**

Колледж космического машиностроения и технологий

КУРСОВОЙ ПРОЕКТ

По МДК.01.02 «Прикладное программирование»

Тема: «Разработка приложения «Ежедневник»»

Выполнил студент

Филиппов Леонид Вячеславович

Группа П2-17

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Проверил преподаватель

Гусятинер Леонид Борисович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

Королёв 2020 г

[Задание на курсовой проект 4](#_Toc43825486)

[Введение 4](#_Toc43825487)

[Глава 1. Теоретическая часть 5](#_Toc43825488)

[1.1. Описание предметной области 5](#_Toc43825489)

[1.1.1. Виды тайм-менеджмента 5](#_Toc43825490)

[1.1.2. Описание библиотеки Django 7](#_Toc43825491)

[1.1.3. Описание языка программирования python 8](#_Toc43825492)

[1.2. Описание существующих разработок 8](#_Toc43825493)

[Глава 2. Проектная часть 12](#_Toc43825494)

[2.1. Диаграмма прецедентов 12](#_Toc43825495)

[2.2. Выбор инструментов 12](#_Toc43825496)

[2.2.1. Библиотека Django 13](#_Toc43825497)

[2.2.2. Интегрированная среда разработки PyCharm 13](#_Toc43825498)

[2.2.3. Интерпретатор Python 13](#_Toc43825499)

[2.2.4. Интерпретатор JavaScript 13](#_Toc43825500)

[2.2.5. Язык гипертекстовой разметки HTML 13](#_Toc43825501)

[2.3. Проектирование сценария 14](#_Toc43825502)

[2.4. Диаграмма классов 15](#_Toc43825503)

[2.4.1. Диаграмма классов для модуля views.py 15](#_Toc43825504)

[2.4.2. Диаграмма классов для модуля models.py 16](#_Toc43825505)

[2.4.3. Диаграмма классов для модуля forms.py 16](#_Toc43825506)

[2.5. Описание главного модуля 17](#_Toc43825507)

[2.6. Описание спецификаций к модулям 17](#_Toc43825508)

[2.7. Описание модулей 19](#_Toc43825509)

[2.8. Описание тестовых наборов модулей 22](#_Toc43825510)

[2.9. Описание применения средств отладки 27](#_Toc43825511)

[2.10. Анализ оптимальности использования памяти и быстродействия 28](#_Toc43825512)

[Глава 3. Эксплуатационная часть 28](#_Toc43825513)

[Заключение 28](#_Toc43825514)

[Список литературы и интернет-источников 28](#_Toc43825515)

[Приложения 28](#_Toc43825516)

# Задание на курсовой проект

Разработка приложения "Ежедневник"

# Введение

Ежедневник – отличный инструмент по планированию дел и тайм-менеджмента. Предварительное планирование дел помогает повысить плодотворность любой деятельности, как личной, так и профессиональной. В современном мире, конечно, лучше использовать электронный ежедневник.

Плюсы электронного ежедневника:

в первую очередь – компактность. Электронный ежедневник не займёт много места.

Практически любой электронный ежедневник – с напоминаниями, что является большим плюсом.

Цель курсового проекта заключается в создании электронного ежедневника.

Для осуществления обозначенной цели служат следующие задачи:

1. Изучение документации Django
2. Создание базы данных SQlite
3. Создание интерфейса
4. Написание кода
5. Описание руководства оператора

# Глава 1. Теоретическая часть

## Описание предметной области

### Виды тайм-менеджмента

1. Метод “Слоны и лягушки”

**Задачи «лягушки»**

Выделяем задачи -«лягушки». «Лягушками» в тайм-менеджменте называют необходимые, но неприятные, постоянно откладывающиеся дела. Как правило, это дела, не требующие много времени на их выполнение.

**Задачи «слоны»**

«Слоны» в тайм-менеджменте — это большие задачи, объемные и сложные.

Задачи “Мамонты”

«Мамонты» — это большие дела, которые долгое время откладывались и совсем потеряли свою актуальность. Такие дела лучше вычеркнуть из списка и не тратить на них свою энергию.

**Задачи «единороги»**

«Розовые слоны» или «единороги» — это дела-мечты, за претворение в жизнь которых мы никак не можем приступить.

**Краткий алгоритм применения техники «Слоны и лягушки»:**

1. Выпишите на лист все свои дела, которые нужно сделать;  
2. Вычеркните из списка дела – «мамонты»;  
3. Выделите зеленым цветом задачи – «лягушки»;  
4. Задачи – «слоны» разбейте на элементарные подзадачи – «бифштексы»;  
5. Для записи подзадач используйте краткую форму:  
«название главной задачи – «слона»/подзадача/дата планируемого выполнения»  
6. Заполнить таблицу – «пинарик»;  
7. Организация ежедневного учета и контроля выполнения задач в таблице – «пинарике».

1. Техника помидора
2. Определитесь с задачей, которую будете выполнять.
3. Поставьте помидор (таймер) на 25 минут.
4. Работайте, ни на что не отвлекаясь, пока таймер не прозвонит. Если что-то отвлекающее возникло у вас в голове, запишите это и немедленно возвращайтесь к работе.
5. Сделайте короткий перерыв (3-5 минут).
6. После каждого 4-го «помидора» сделайте длинный перерыв (15-30 минут).
7. Метод А. А. Любищева

Принцип системы был предельно прост — вести подробный учет времени, будь оно потрачено на научную работу, чтение художественной литературы либо отдых, вроде общения с друзьями. Нужно делать ежедневные записи и в конце месяца отчёт о потраченном времени на что-либо.

1. Карта или фотография рабочего времени

В течение которого фиксируется время, затраченное сотрудником на каждое его значительное действие, совершаемое во время рабочего периода: дня, смены, суток в зависимости от его длительности. Такие действия могут быть как связанны, так и не связанны с его рабочими задачами.

Наблюдение продолжается несколько рабочих дней, чаще без ведома сотрудника. При этом самым действенным способом является съемка на камеру с одновременным наложением времени в углу картинки.  Это существенно облегчает работу аналитика или консультанта. Видео информация в последствии превращается в детализированный отчет о затратах времени в виде понятных, с первого взгляда, заказчику диаграмм и таблицы фотографии рабочего времени сотрудника.  
Как составляется  диаграмма отчета «Карта рабочего времени»:

* выявляются основные блоки времени и затем все время сотрудника объединяется в эти группы согласно полученной видеозаписи.
* рассчитывается точный срез информации о том, на какие группы и виды деятельности и в каких пропорциях тратил этот сотрудник свое рабочее время.

Необходимо несколько повторяющихся записей через двух-трех недельный промежуток. Это связано с тем, что какой-то период его деятельность может быть «моно групповой». Например, в день съемки сотрудник готовил ежегодный отчет и писал его с 9.00 до 18.00 «не вставая», так как опаздывал с его сдачей. Избежать подобных выбросов поможет выборочная съемка через 5, 7 и 14 дней после первого захода.

Результатом этих четырех заходов может быть общая «Карта рабочего времени» сотрудника определенного отдела.

### Описание библиотеки Django

Django — [фреймворк](https://web-creator.ru/articles/about_frameworks) для веб-приложений на языке [Python](https://web-creator.ru/articles/python). Один из основных принципов фреймворка — [DRY](https://web-creator.ru/articles/dry) (don't repeat yourself). Веб-системы на Django строятся из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из заметных архитектурных отличий этого фреймворка от некоторых других (например, [Ruby on Rails](https://web-creator.ru/articles/about_ruby_on_rails)). Также, в отличие от многих других фреймворков, обработчики URL в Django конфигурируются явно (при помощи регулярных выражений), а не автоматически задаются из структуры контроллеров.

Django проектировался для работы под управлением [Apache](https://web-creator.ru/articles/httpd) (с модулем mod\_python) и с использованием [PostgreSQL](https://web-creator.ru/articles/postgresql) в качестве базы данных. В настоящее время, помимо PostgreSQL, Django может работать с другими СУБД: [MySQL](https://web-creator.ru/articles/mysql) ([MariaDB](https://web-creator.ru/articles/mariadb)), SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle. Для работы [с базой данных](https://web-creator.ru/articles/about_databases) Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Архитектура Django похожа на [«Модель-Представление-Контроллер» (MVC)](https://web-creator.ru/articles/mvc). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (Templates). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV).

### Описание языка программирования python

Python — это высокоуровневый язык программирования общего назначения, который используется в том числе и для разработки веб-приложений. Язык ориентирован на повышение производительности разработчика и читаемости кода.

Python поддерживает несколько парадигм программирования: структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. В языке присутствет динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных

## Описание существующих разработок

Существует множество электронных ежедневников, мной были рассмотрены некоторые из них.

Первое рассмотренное решение – “ЛидерТаск”:

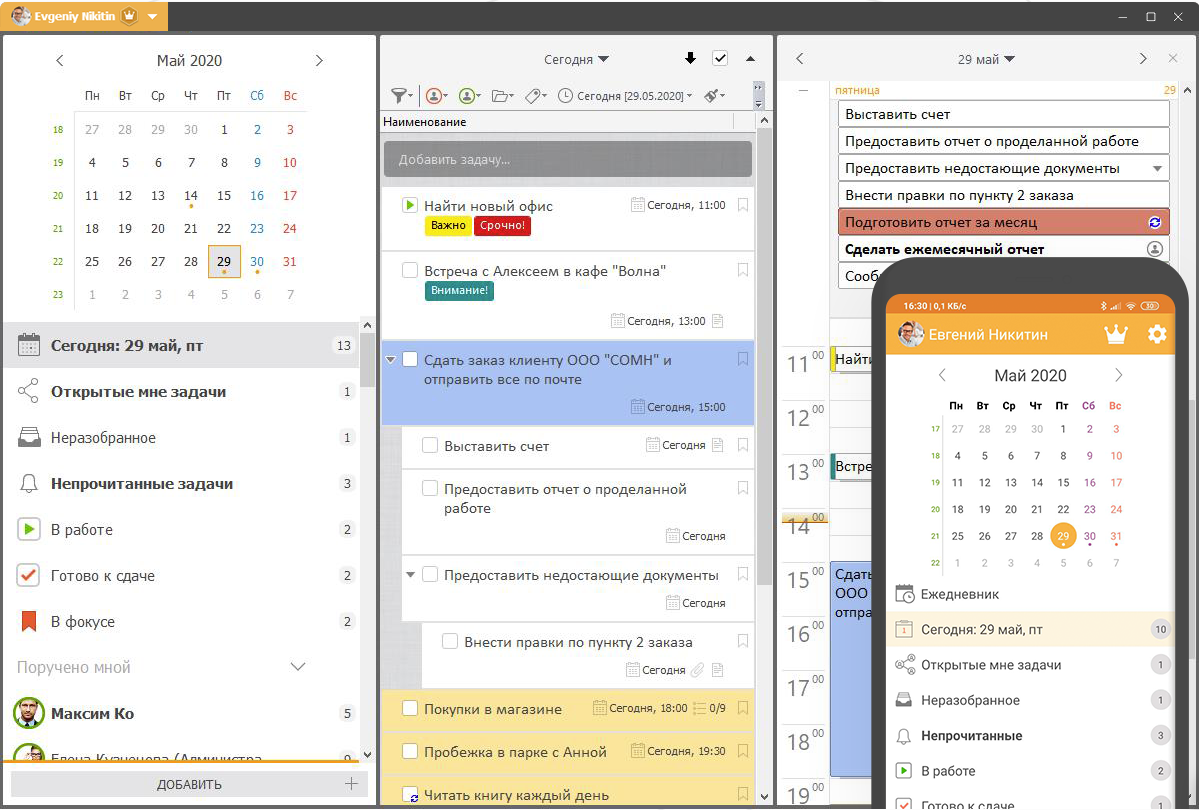


Рисунок 1 ЛидерТаск

В нем есть огромный набор функционала: задачи, проекты, сотрудники, контакты, метки, цвета, общие проекты, доступ к задаче, фокус, статусы, комментарии, файлы, чеклисты, напоминания, хронометраж, фильтрация, сортировка. С таким мощным функционалом вы сможете сделать свою систему тайм-менеджмента, которая отвечает вашим требованиям.

ЛидерТаск обладает методами тайм-менеджмента: Getting Things Done от Дэвида Аллена, Agile, AutoFocus, DIT (Do It Tomorrow), SCRUM, Система управления временем Брайана Трейси, Принцип Парето и многие многие другие.

Достоинства:

* Удобный пк интерфейс
* Много методов тайм-менеджмента
* Разбиение задачи на подзадачи
* Добавление коллег или подчинённых
* Маркеры важно, сложно для задач
* Папки с задачами
* Добавление сотрудников

Недостатки:

* Неудобный мобильный дизайн
* Пробная версия 23 часа
* Приложение работает по платной подписке

Второе рассмотренное решение – приложение “Мой ежедневник”:

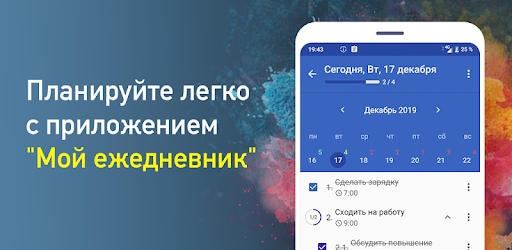


Рисунок 2 Мой ежедневник

Приложение состоит из следующих инструментов:

* Ежедневник
* Календарь
* Планировщик дел
* Напоминания
* Список дел

Достоинства:

* Удобный интерфейс
* Разбиение задачи на подзадачи
* Создание повторяющихся задач
* Метод тайм-менеджмента “Слоны и лягушки”
* Неограниченное количество напоминаний
* Автоматический расчет прогресса
* Существует бесплатная версия

Недостатки:

* Нету версии для ПК
* Используется только один метод тайм-менеджмента
* Нету маркеров для задач
* Бесплатной является ограниченная версия

# Глава 2. Проектная часть

## Диаграмма прецедентов

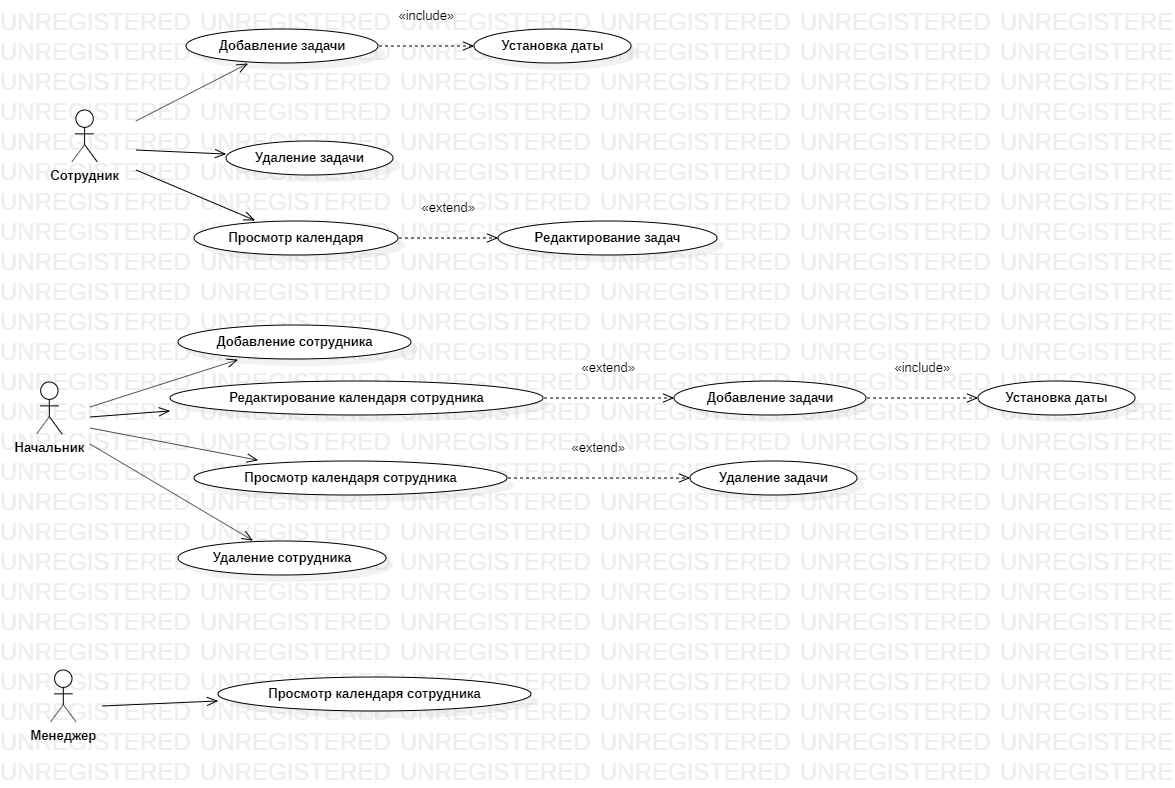


Рисунок 3 Диаграмма прецедентов

По просьбе преподавателя диаграмма прецедентов была спроектирована для бизнес ежедневника, т.к. для обычного ежедневника диаграмма была слишком маленькой.

## Выбор инструментов

При написании курсового проекта были выбраны следующие программные решения: библиотека Django, интегрированная среда разработки PyCharm, интерпретатор Python, интерпретатор JavaScript, язык гипертекстовой разметки HTML

|  |  |  |
| --- | --- | --- |
|  | Python | C# |
| **Простота синтаксиса** | 9 | 6 |
| **Документация на русском** | 6 | 10 |
| **Наличие библиотек** | 10 | 10 |
| **Скорость разработки** | 10 | 5 |
| **Итог:** | 35 | 31 |

|  |  |  |
| --- | --- | --- |
|  | Sublime Text 3 | PyCharm |
| **Интерфейс** | 5 | 8 |
| **Скорость разработки** | 7 | 10 |
| **Документация на русском** | 7 | 5 |
| **Удобство разработки** | 5 | 10 |
| **Итого** | 24 | 33 |

Вывод: в совокупности всех факторов, был выбран язык программирования Python и среда разработки PyCharm.

### Библиотека Django

Использовалась для написания backend’а к приложению “Ежедневник”.

### Интегрированная среда разработки PyCharm

Использовалась для написания кода

### Интерпретатор Python

Использовалась для написания backend’а к приложению “Ежедневник”.

### Интерпретатор JavaScript

Использовался для написания календаря и запросов к backend’у.

### Язык гипертекстовой разметки HTML

Использовался для написания frontend’а к приложению “Ежедневник”.

## Проектирование сценария

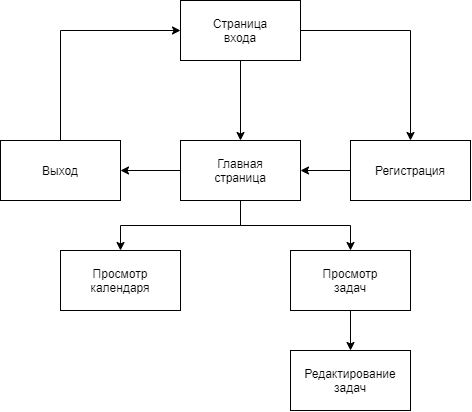


Рисунок 4 Диаграмма сценария

После запуска программы пользователь имеет несколько вариантов действий:

* Авторизация
* Регистрация

После авторизации:

* Просмотр главной страницы
* Просмотр задач
* Добавление задач
* Просмотр календаря
* Выйти из аккаунта

## Диаграмма классов

### Диаграмма классов для модуля views.py

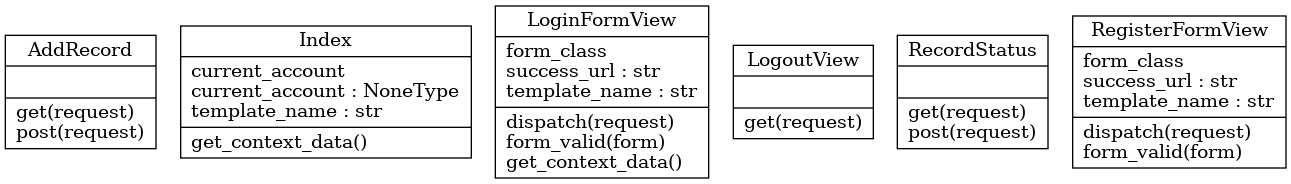


Рисунок 5 Диаграмма классов для модуля views.py

Класс “AddRecod”

Публичные методы:

* get(request) – обрабатывает get запрос, возвращает все записи которые назначены на определённый день в виде json
* post(request) – обрабатывает post запрос, добавляет новую задачу в БД

Класс “Index”

Публичные поля: current\_account, template\_name

Публичные методы:

* get\_context\_data() - функция для обновления контекста

Класс “LoginFormView”

Публичные поля: form, success\_url, template\_name

Публичные методы:

* dispatch(request) - функция для перенаправление на главную страницу, если пользователь уже авторизован
* form\_valid(form) - функция для обработки данных формы, если они правильные
* get\_context\_data() - функция для обновления контекста

Класс “LogoutView”

Публичные методы:

* get() - функция для обработки get запроса, реализует деавторизацию

Класс “RecordStatus”

Публичные методы:

* get(request) - функция для обработки get запроса, возвращает количество задач, количество невыполненных задач, количество выполненных задач
* post(request) - функция для обработки post запроса, который ставит задаче статус выполнено

Класс “RegisterFormView”

Публичные поля: form\_class, success\_url, template\_name;

Публичные методы:

* dispatch(request) - функция для перенаправление на главную страницу, если пользователь уже авторизован
* form\_valid(form) - функция для сохранения данных пользователя, если они правильные

### Диаграмма классов для модуля models.py

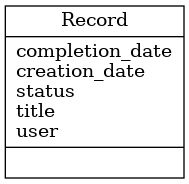


Рисунок 6 Диаграмма классов для модуля models.py

Класс “Record”

Публичные поля: completion\_date, creation\_date, status, title, user

### Диаграмма классов для модуля forms.py

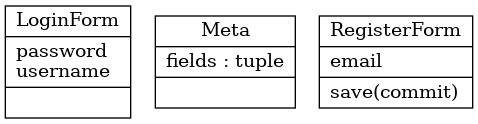


Рисунок 7 Диаграмма классов для модуля forms.py

Класс “LoginForm”

Публичные поля: password, username

Класс “Meta”

Публичные поля: fields

Класс “RegisterForm”

Публичные поля: email

Публичные методы:

* save(commit) – сохраняет email пользователя

## Описание главного модуля

Главный модуль представляет собой файл manage.py.

В главный модуль разработки входит запуск файла views.py, который отвечает за дальнейшее использование приложения.

Программа запускается с помощью файла manage.py командой python manage.py runserver

Данный модуль запускается единожды и является самым первым при запуске приложения. Код приведён в листинге 1.

#### Листинг 1

import os

import sys

def main():

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'dailyPlannerSite.settings')

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError() from exc

execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

main()

## Описание спецификаций к модулям

Описание всех членов модулей курсового проекта

Главный модуль manage.py

В главном модуле вызывается файл views.py

Модуль вывода информации views.py

Get() – обработка get запроса

Post() – обработка post запроса

Form\_valid() – проверка на правильность данных

Dispatch() – определяется тип запроса и вызывается соответствующий метод.

Get\_context\_data() – добавляет объект, который будет доступен в шаблоне

Template\_name – имя шаблона для отображения данных

Current\_account – текущий аккаунт

Form\_class – название формы для обработки данных

Модуль для определения форм

Username – поле для ввода имя пользователя

Password – поле для ввода пароля пользователя

Model – название модели БД, в которую сохранятся данные

Fields – поля формы

Widgets – определение полей формы

Labels – определение названий для полей формы

Модуль посылки запросов и отображения информации calendar.js

Day() – приводить ткущий день к строковому виду

dateRecords() – Отправляет запрос на получение всех записей за определённый день

newElement() – добавляет новую запись на определённый день

next() – метод для перехода на новый месяц в календаре

previous() – метод для перехода на предыдущий месяц в календаре

jump() – метод для перехода на определённый год и месяц в календаре

showCalendar() – метод для отображения календаря

daysInMonth() – метод для определения количества дней в месяце

today – текущий день

currentMonth – текущий месяц

currentYear – текущий код

selectYear – выбранный год

selectMonth – выбранный месяц

months – массив с месяцами

## Описание модулей

В данном пункте будут описаны все оставшиеся модули курсового проекта.

* Центральный модуль

Модуль представляет собой набор классов и запросов. Он отвечает за управление запросами, записью и получением информации из базы данных.

* Модуль для построения таблицы базы данных

Данный модуль представляет собой класс с полями описывающий таблицу БД. За запросы к этой БД отвечает центральный модуль.

* Модуль для создания форм

Данный модуль представляет собой набор классов, в которых указываются поля формы. После чего эти формы используются в шаблонах и обрабатываются в центральном модуле.

* Модуль для вывода информации на экран

Данный модуль представляет собой код на языке HTML и отвечает за графическую часть. Он реализует вывод на экран календаря и окна с задачами.

* Модуль управления запросами

Данный модуль представляет собой код на языке JavaScript и отвечает за управления запросами. Он отправляет запросы в центральный модуль, после чего центральный модуль обрабатывает их и посылает обратно в модуль управления запросами.

Код для модуля управления запросами представлен в листинге 2.

#### Листинг 2

function dateRecords(date) {

let xhr = new XMLHttpRequest(); // Определение переменной запроса xhr.open('GET','http://127.0.0.1:7000/daily\_planner/add\_record/?data=' + day(date), true); // Настройка запроса

xhr.send(); // Отправка запроса

xhr.addEventListener('readystatechange', function(){ // Обработка ответа запроса

if(xhr.readyState == 4 && xhr.status == 200){ // Если ответ положительный

let massage = JSON.parse(xhr.responseText); // Получение данных запроса

let ul = document.getElementById("myUL"); // Переменная контейнер под задачи

ul.innerHTML = ""; // Очистка контейнера

for (var i = 0; i < massage.length; i++) // Цикл по задачам

{

let li = document.createElement("li"); // Создание контейнера под задачу

li.id = massage[i].pk; // Создание id для задачи

li.innerHTML = massage[i].fields.title; // Помещение задачи в контейнер

ul.appendChild(li); // Помещение задаи в контейнер для задач

}

}

})

}

Блоксхема к методу dateRecords()

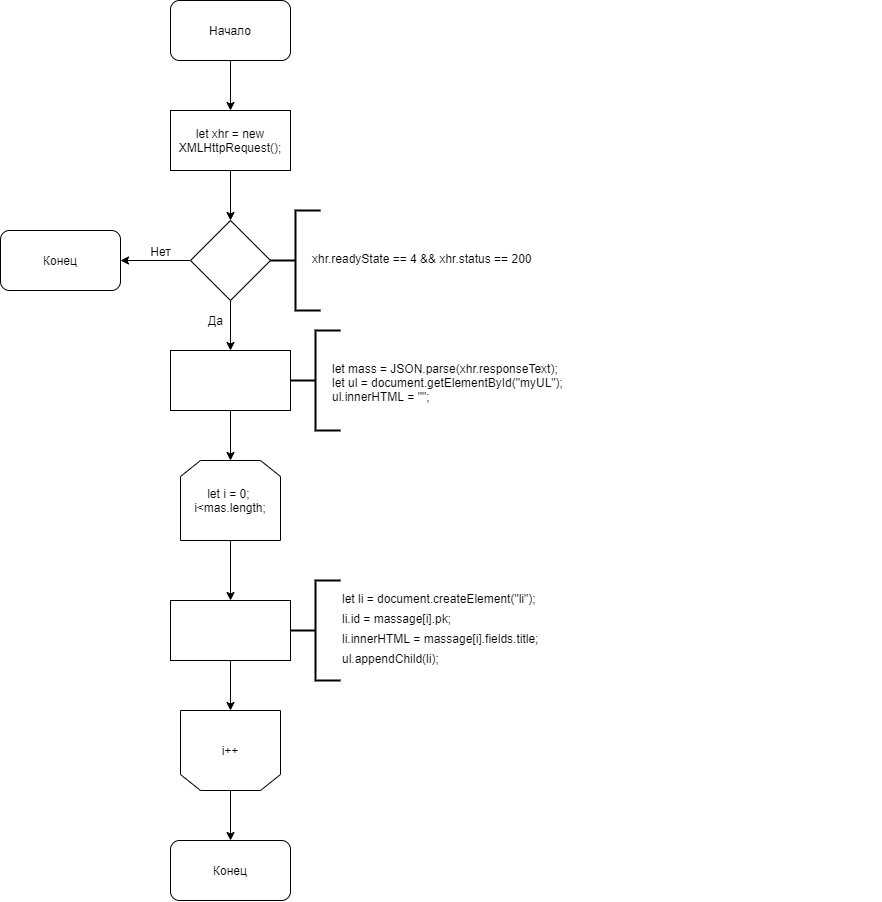


Рисунок 8 Блоксхема к методу dateRecords()

## Описание тестовых наборов модулей

В этом разделе будут продемонстрированы результаты тестирования «черного ящика».

Тест 1. Добавление задачи.

Действия: нажать я ячейку календаря. Добавить новую задачу.

Ожидаемый результат: Эта задача добавится в список задач на этот день.

Результат теста:

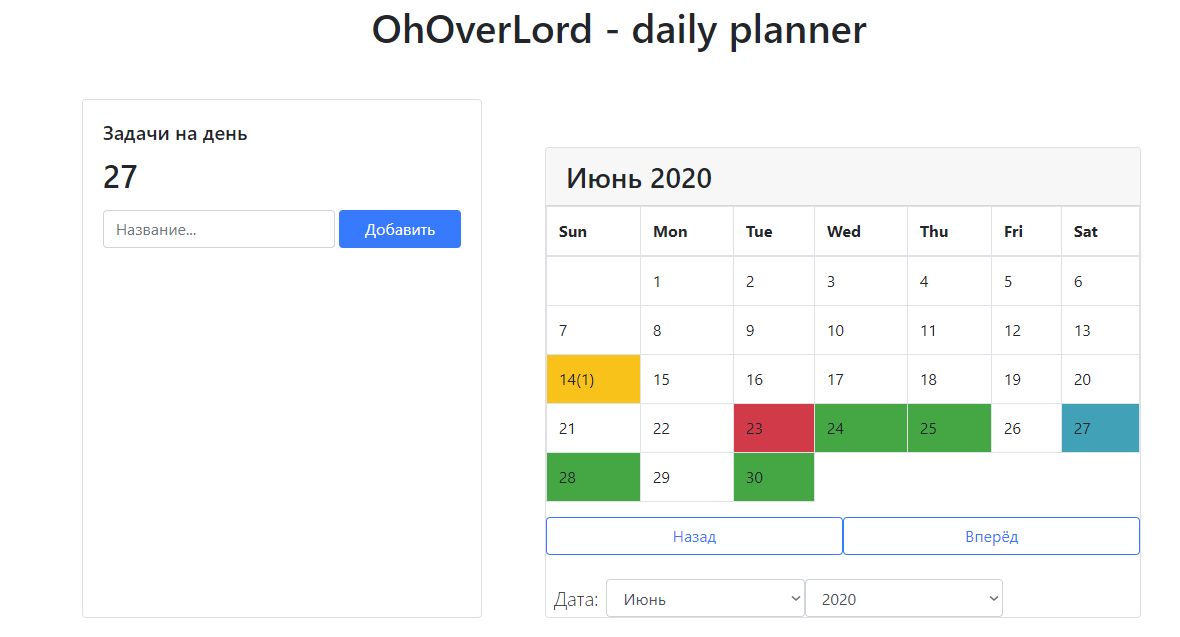


Рисунок 9 Выбор дня в календаре

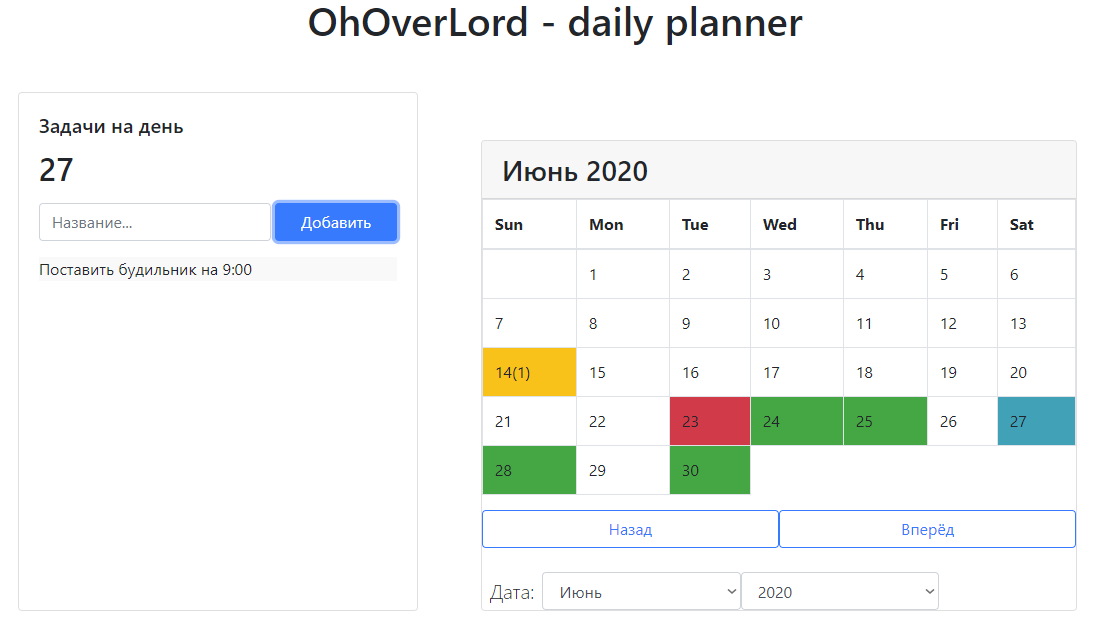


Рисунок 10 Новая задача в списке задач на этот день

Тест 2. Выбор определённого месяца и года.

Действия: выбрать месяц и год в выпадающем списке и перейти на него

Ожидаемый результат: переход на определённый месяц в календаре.

Результат теста:

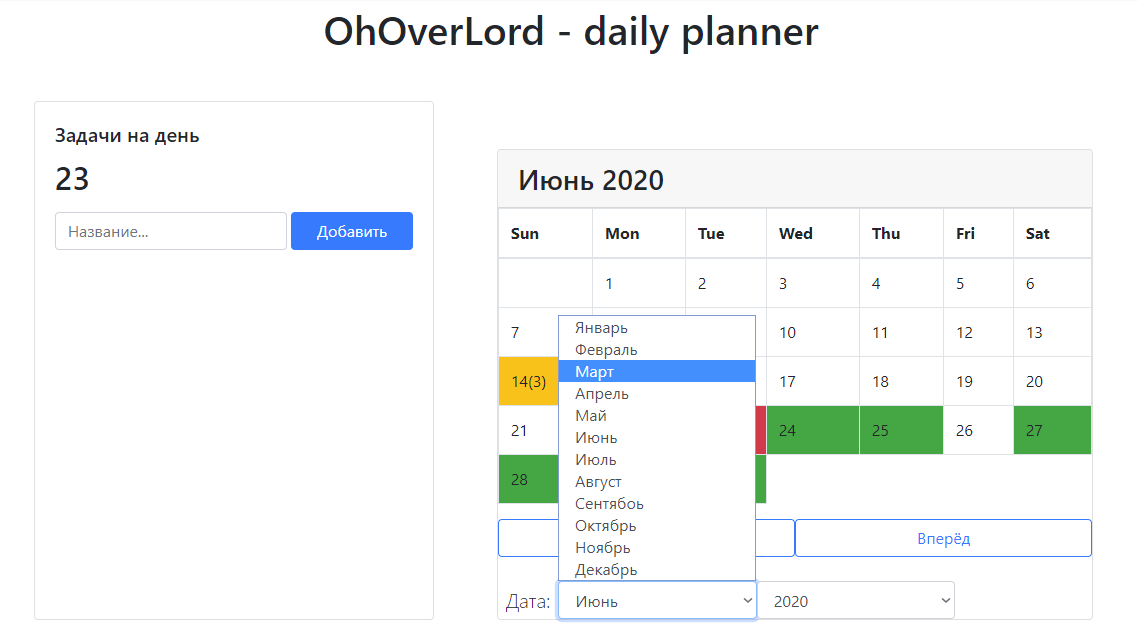


Рисунок 11 Выбор месяца

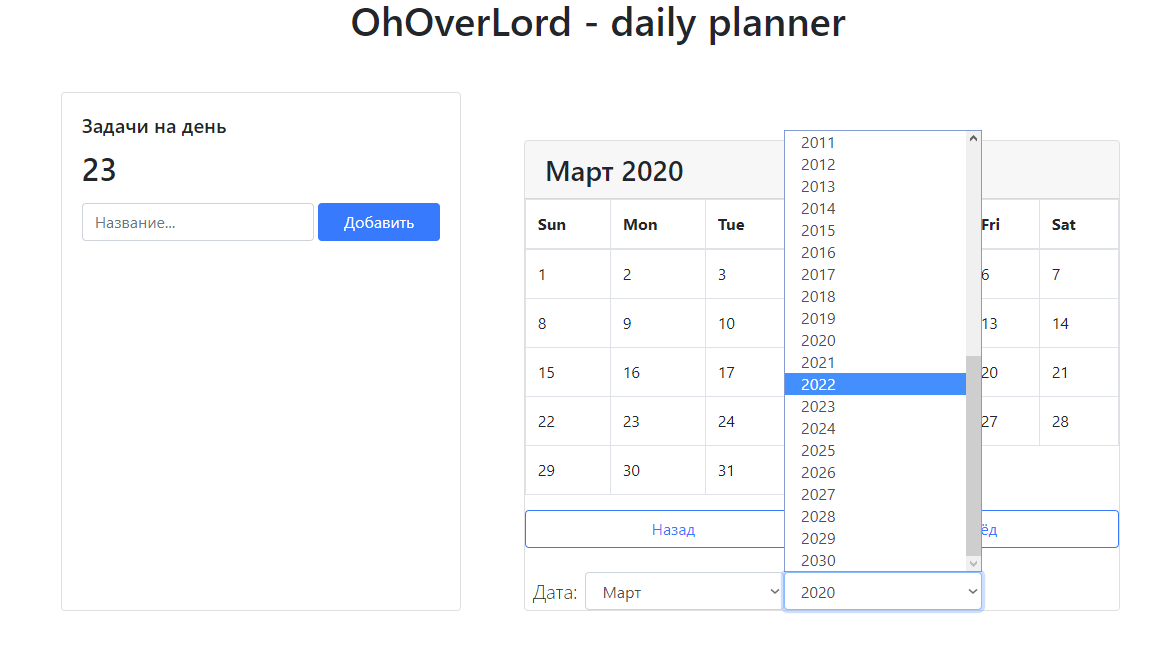


Рисунок 12 Выбор года

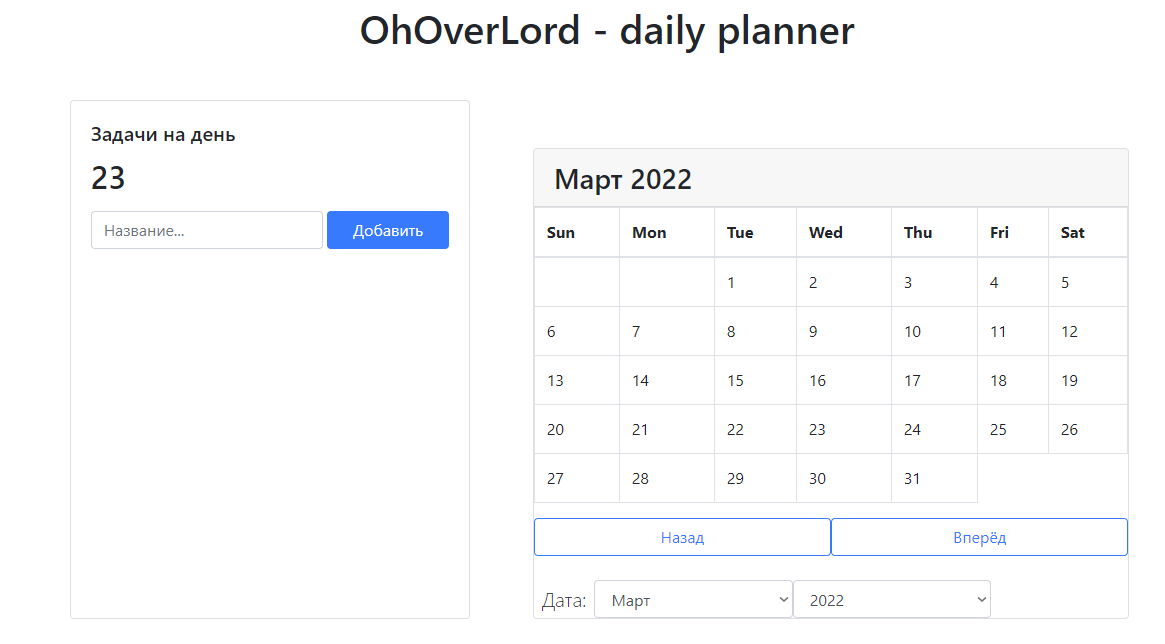


Рисунок 13 Переход на новый месяц

Тест 3. Добавление задачи на прошедший день.

Действия: выбрать дату, которая уже прошла и добавить туда задачу.

Ожидаемый результат: ячейка календаря должна стать жёлтой.

Результат теста:

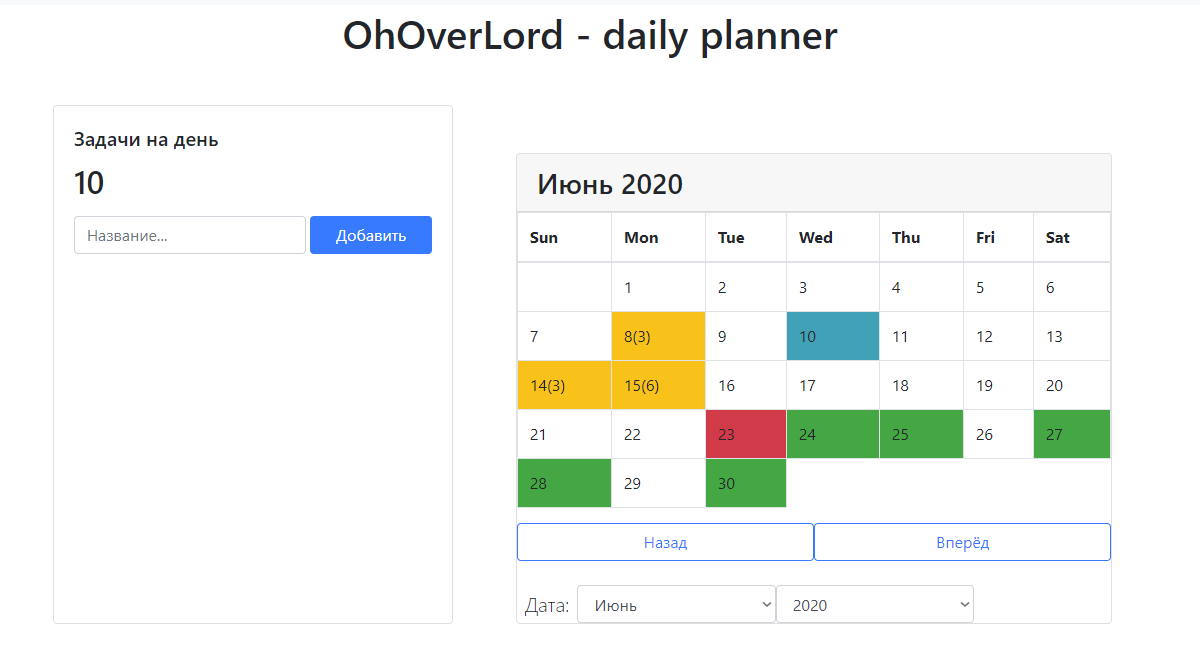


Рисунок 14 Выбор прошедшей даты

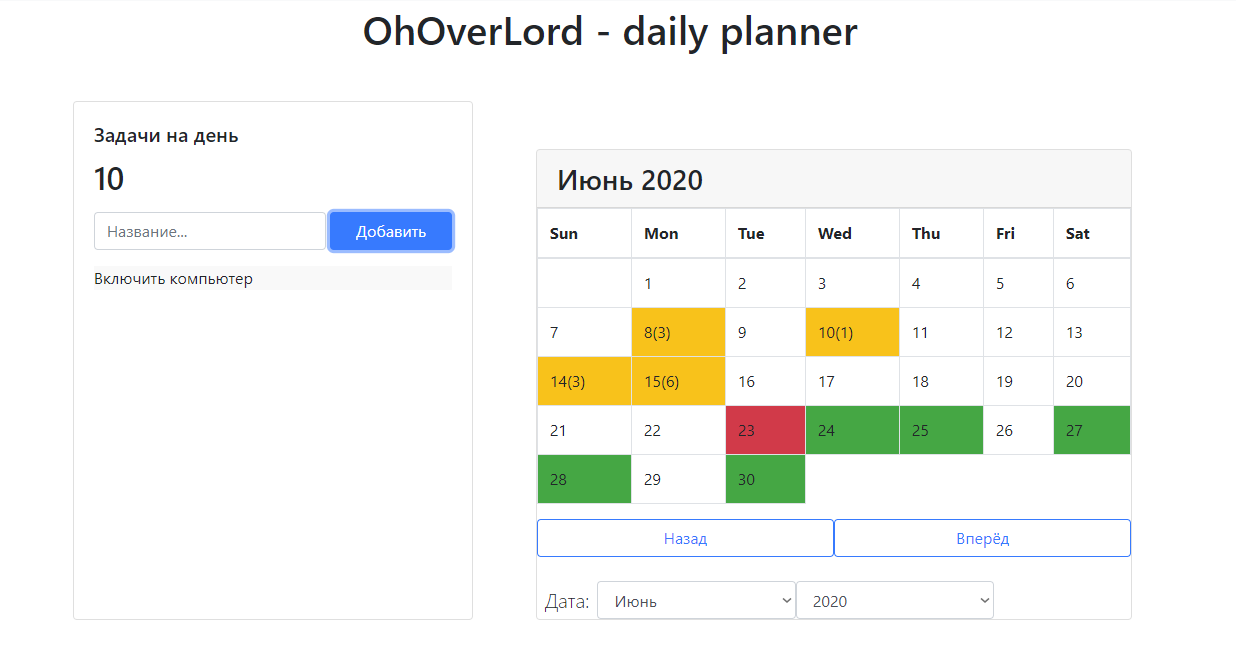


Рисунок 15 Добавление задачи

Тест 4. Регистрация.

Действия: зарегистрироваться через форму на странице регистрации.

Ожидаемый результат: переход на страницу авторизации.

Результат теста:

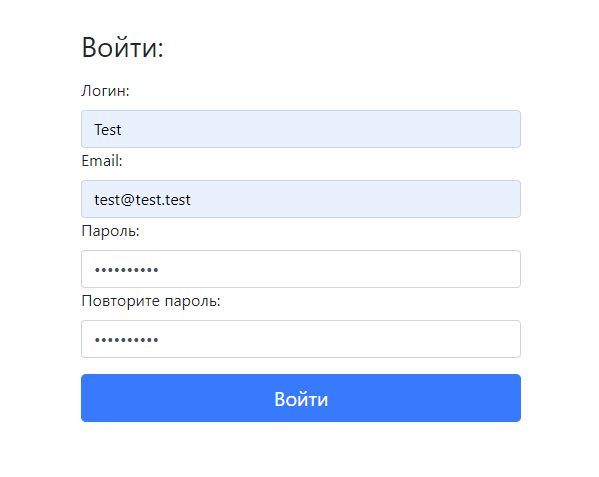


Рисунок 16 Регистрация пользователя

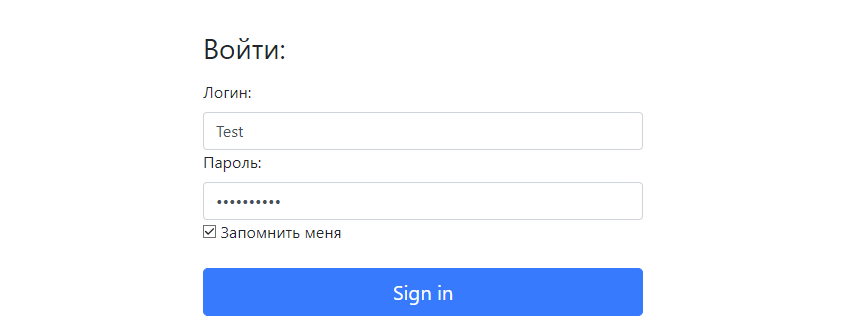


Рисунок 17 Форма входа

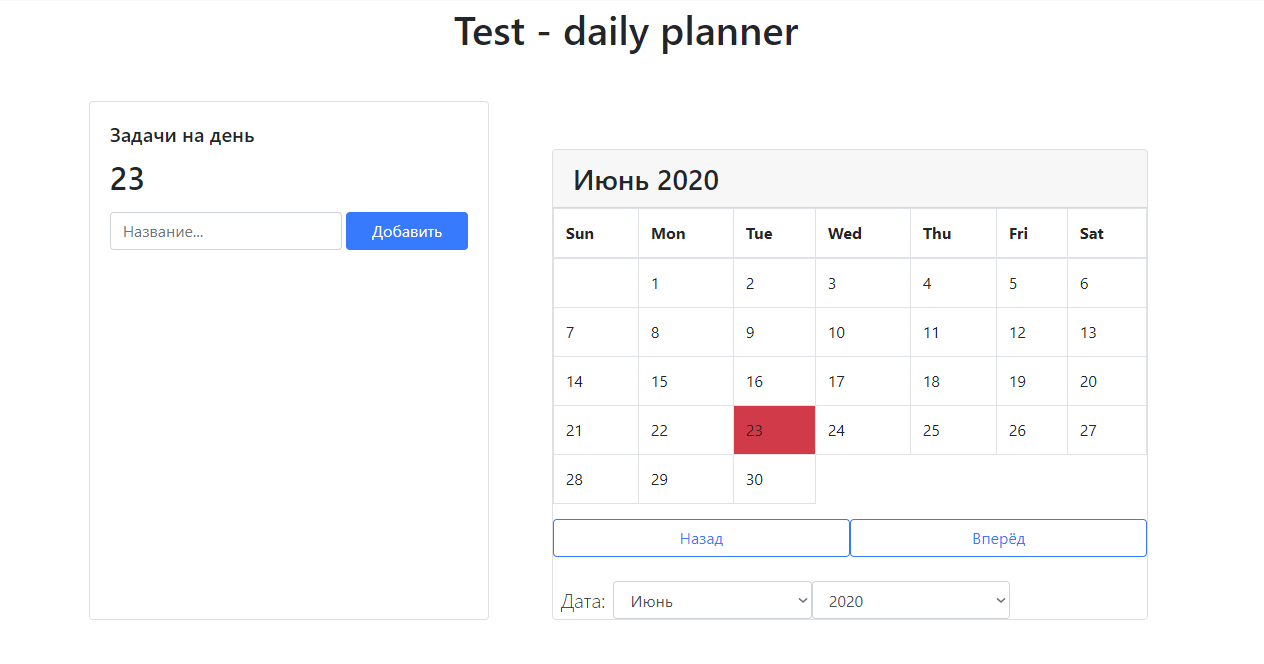


Рисунок 18 Страница с календарём для нового пользователя

## Описание применения средств отладки

В этом разделе показано умение применять средства отладки.

В ходе написания курсового проекта при попытке запустить скрипт было получено данное сообщение:



Рисунок 19 Ошибка в модуле calendar.js

После получения данного сообщения были просмотрены 54, 170, 86 строки модуля calendar.js и была обнаружена ошибка, которая впоследствии была устранена, а после попытки запуска скрипта не было получено ошибок, следовательно, ошибка была устранена и скрипт запустился.

## Анализ оптимальности использования памяти и быстродействия

В данном разделе будет проведен анализ оптимальности использования памяти и быстродействия программы.

Самая большая функция showCalendar() в модуле calendar.js имеет 2 вложенных цикла, следовательно, её сложность O(n^2).

Проведя тест по скорости выполнения, было вычислено время выполнения функции showCalendar(), оно составило 13.0850 миллисекунд.



Рисунок 20 Время выполнения функции showCalendar()

# Глава 3. Эксплуатационная часть

# Заключение

# Список литературы и интернет-источников

# Приложения