**

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

[Введение 3](#_Toc43926816)

[Глава 1. Теоретическая часть 4](#_Toc43926817)

[1.1. Описание предметной области 4](#_Toc43926818)

[1.1.1. Виды тайм-менеджмента 4](#_Toc43926819)

[1.1.2. Описание библиотеки Django 6](#_Toc43926820)

[1.1.3. Описание языка программирования python 7](#_Toc43926821)

[1.2. Описание существующих разработок 7](#_Toc43926822)

[Глава 2. Проектная часть 11](#_Toc43926823)

[2.1. Диаграмма прецедентов 11](#_Toc43926824)

[2.2. Выбор инструментов 11](#_Toc43926825)

[2.3. Проектирование сценария 13](#_Toc43926826)

[2.4. Диаграмма классов 13](#_Toc43926827)

[2.4.1. Диаграмма классов для модуля views.py 13](#_Toc43926828)

[2.4.2. Диаграмма классов для модуля models.py 15](#_Toc43926829)

[2.4.3. Диаграмма классов для модуля forms.py 15](#_Toc43926830)

[2.5. Описание главного модуля 16](#_Toc43926831)

[2.6. Описание спецификаций к модулям 16](#_Toc43926832)

[2.7. Описание модулей 18](#_Toc43926833)

[2.8. Описание тестовых наборов модулей 21](#_Toc43926834)

[2.9. Описание применения средств отладки 27](#_Toc43926835)

[2.10. Анализ оптимальности использования памяти и быстродействия 28](#_Toc43926836)

[Глава 3. Эксплуатационная часть 29](#_Toc43926837)

[3.1. Руководство оператора 29](#_Toc43926838)

[3.1.1 Состав функций 29](#_Toc43926839)

[3.1.2 Условия выполнения программы 29](#_Toc43926840)

[3.1.3 Выполнение программы 30](#_Toc43926841)

[Заключение 35](#_Toc43926842)

[Список литературы и интернет-источников 35](#_Toc43926843)

[Приложения 35](#_Toc43926844)

# 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»:**

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

1. Техника помидора

* Определитесь с задачей, которую будете выполнять.
* Поставьте помидор (таймер) на 25 минут.
* 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.
* Сделайте короткий перерыв (3-5 минут).
* После каждого 4-го «помидора» сделайте длинный перерыв (15-30 минут).
* 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 для веб-приложений на языке Python. Один из основных принципов фреймворка — DRY[[1]](#footnote-1). 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). 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 (с модулем mod\_python) и с использованием PostgreSQL в качестве базы данных. В настоящее время, помимо PostgreSQL, Django может работать с другими СУБД: MySQL[[2]](#footnote-2), SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle. Для работы с базой данных 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). 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).

[Документация Django](https://www.djangoproject.com)

[Документация Ruby on Rails](https://rubyonrails.org)

[Документация SQLite](https://www.sqlite.org/docs.html)

### 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”:

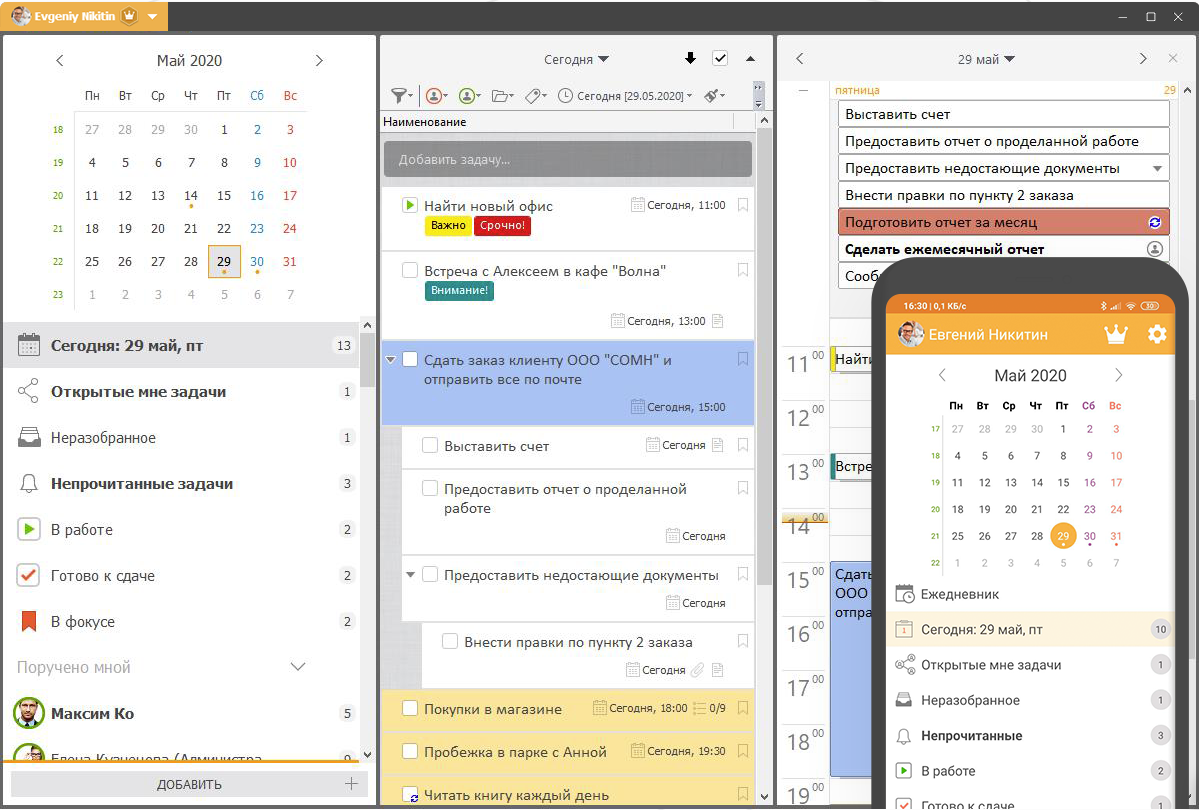


Рисунок 1 ЛидерТаск

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[[3]](#footnote-3), 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

[Сайт ЛидерТаск](https://www.leadertask.ru)

Второе рассмотренное решение – приложение “Мой ежедневник”:

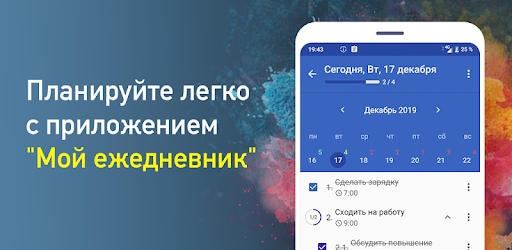


Рисунок 2 Мой ежедневник

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

[**Мой ежедневник в google play**](https://play.google.com/store/apps/details?id=com.time_management_studio.my_daily_planner)

# Глава 2. Проектная часть

## Диаграмма прецедентов

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

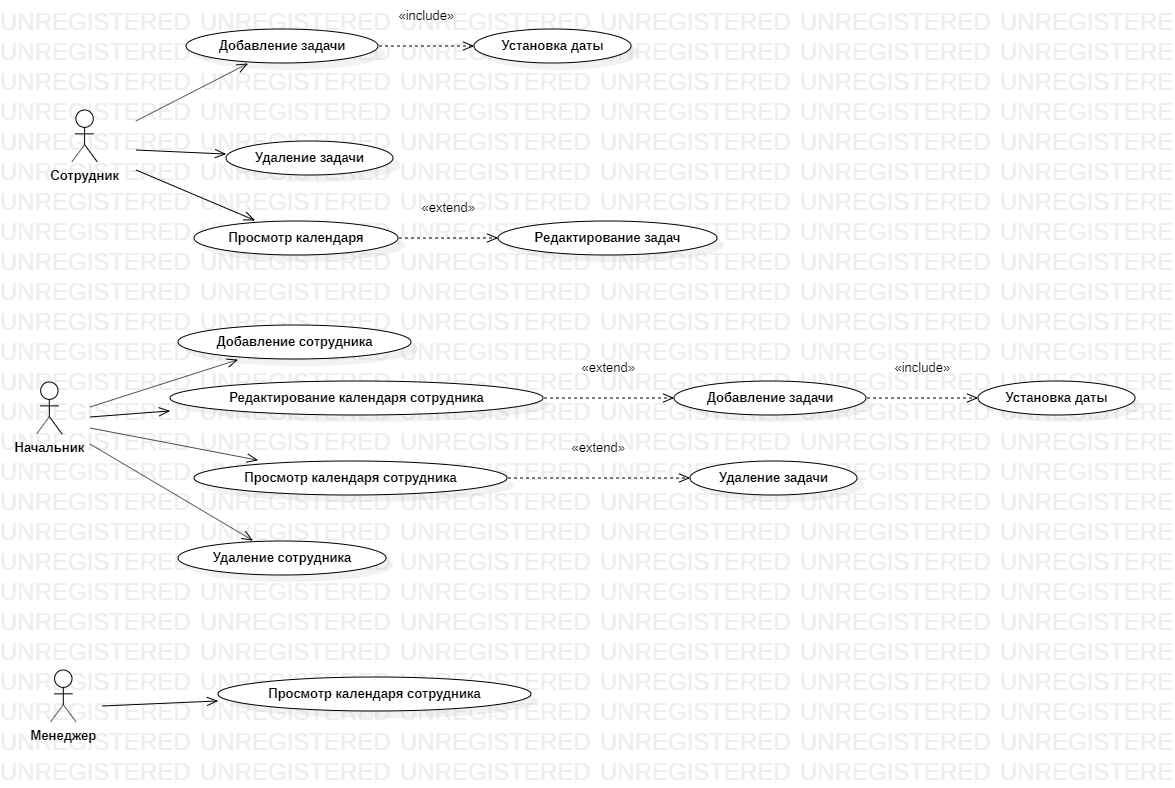


Рисунок 3 Диаграмма прецедентов

## 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

Таблица 1. Оценка языков программирования

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

Таблица 2. Оценка сред разработки

|  |  |  |
| --- | --- | --- |
|  | 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

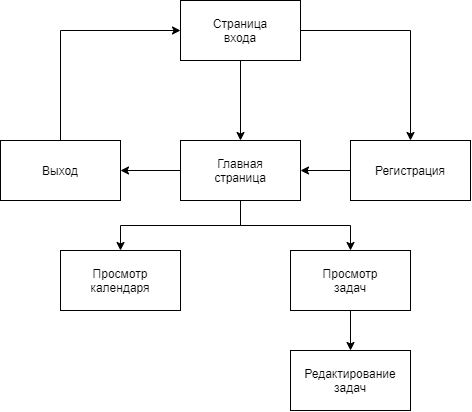


Рисунок 4 Диаграмма сценария

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

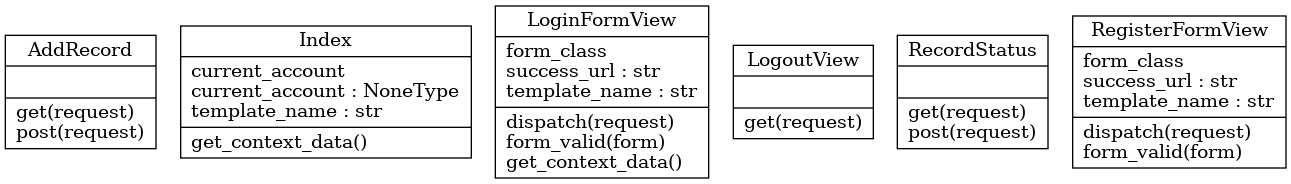


Рисунок 6 Диаграмма классов для модуля views.py

### Диаграмма классов для модуля models.py

**Класс “Record”**

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

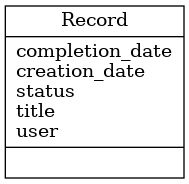


Рисунок 7 Диаграмма классов для модуля models.py

### 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

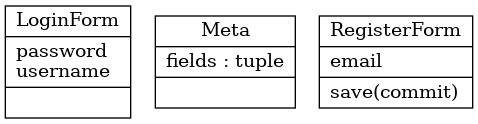


Рисунок 8 Диаграмма классов для модуля forms.py

## 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. Код файла manage.py

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

**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 Код к функции dateRecords(date)

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 функции dateRecords(date)**

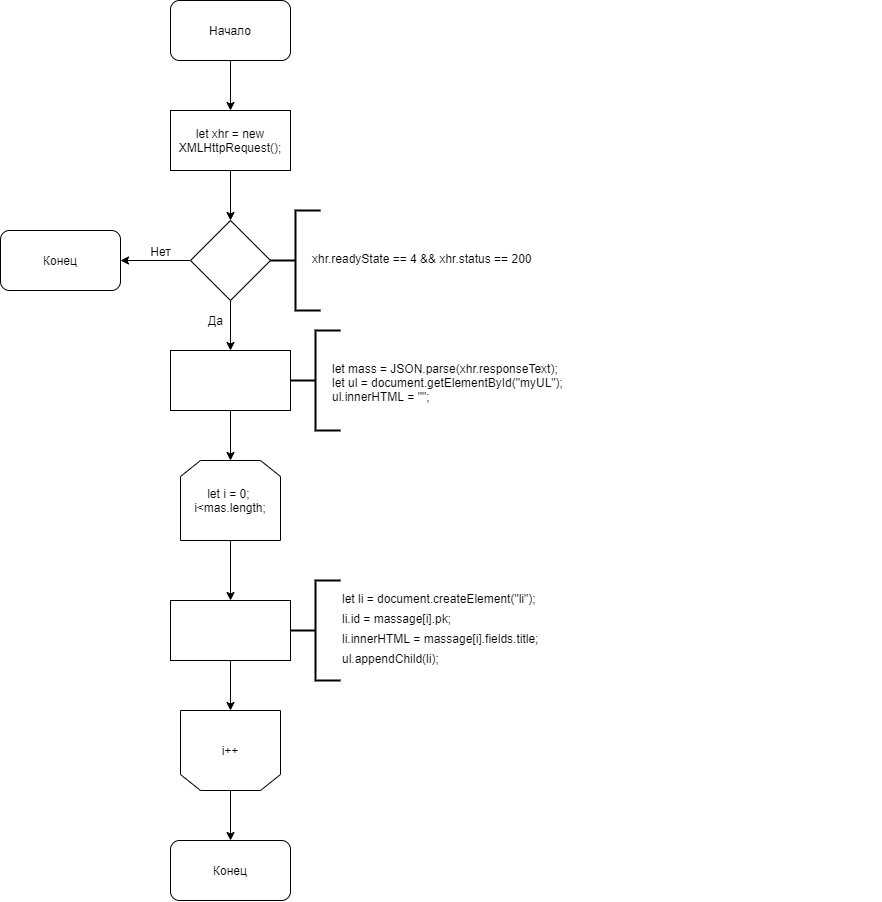


Рисунок 9 Блоксхема к методу dateRecords()

## 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:

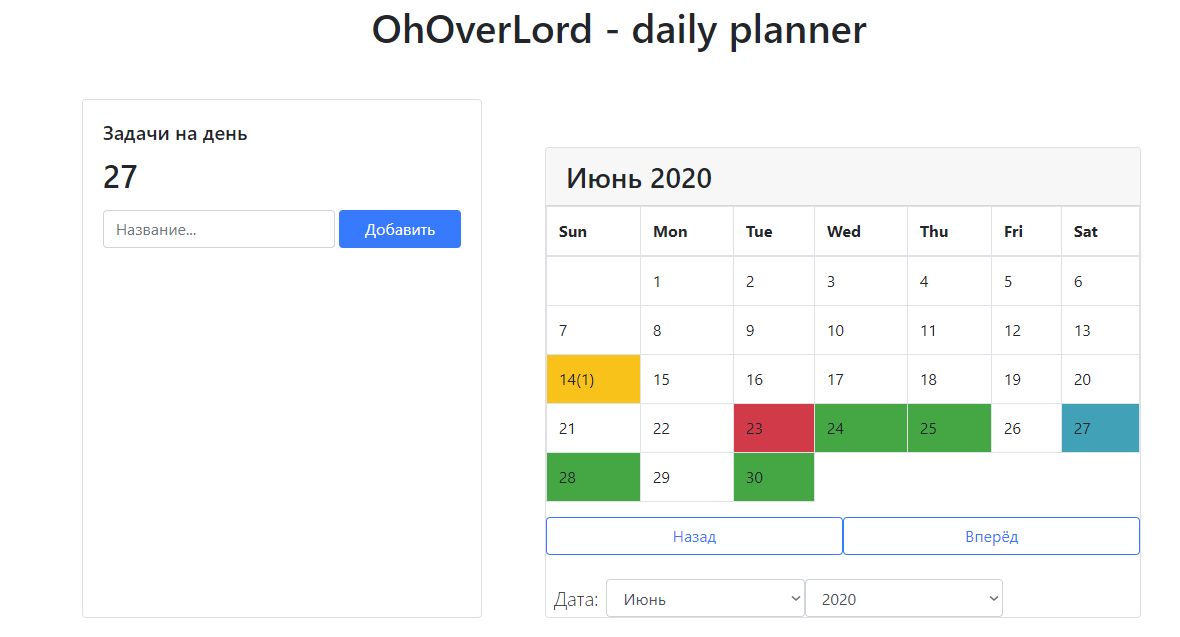


Рисунок 10 Выбор дня в календаре

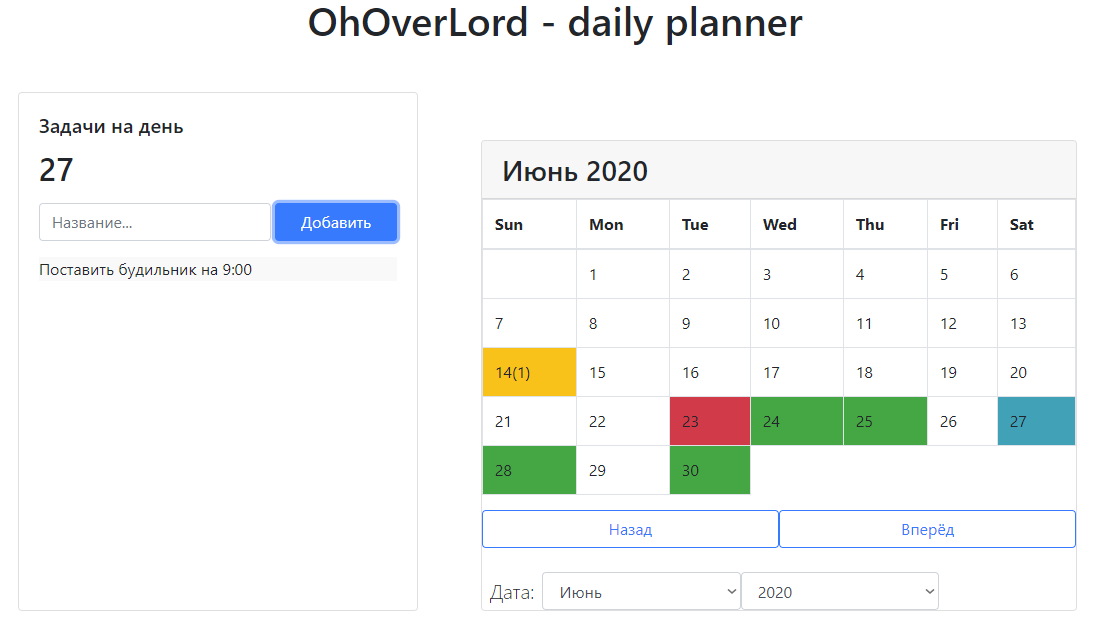


Рисунок 11 Новая задача в списке задач на этот день

**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:

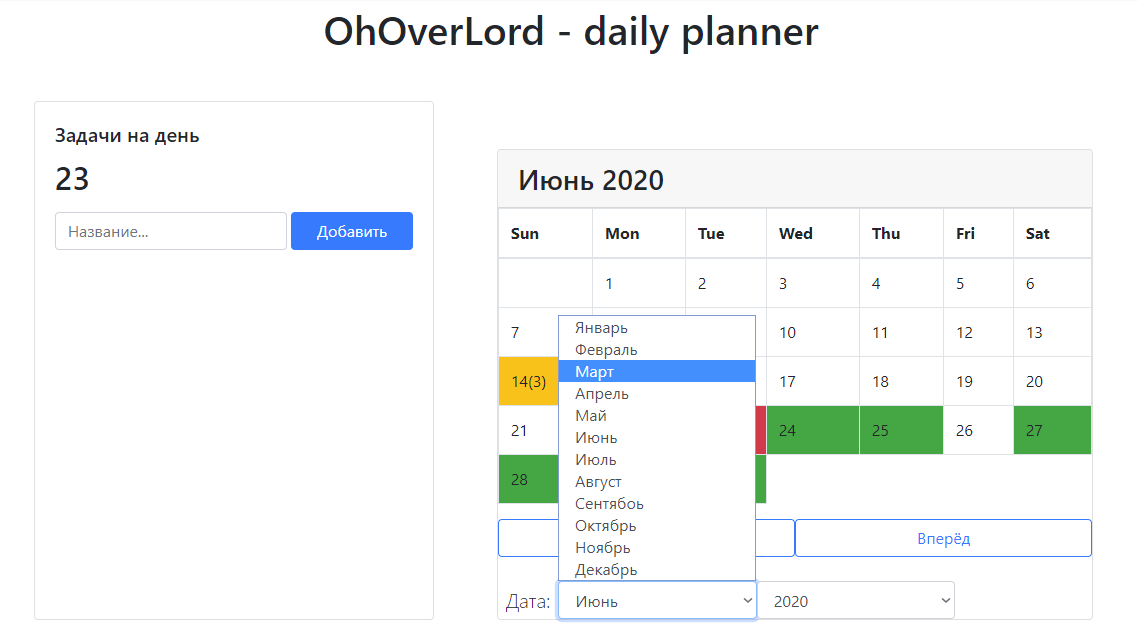


Рисунок 12 Выбор месяца

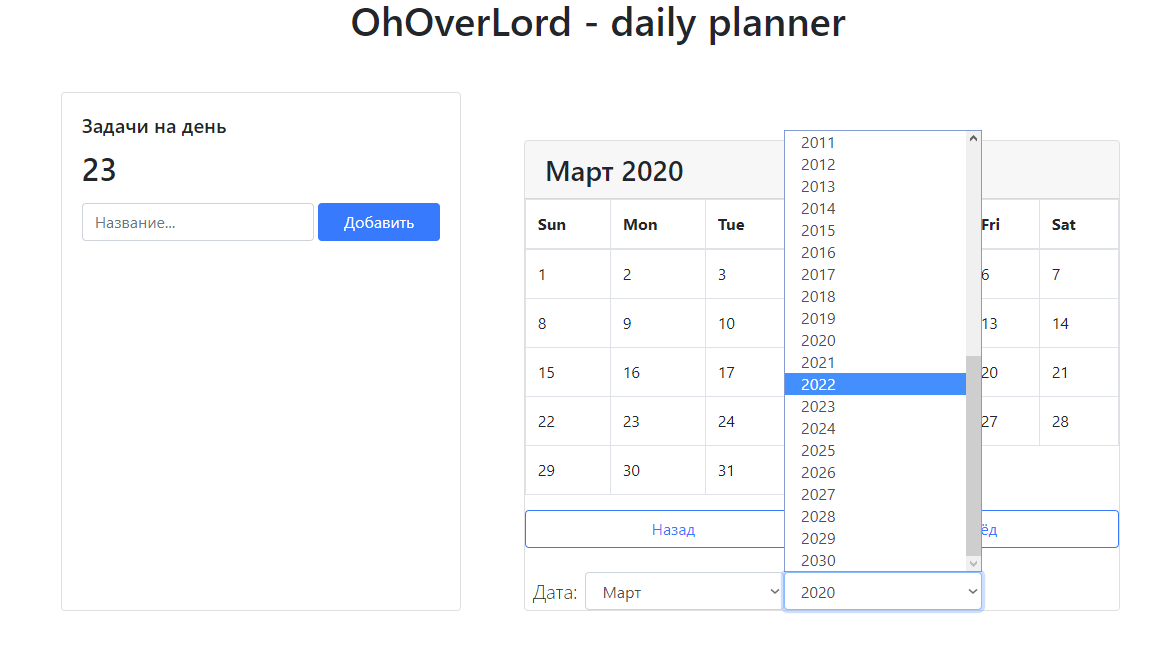


Рисунок 13 Выбор года

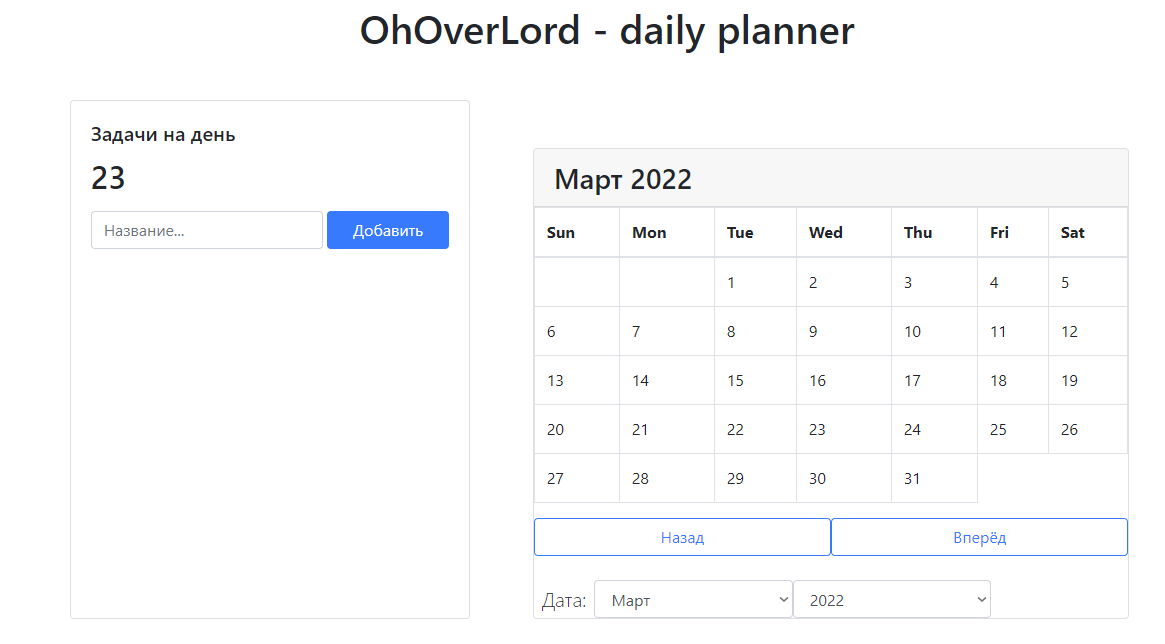


Рисунок 14 Переход на новый месяц

**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:

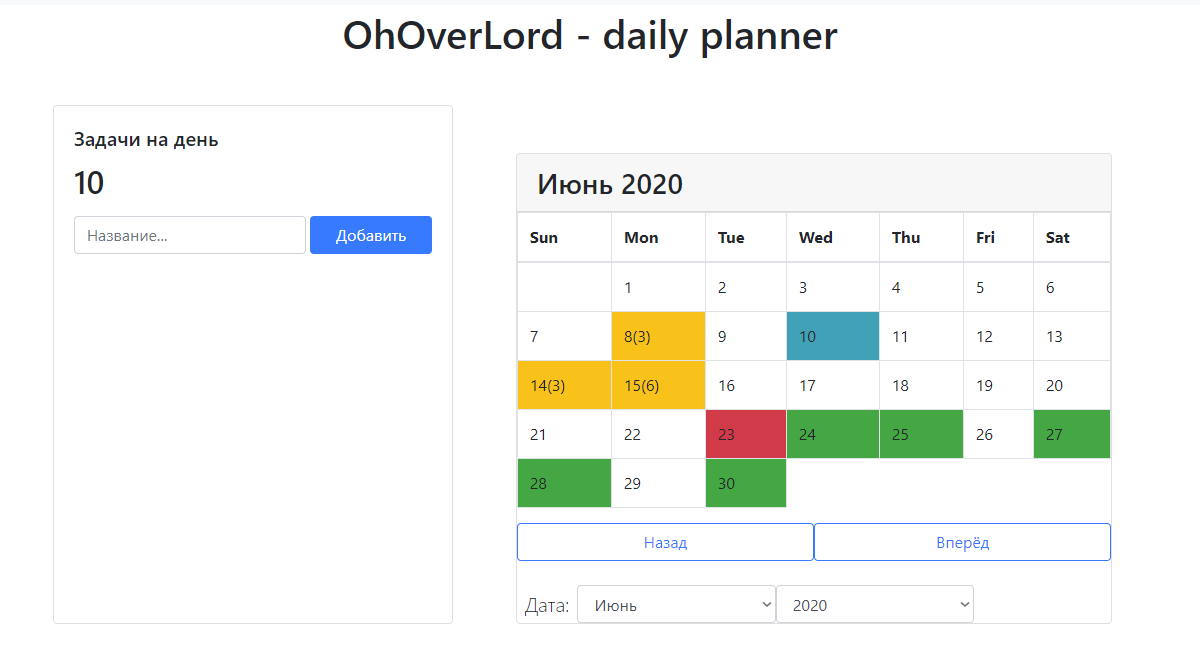


Рисунок 15 Выбор прошедшей даты

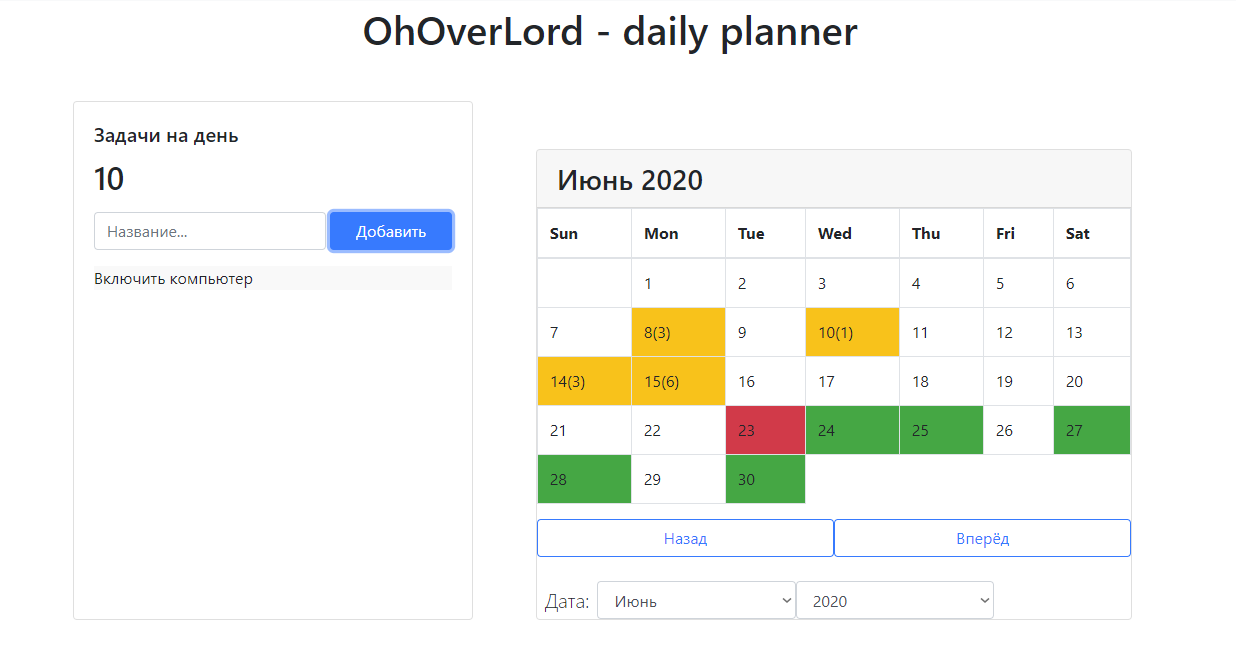


Рисунок 16 Добавление задачи

**Тест 4. Регистрация.**

Действия:

зарегистрироваться через форму на странице регистрации.

Ожидаемый результат:

переход на страницу авторизации.

Результат теста:

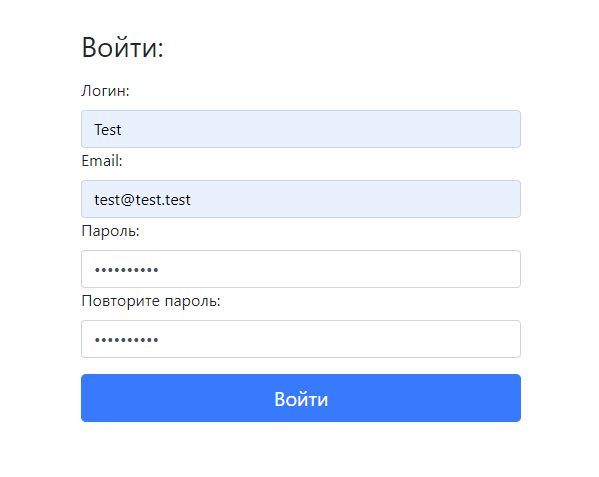


Рисунок 17 Регистрация пользователя

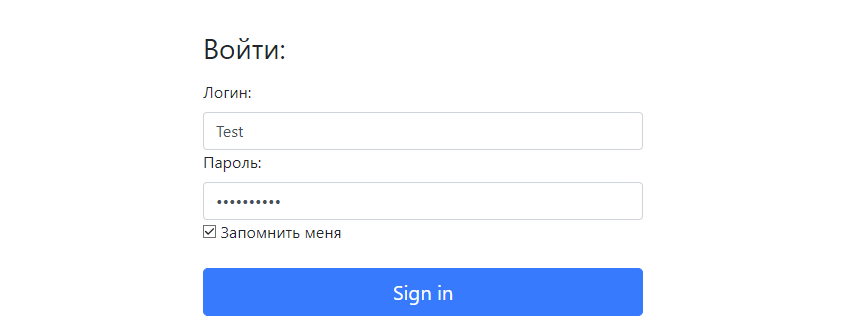


Рисунок 18 Форма входа

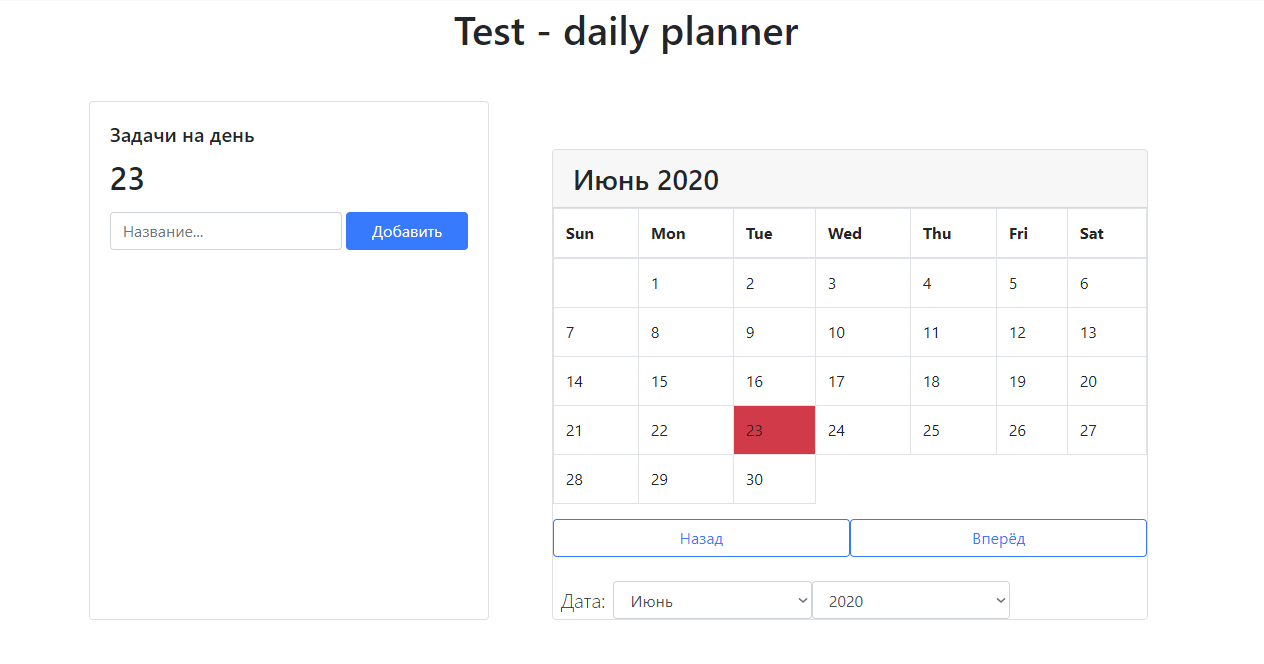


Рисунок 19 Страница с календарём для нового пользователя

## Описание применения средств отладки

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



Рисунок 20 Ошибка в модуле calendar.js

Суть ошибки заключается в опечатке, вместо XMLHttpRequest было написано XMLHttpReques. 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 21 Время выполнения функции showCalendar()

# Глава 3. Эксплуатационная часть

## Руководство оператора

#### Функциональное назначение программы

Основной функцией проекта “Daily planner” является управление тайм-менеджментом. Приложение помогает следить за поставленными задачами и грамотно распоряжаться своим временем.

#### Эксплуатационное назначение программы

Приложение может быть использовано на любом телефоне или компьютере.

## Состав функций

#### Функция авторизации

Эта функция позволяет пользователю авторизироваться.

#### Функция регистрации

Эта функция позволяет пользователю зарегистрироваться.

#### Функция перехода на определённый день, месяц или год

Эта функция позволяет выбрать в календаре любой день месяц или год.

#### Функция добавление задачи

Эта функция позволяет добавить задачу на определённый день.

#### Функция завершения задачи

Эта функция позволяет завершить задачу

## Условия выполнения программы

#### Минимальный состав аппаратных средств

* Устройство имеющее выход в интернет

#### Минимальный состав программных средств

Для функционирования проекта “Daily planner” необходим браузер поддерживающий HTML5 и JavaScript версии ES9.

#### Требование к персоналу (Пользователю)

Конечный пользователь должен обладать элементарными навыками работы с браузером.

## Выполнение программы

#### Загрузка и запуск программы

Загрузка и запуск проекта “Daily planner” осуществляется в следующем порядке.

Необходимо скачать проект с githab’а и распаковать его в папку.

[Репозиторий "Daily planner" на github](https://github.com/OhOverLord/daily-planner)

Для запуска программы необходимо:

* [Скачать Django](https://www.djangoproject.com/download/)
* Открыть терминал и перейти в каталог dailyPlannerSite
* Ввести команду python manage.py runserver.

При запуске надо перейти по ссылке, которая появится в терминале. Далее будет отображаться страница входа.

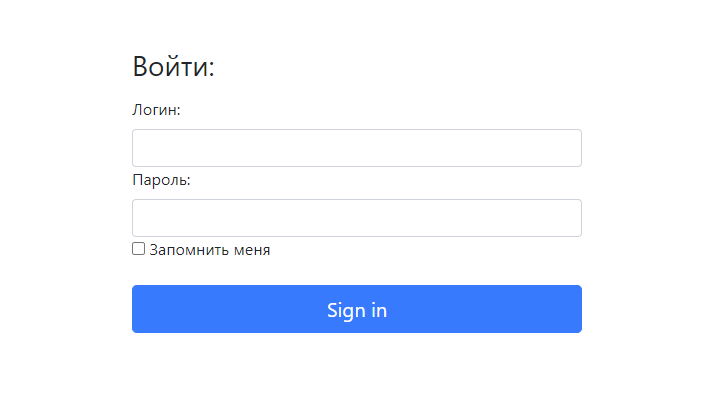


Рисунок 22 Окно авторизации

После можно либо авторизоваться либо перейти в окно регистрации

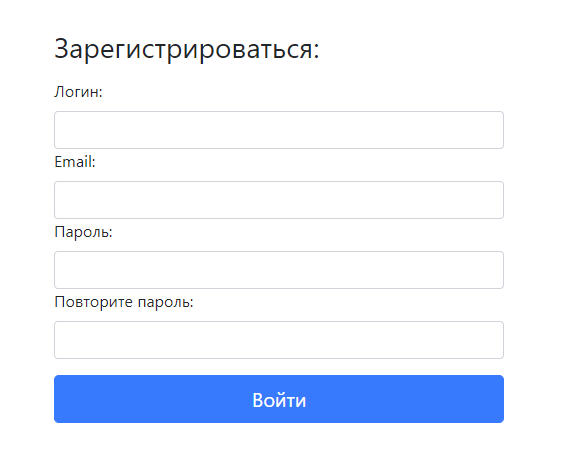


Рисунок 23 Окно регистрации

После регистрации откроется окно авторизации.

После авторизации откроется окно с календарём и с добавлением и отображением задач.

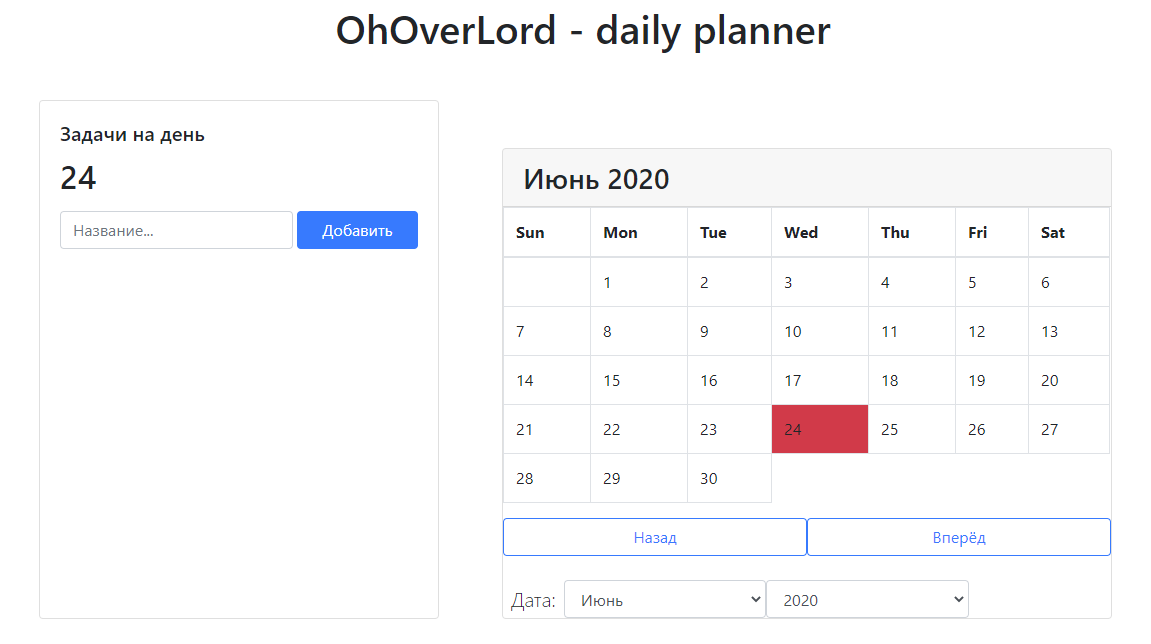


Рисунок 24 Окно с календарём и с добавлением и отображением задач

#### Выполнение программы

Далее будут рассмотрены действия, которые доступны авторизованному пользователю

1. Выбор даты

Пользователь может выбрать любую дату на календаре.

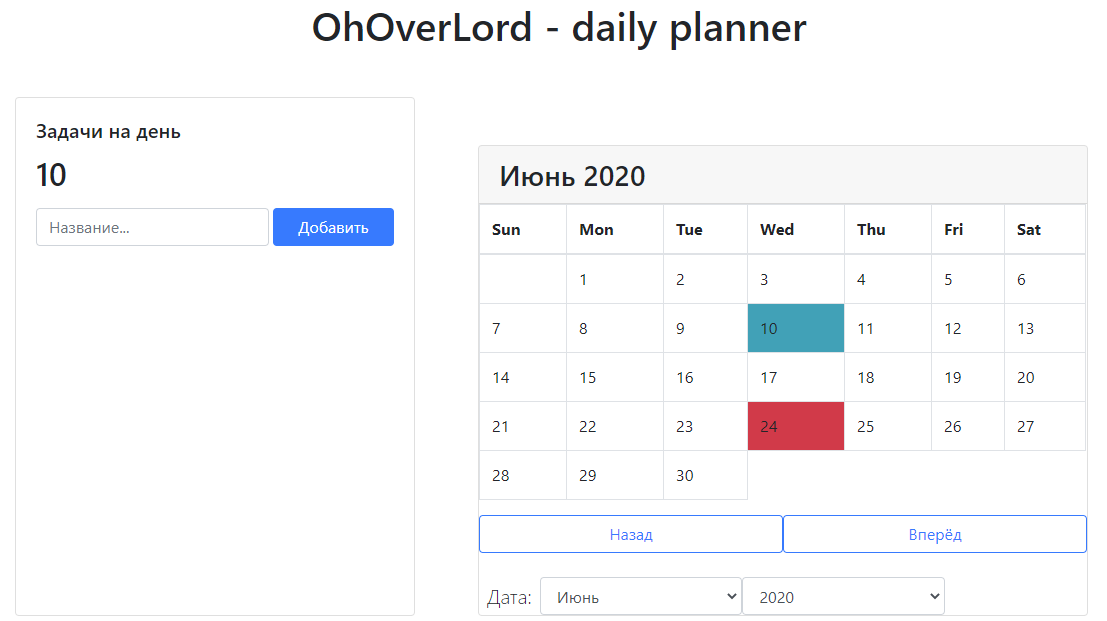


Рисунок 25 Выбор даты

1. Выбор месяца

Пользователь может выбрать из выпадающего списка любой месяц и на календаре будет отображаться месяц, который выбрал пользователь.

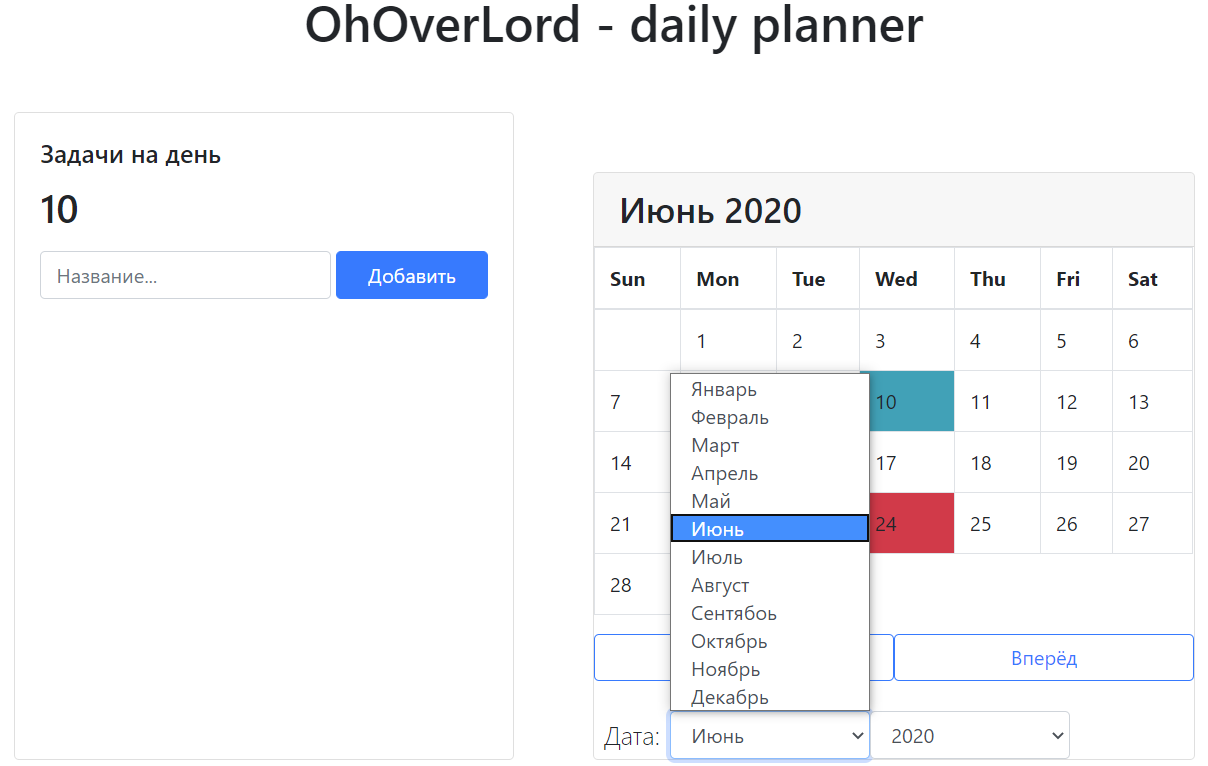


Рисунок 26 Выбор месяца

1. Выбор года

Пользователь может выбрать из выпадающего списка любой год и на календаре будет отображаться год и месяц, который выбрал пользователь.

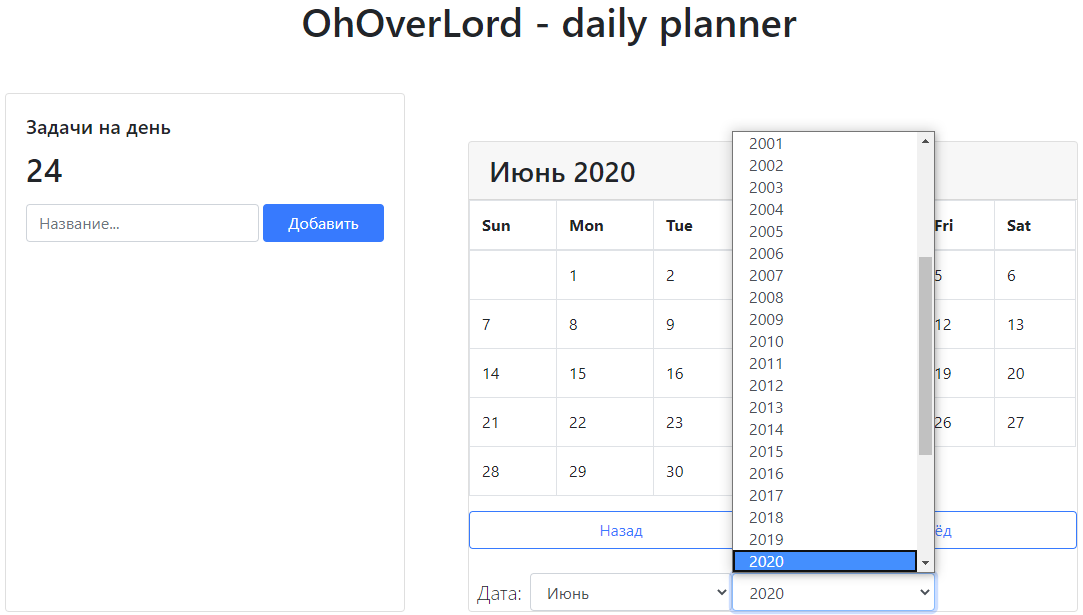


Рисунок 27 Выбор даты

1. Добавление задачи

Вписав задачу в поле и нажав на кнопку добавить, пользователь добавит задачу на определённый день, который он выбрал.

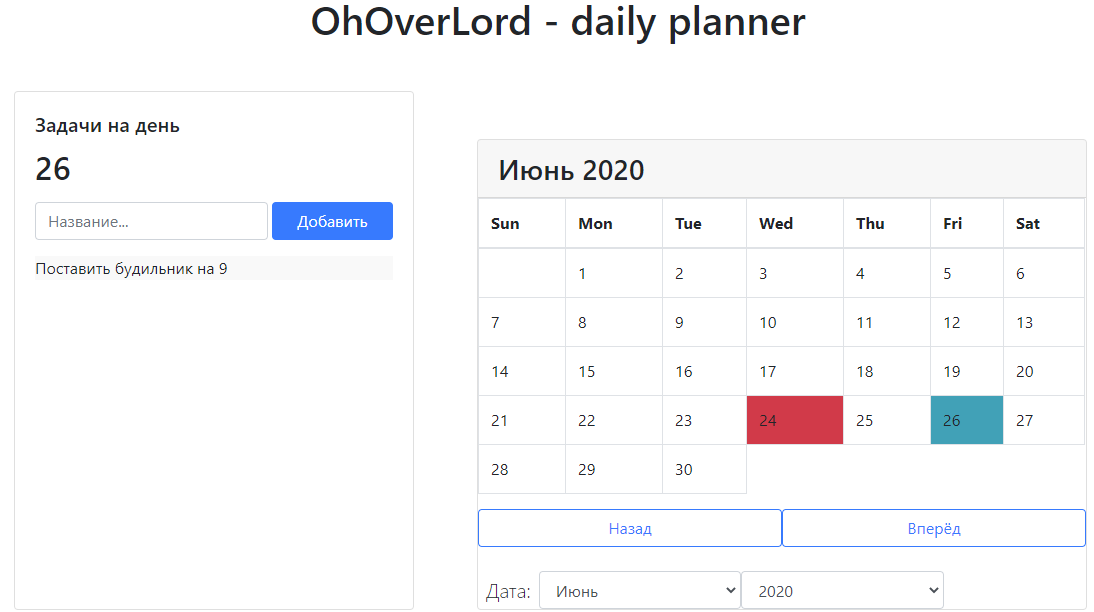


Рисунок 28 Добавление задачи

1. Завершение задачи

Пользователь может кликнуть по задаче, тем самым завершив её.

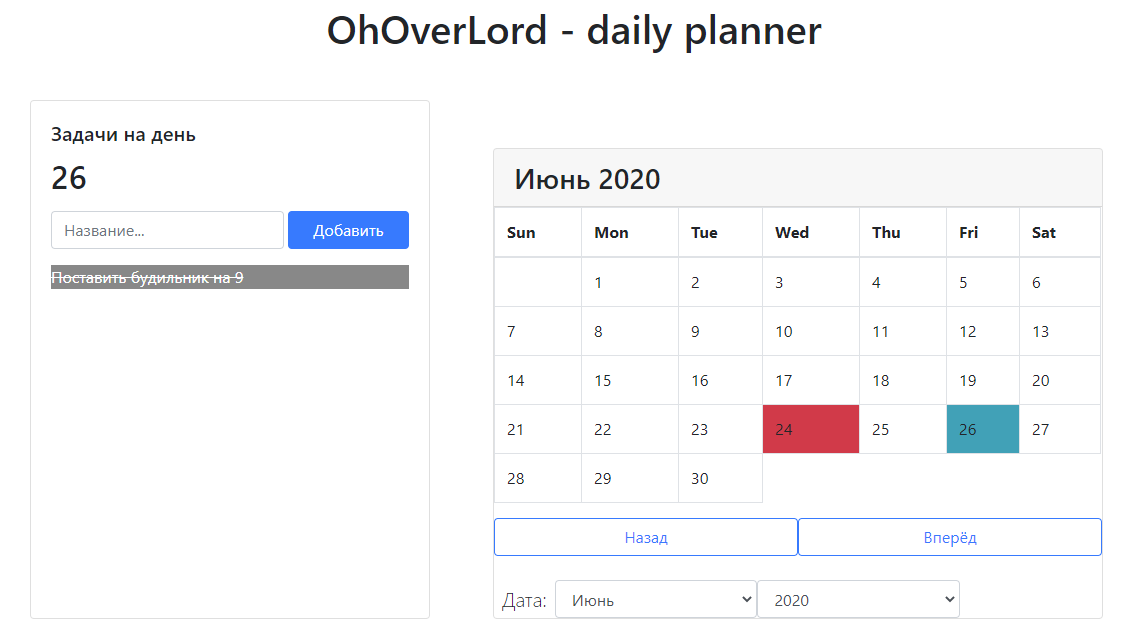


Рисунок 29 Завершение задачи

1. Просмотр календаря

* Клетка будет жёлтой, если на эту дату есть незавершённые задачи, на дату, которая уже прошла, и в скобочках будет отображаться количество невыполненных задач.
* Клетка будет зелёной, если добавить задачу на день который ещё не наступил.
* Клетка сегодняшнего дня всегда будет красной

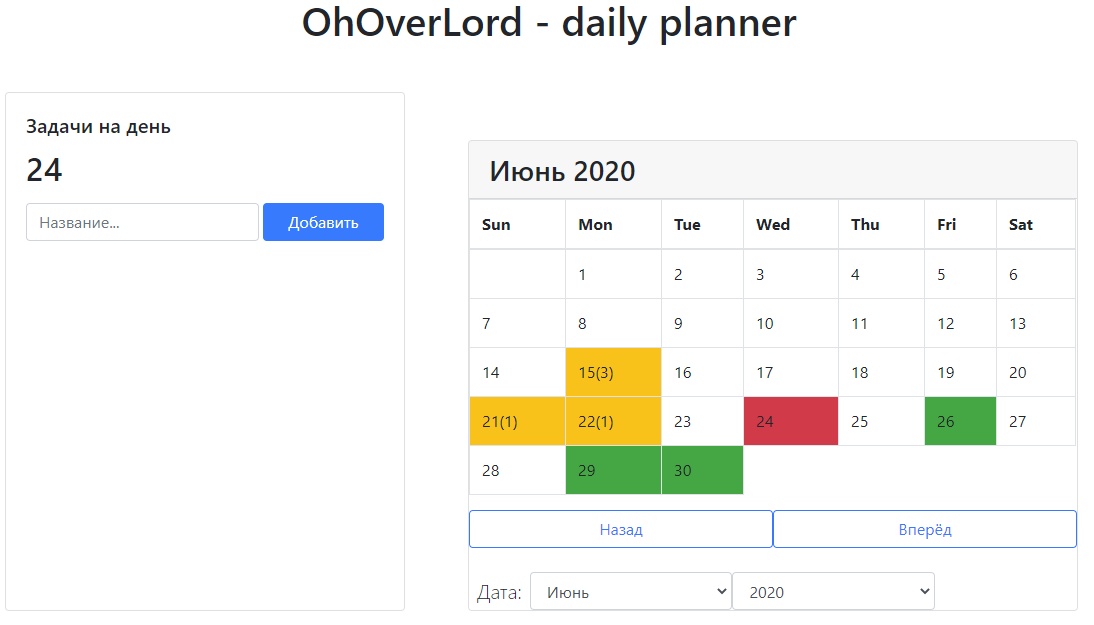


Рисунок 30 Просмотр календаря

# Заключение

В результате выполнения курсового проекта была написана программа “Daily planner” для управления тайм-менеджментом.

В ходе работы были проанализированы предметная область, существующие разработки, посвященные данному направлению, получены практические навыки по созданию сайтов в PyCharm на основе библиотеки Django.

Так же планируется продолжить работу над данным проектом с целью расширения возможностей и удобства для пользователей. Планы по доработке представлены ниже.

* Реализация разных методов тайм-менеджмента.
* Доработка интерфейса
* Реализация бизнес-ежедневника
* Реализация профиля пользователя
* Реализация добавления в друзья других пользователей

# Список литературы и интернет-источников

1. Официальный сайт Django

<https://www.djangoproject.com/>

1. Метод “Слоны и лягушки”

<http://tvoe-reshenie.com/taym-menedzhment-kak-vse-uspevat-tehnika-slonyi-i-lyagushki/>

1. Техника помидора

<https://ru.wikipedia.org/wiki/Метод_помидора>

1. Карта или фотография рабочего времени

<https://lico.ru/o_kompanii/stati/razvitie_personala/karta_rabochego_vremeni_i_fotografiya_rabochego_vremeni/>

1. Метод А. А. Любищева

<https://habr.com/ru/post/132894/>

1. ЛидерТаск

<https://www.leadertask.ru>

1. Мой ежедневник

<https://play.google.com/store/apps/details?id=com.time_management_studio.my_daily_planner>

1. SQLite

<https://www.sqlite.org/docs.html>

# Приложения

#### Приложение 1. Код главного модуля manage.py

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()

#### Приложение 2. Код модуля views.py

import datetime

import json

from django.contrib import messages

from django.contrib.auth import login, authenticate, logout

from django.contrib.auth.decorators import login\_required

from django.contrib.auth.forms import AuthenticationForm

from django.contrib.auth.models import User

from django.core import serializers

from django.http import JsonResponse, HttpResponse

from django.shortcuts import render, redirect

from django.utils.decorators import method\_decorator

from django.views.generic import FormView

from django.views.generic.base import View, TemplateView

from .forms import \*

@method\_decorator(login\_required, name='dispatch')

class Index(TemplateView):

"""

Класс для отображения гдавной страницы

"""

template\_name = 'index.html'

current\_account = None

def get\_context\_data(self, \*\*kwargs):

"""

Функция для обновления контекста

:param kwargs:

:return: Обновлённый контекст

"""

context = super().get\_context\_data(\*\*kwargs)

self.current\_account = User.objects.get(username=self.request.user)

context = {

'title': ' - daily planner',

'user': self.current\_account,

'record\_form': RecordForm()

}

return context

class LoginFormView(FormView):

"""

Класс для отображения страницы авторизации

"""

template\_name = 'login.html'

form\_class = AuthenticationForm

success\_url = '/'

def dispatch(self, request, \*args, \*\*kwargs):

"""

Функция для редиректа на главную страницу, если пользователь уже авторизован

:param request:

:param args:

:param kwargs:

:return: Редирект на главную страницу

"""

if request.user.is\_authenticated:

return redirect('/')

return super().dispatch(request, \*args, \*\*kwargs)

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context = {

'loginform': LoginForm()

}

return context

def form\_valid(self, form):

"""

Функция для обработки данных формы, если они правильные

:param form:

:return: Форма

"""

user = form.get\_user()

login(self.request, user)

if self.request.method == 'POST' and self.request.POST.get('remember\_me', None):

one\_month\_to\_sec = 60 \* 60 \* 24 \* 30

self.request.session.set\_expiry(one\_month\_to\_sec)

return super().form\_valid(form)

class LogoutView(View):

"""

Класс для деавторизации пользовалетя

"""

def get(self, request):

"""

Функция для обработки get запроса

:param request:

:return: Редирект на главную страницу

"""

logout(request)

one\_hour\_to\_sec = 60 \* 60

self.request.session.set\_expiry(one\_hour\_to\_sec)

return redirect('/')

class RegisterFormView(FormView):

"""

Класс для отображения страницы регистрации

"""

template\_name = 'registration.html'

form\_class = RegisterForm

success\_url = '/daily\_planner/login/'

def dispatch(self, request, \*args, \*\*kwargs):

"""

Функция для редиректа на главную страницу, если пользователь уже авторизован

:param request:

:param args:

:param kwargs:

:return: Редирект на главную страницу

"""

if request.user.is\_authenticated:

return redirect('/')

return super().dispatch(request, \*args, \*\*kwargs)

def form\_valid(self, form):

"""

Функция для сохранения данных пользователя, если они правильные

:param form:

:return:

"""

if form.is\_valid:

form.save()

return super().form\_valid(form)

return redirect('register')

class AddRecord(View):

"""

Класс для добалнения запроса в БД

"""

def get(self, request):

"""

Функция для обработки get запроса

:param request:

:return: Все записи которые назначены на определённый день в виде json

"""

data = request.GET.get("data", "")

current\_user = User.objects.get(username=request.user)

return HttpResponse(serializers.serialize('json', current\_user.record\_set.filter(completion\_date=data))

, content\_type='application/json')

def post(self, request):

"""

Функция для обработки post запроса, который добавляет новую задачу в БД

:param request:

:return: Редирект на главную страницу

"""

current\_user = request.user

record = Record()

record.user = current\_user

record.title = request.POST.get("title")

record.completion\_date = request.POST.get("completion\_date")

record.save()

return redirect("/")

class RecordStatus(View):

"""

Класс для управления статусами задач

"""

def get(self, request):

"""

Функция для обработки get запроса

:param request:

:return: Количество заадач, количество невыполненых задач, количество выполненых задач

"""

current\_user = User.objects.get(username=request.user)

now = datetime.datetime.now()

data = request.GET.get("data", "")

ln = current\_user.record\_set.filter(completion\_date=data)

ln\_success = current\_user.record\_set.filter(completion\_date=data, status="Выполнено")

data = data.split('-')

sch = 0

if int(data[0]) <= now.year:

if int(data[1]) <= now.month or int(data[2]) <= now.day + 1:

for el in ln:

if el.status != 'Выполнено':

el.status = 'Не выполнено'

el.save()

sch += 1

return HttpResponse(json.dumps({"length": len(ln), "count": sch, "ln\_success": len(ln\_success)}), content\_type="application/json")

def post(self, request):

"""

Функция для обработки post запроса, который ставит задаче статус выполнено

:param request:

:return:

"""

record = Record.objects.get(id=request.POST.get("id"))

record.status = "Выполнено"

record.save()

return HttpResponse("True")

#### Приложение 3. Код модуля calendar.js

// Функция для получения csrftoken

var csrfcookie = function() {

var cookieValue = null,

name = 'csrftoken';

if (document.cookie && document.cookie !== '') {

var cookies = document.cookie.split(';');

for (var i = 0; i < cookies.length; i++) {

var cookie = cookies[i].trim();

if (cookie.substring(0, name.length + 1) == (name + '=')) {

cookieValue = decodeURIComponent(cookie.substring(name.length + 1));

break;

}

}

}

return cookieValue;

};

// Функция для добавления нового элемента в список задач

function newElement() {

var li = document.createElement("li");

li.innerHTML = document.getElementById("myInput").value;

if (document.getElementById("myInput").value === '') {

alert("You must write something!");

} else {

let currentDay = document.getElementById("titleDay").innerHTML;

var request = new XMLHttpRequest();

request.open('POST', 'http://127.0.0.1:7000/daily\_planner/add\_record/', true);

request.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded; charset=UTF-8');

request.setRequestHeader('X-CSRFToken', csrfcookie());

var body = 'title=' + encodeURIComponent(document.getElementById("myInput").value) +

'&completion\_date=' + encodeURIComponent(day(currentDay));

request.send(body);

dateRecords(currentDay);

document.getElementById("myInput").value = "";

cell = document.getElementById(day(currentDay));

if(new Date().getMonth() < Number(selectMonth.value) || (new Date().getMonth() == Number(selectMonth.value) && new Date().getDate() < Number(currentDay)))

cell.classList.add("bg-success");

else

{

if(cell.lastChild.innerHTML == '')

cell.lastChild.innerHTML = '(1)';

else

cell.lastChild.innerHTML = '(' + String(Number(cell.lastChild.innerHTML[1]) + 1) + ')';

cell.classList.add("bg-warning");

}

}

}

// Функция для приведения даты к виду ГГГГ-ММ-ДД

function day(date) {

if(String(currentMonth + 1).length == 1)

return currentYear + '-0' + (currentMonth + 1) + '-' + date;

return currentYear + '-' + (currentMonth + 1) + '-' + date;

}

// Функция для получения всех задач на определённый день

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

if(massage[i].fields.status == 'Выполнено')

li.classList.add('checked');

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

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

}

}

})

}

//Календарь

today = new Date(); // Переменная текущего дня

currentMonth = today.getMonth(); // Переменная текущего метода

currentYear = today.getFullYear(); // Переменная текущего года

selectYear = document.getElementById("year"); // Переменная выбранного года

selectMonth = document.getElementById("month"); // Переменная выбранного месяца

thisDay = document.getElementById("day-plan"); // Переменная - контейнер для задач на сегодняшний день

titleDay = document.getElementById("titleDay"); // Переменная - контейнер для названия текущего дня

months = ["Январь", "Февраль", "Март", "Апрель", "Май", "Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"]; // массив с месяцами

monthAndYear = document.getElementById("monthAndYear"); // Переменная - контейнер для названия месяца и года

showCalendar(currentMonth, currentYear); // ызов функции для отрисовки календаря

// Функция для перехода на следующий месяц

function next() {

currentYear = (currentMonth === 11) ? currentYear + 1 : currentYear;

currentMonth = (currentMonth + 1) % 12;

showCalendar(currentMonth, currentYear);

}

// Функция для перехода на предыдущий месяц

function previous() {

currentYear = (currentMonth === 0) ? currentYear - 1 : currentYear;

currentMonth = (currentMonth === 0) ? 11 : currentMonth - 1;

showCalendar(currentMonth, currentYear);

}

// Функция для перехода на определённый месяц и год

function jump() {

currentYear = parseInt(selectYear.value);

currentMonth = parseInt(selectMonth.value);

showCalendar(currentMonth, currentYear);

}

// Функция для отрисовки календаря

function showCalendar(month, year) {

let firstDay = (new Date(year, month)).getDay();

tbl = document.getElementById("calendar-body");

tbl.innerHTML = "";

monthAndYear.innerHTML = months[month] + " " + year;

selectYear.value = year;

selectMonth.value = month;

let date = 1;

for (let i = 0; i < 6; i++) {

let row = document.createElement("tr");

for (let j = 0; j < 7; j++) {

if (i === 0 && j < firstDay) {

cell = document.createElement("td");

cellText = document.createTextNode("");

cell.appendChild(cellText);

row.appendChild(cell);

}

else if (date > daysInMonth(month, year)) {

break;

}

else {

cell = document.createElement("td");

cell.id = day(date);

cellText = document.createElement("span");

cellText.innerHTML = date;

let sch = document.createElement("span");

let statusXhr = new XMLHttpRequest();

statusXhr.open('GET','http://127.0.0.1:7000/daily\_planner/record\_status/?data=' + day(date), true);

statusXhr.send();

statusXhr.addEventListener('readystatechange', function(){

if(statusXhr.readyState == 4 && statusXhr.status == 200){

let massage = JSON.parse(statusXhr.responseText);

if(massage.length != massage.ln\_success)

if(new Date().getMonth() > Number(selectMonth.value))

{

sch.innerHTML = '(' + massage.count + ')';

sch.parentElement.classList.add("bg-warning");

}

else if(new Date().getMonth() == Number(selectMonth.value) && new Date().getDate() >= Number(sch.previousSibling.innerHTML))

{

sch.innerHTML = '(' + massage.count + ')';

sch.parentElement.classList.add("bg-warning");

}

else

sch.parentElement.classList.add("bg-success");

}

})

if (date === today.getDate() && year === today.getFullYear() && month === today.getMonth()) {

cell.classList.add("bg-danger");

titleDay.innerHTML = date;

dateRecords(date);

}

cell.appendChild(cellText);

cell.appendChild(sch);

row.appendChild(cell);

date++;

}

}

tbl.appendChild(row);

}

}

// Функция возвращает количество дней в месяце

function daysInMonth(iMonth, iYear) {

return 32 - new Date(iYear, iMonth, 32).getDate();

}

//Отлов нажатия на число и вывод задач на это число

tbl.onclick = function(event) {

let target = event.target;

while (target != this) {

if (target.tagName == 'TD' && target.innerHTML != "") {

$('table td').removeClass('bg-info');

$(target).addClass('bg-info');

titleDay.innerHTML = target.firstChild.innerHTML;

dateRecords(target.firstChild.innerHTML);

var list = document.querySelector('ul');

list.addEventListener('click', function(ev) {

if (ev.target.tagName === 'LI') {

ev.target.classList.add('checked');

if(target.lastChild.innerHTML != "")

if(Number(target.lastChild.innerHTML[1]) - 1 != 0)

target.lastChild.innerHTML = '(' + String(Number(target.lastChild.innerHTML[1]) - 1) + ')';

else

{

target.lastChild.innerHTML = '';

target.classList.remove('bg-warning');

}

else

target.classList.remove('bg-success');

var request = new XMLHttpRequest();

request.open('POST', 'http://127.0.0.1:7000/daily\_planner/record\_status/', true);

request.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded; charset=UTF-8');

request.setRequestHeader('X-CSRFToken', csrfcookie());

var body = 'id=' + encodeURIComponent(ev.target.id);

request.send(body);

}

}, false);

return;

}

target = target.parentNode;

}

}

#### Приложение 4. Код модуля urls.py

from django.urls import path

from .views import \*

urlpatterns = [

path('', Index.as\_view(), name='index'), # Ссылка на главную страницу

path('login/', LoginFormView.as\_view(), name='login'), # Ссылка на страницу авторизации

path('logout/', LogoutView.as\_view(), name='logout'), # Ссылка на страницу деавторизации

path('add\_record/', AddRecord.as\_view(), name='add\_record'), # Ссылка на добавление задачи

path('record\_status/', RecordStatus.as\_view(), name='record\_status'), # Ссылка на получение статуса задачи

path('registration/', RegisterFormView.as\_view(), name='registration'), # Ссылка на регистрацию

]

#### Приложение 5. Код модуля models.py

from django.contrib.auth.models import User

from django.db import models

class Record(models.Model):

"""

Класс для реализации таблицы Record

"""

user = models.ForeignKey(User, on\_delete=models.CASCADE)

title = models.CharField(max\_length=255, default='')

creation\_date = models.DateTimeField(auto\_now\_add=True)

completion\_date = models.DateField()

status = models.CharField(max\_length=30, null=True, blank=True)

#### Приложение 6. Код модуля forms.py

from django import forms

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

from django.utils.translation import gettext\_lazy as \_

from .models import Record

class LoginForm(forms.Form):

"""

Класс для отображения формы авторизации

"""

username = forms.CharField(

max\_length=40,

required=True,

widget=forms.TextInput(attrs={'name': 'username', 'class': 'form-control'})

)

password = forms.CharField(

max\_length=100,

required=True,

widget=forms.PasswordInput(attrs={'name': 'password', 'class': 'form-control'})

)

class RecordForm(forms.ModelForm):

"""

Класс для отображения формы записи

"""

class Meta:

model = Record

fields = ['title', 'completion\_date']

widgets = {

'title': forms.TextInput(attrs={'class': 'form-control', 'name': 'title'}),

'completion\_date': forms.DateInput(attrs={'class': 'form-control', 'name': 'creation\_date',

'type': 'date'}),

}

labels = {

'title': 'Название',

'completion\_date': 'Дата выполнения',

}

class RegisterForm(UserCreationForm):

"""

Класс для отображения формы регистрации

"""

class Meta(UserCreationForm.Meta):

fields = (

User.USERNAME\_FIELD,

User.EMAIL\_FIELD,

'password1',

'password2',

)

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.email = forms.CharField(

label=\_('Email'),

strip=False,

widget=forms.EmailInput,

help\_text=\_('Enter your email.'),

)

def save(self, commit=True):

"""

Функция для проверки email и сохранения пользователя в БД

:param commit:

:return:

"""

user = super().save(commit=False)

user.email = self.cleaned\_data['email']

if commit:

user.save()

return user

1. don't repeat yourself [↑](#footnote-ref-1)
2. MariaDB [↑](#footnote-ref-2)
3. Do It Tomorrow [↑](#footnote-ref-3)