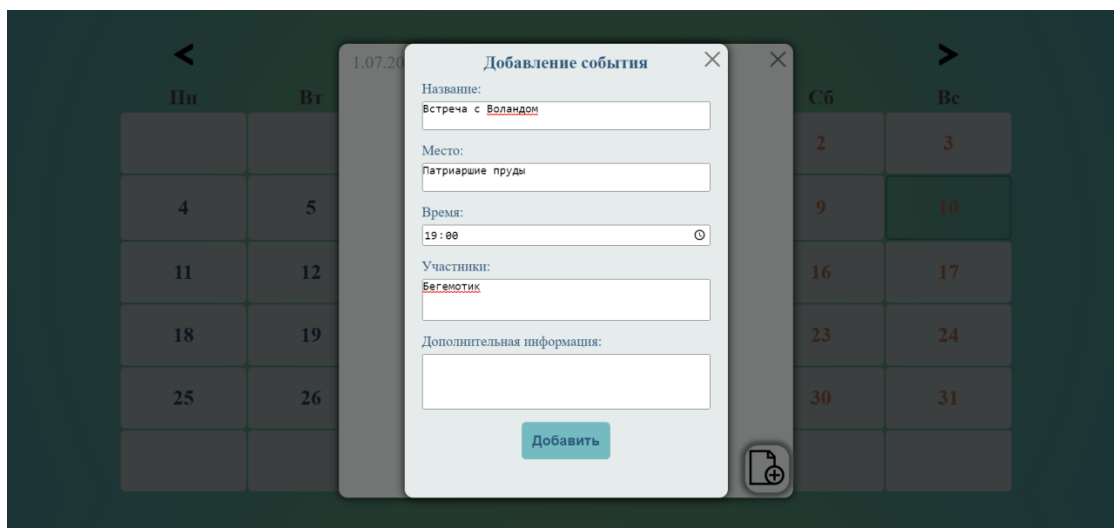


Рисунок Б.4 – Тест 4 (добавление записи)

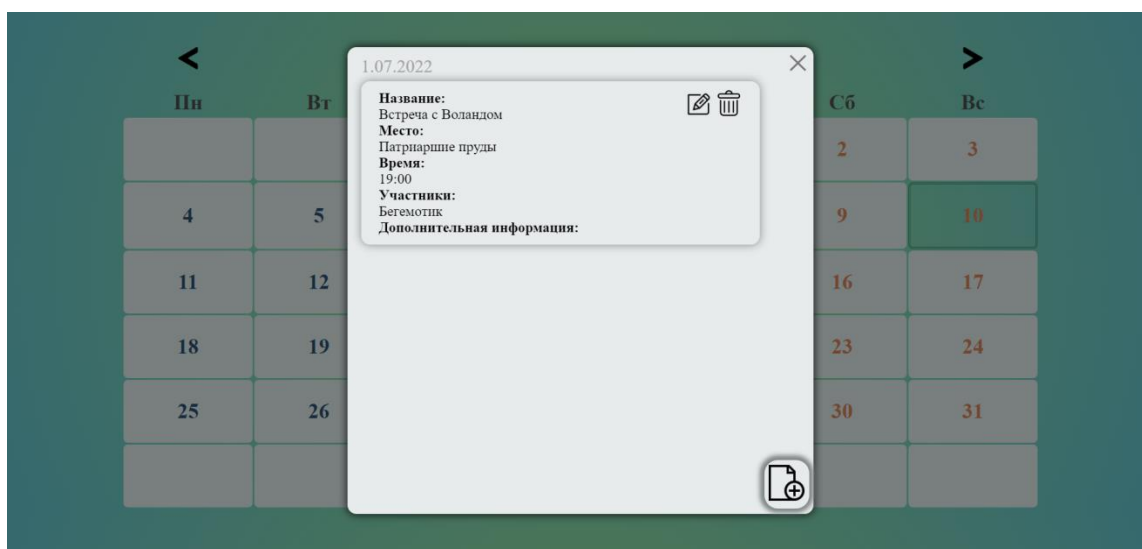


Рисунок Б.5 – Тест 4 (отображение добавленной записи)

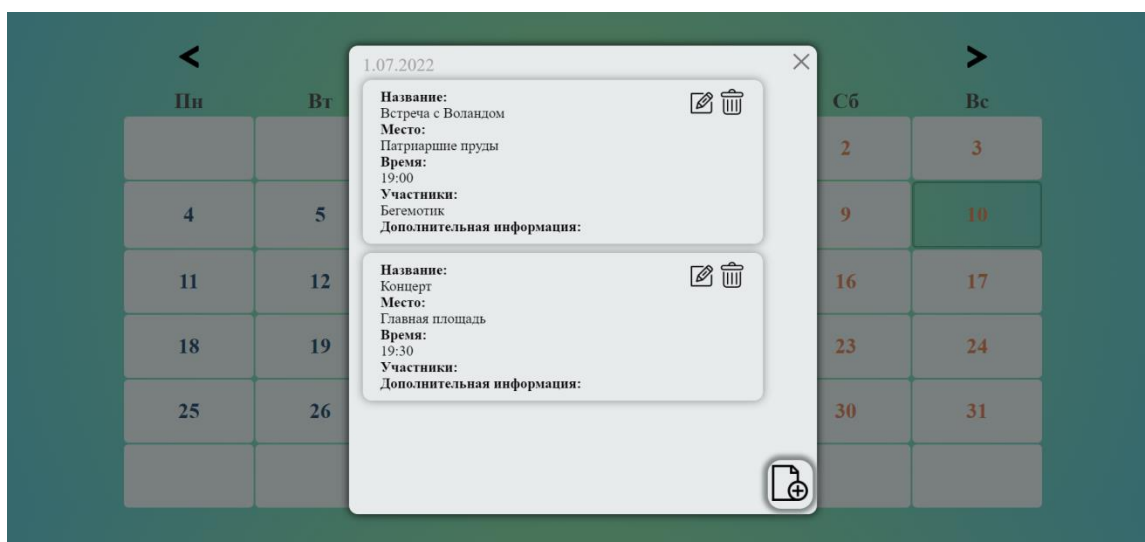


Рисунок Б.6 – Тест 5 (отображение записей)

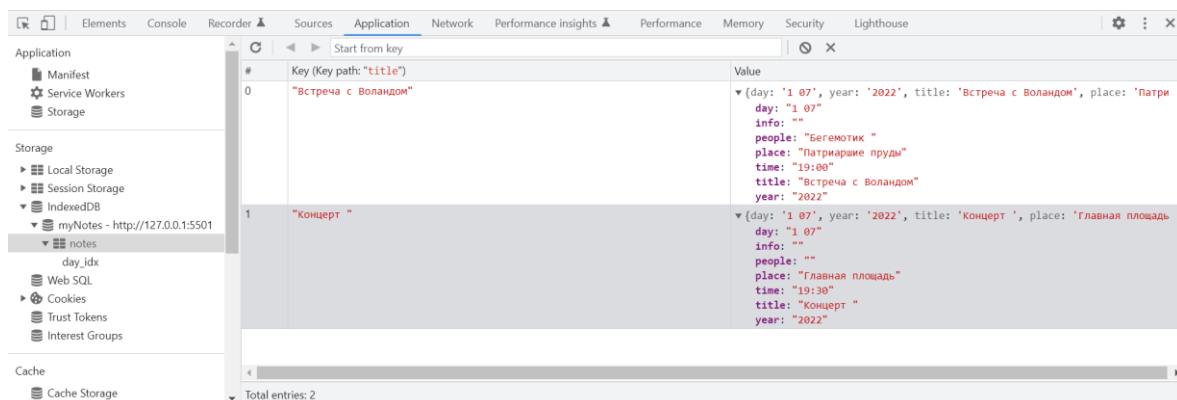


Рисунок Б.7 – Тест 5 (база данных)

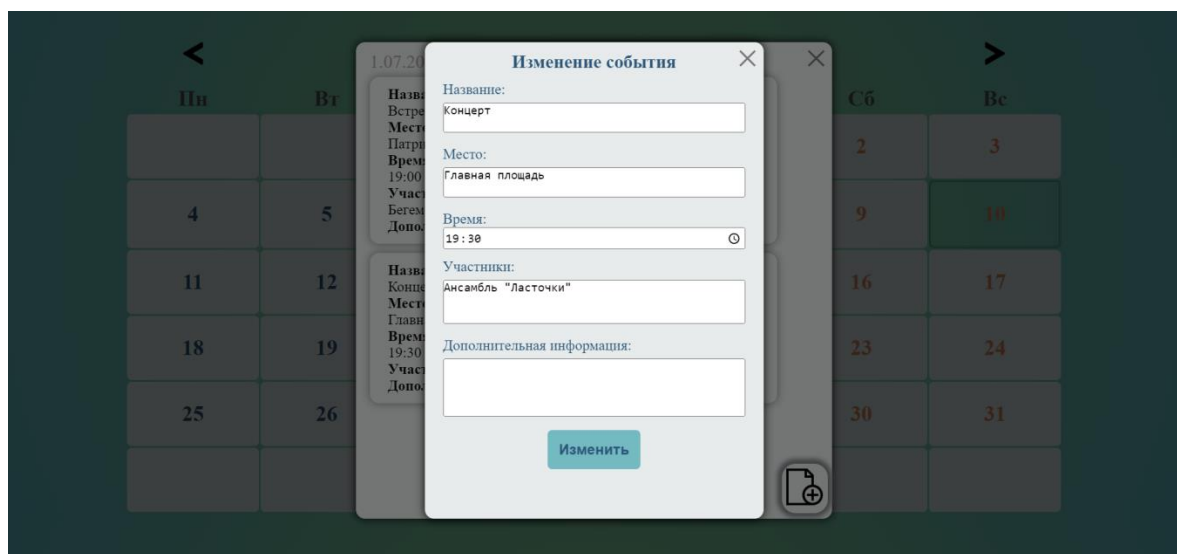


Рисунок Б.8 – Тест 6 (изменение записи)

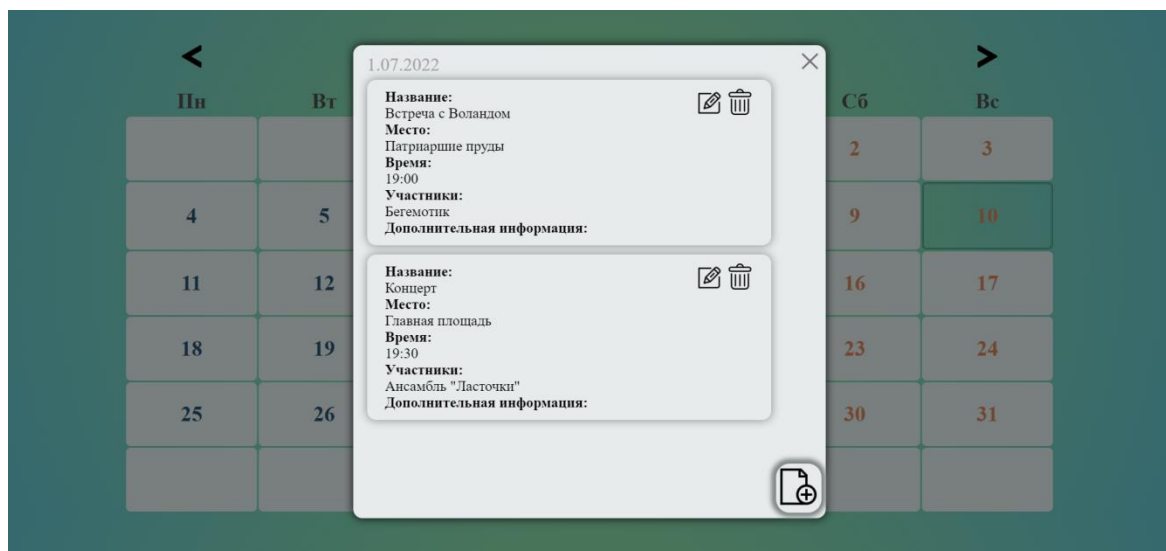


Рисунок Б.9 – Тест 6 (отображение измененной записи)

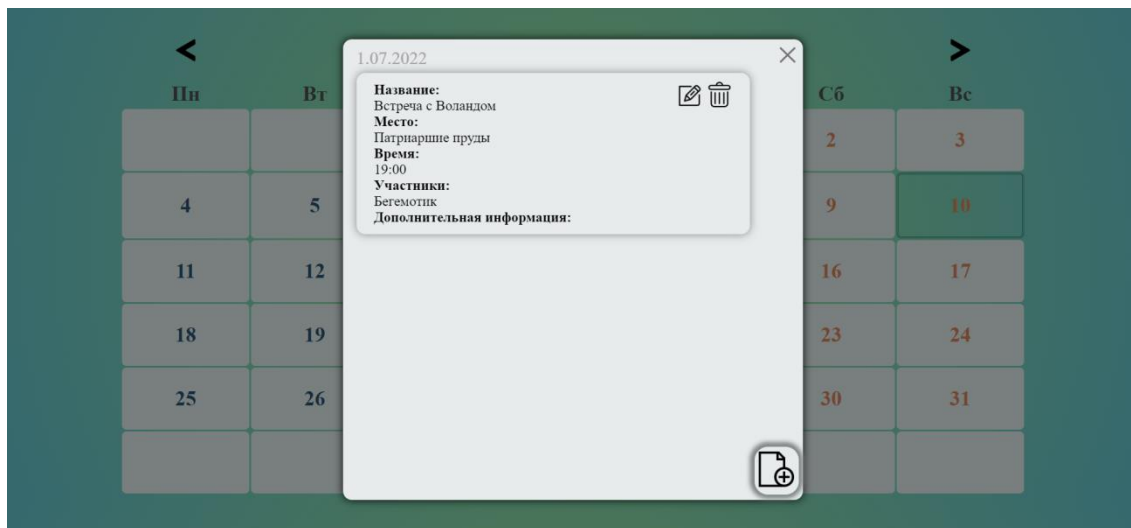


Рисунок Б.10 – Тест 7

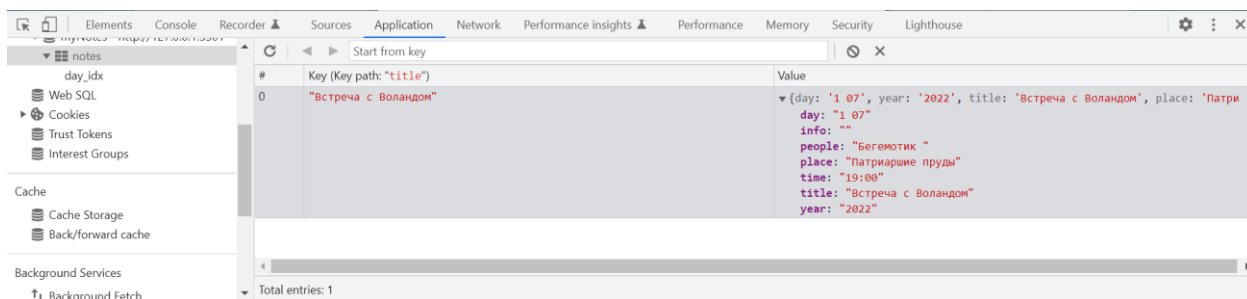


Рисунок Б.11 – Тест 7

Приложение В.
СЕРТИФИКАТ ОТКРЫТОГО УНИВЕРСИТЕТА



Рисунок В.1 - Сертификат ИНТУИТ по курсу «Язык UML 2 в анализе и проектировании программных систем и бизнес-процессов»