

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 21.Б11-мм

Приложение для поиска подходящего времени для проведения мероприятий

КРАЕВ Антон Иванович

Отчёт по производственной практике
в форме «Производственное задание»

Научный руководитель:
старший преподаватель каф. СП, Я. А. Кириленко

Санкт-Петербург
2024

Оглавление

Введение	3
1. Постановка задачи	4
2. Обзор	5
3. Реализация	6
3.1. Timetable API	6
3.2. Серверная часть	7
3.3. Клиентская часть	9
Заключение	10
Список литературы	11

Введение

Для планирования времени и места проведения мероприятий (факультатив, семинар, рабочая встреча, регулярная лекция и пр.) приходится открывать несколько вкладок Timetable[3] для каждого желаемого участника (группы, преподавателя) и смотреть их расписания для того, чтобы поставить мероприятие в удобное для всех время. Также нужно искать подходящую свободную аудиторию, просматривая расписание каждой. Иногда так же надо учитывать расписания и из других источников (например, из Google календаря). Это усложняется тем, что нужно учитывать особенности разных мероприятий, например:

- при планировании очных встреч: участники могут находиться не на факультете в желаемое время, нужно учитывать время на дорогу;
- при планировании регулярных встреч: нужно учитывать, что расписание может быть разным в разные недели (четные и нечетные);
- нормативные документы.

Хочется иметь возможность посмотреть всю необходимую информацию в одном месте и определить подходящее время и место проведения мероприятия.

1. Постановка задачи

Целью работы является упрощение выбора удобного времени и места проведения мероприятий путем реализации веб-приложения. Для её выполнения были поставлены следующие задачи:

1. Разработать API, расширяющий возможности Timetable API, для получения данных о расписании преподавателей, групп и аудиторий.
2. Спроектировать приложение, предусмотрев возможность добавления новых мероприятий и календарей.
3. Реализовать функциональность для работы с расписанием из Timetable.

2. Обзор

Для решения проблемы, описанной выше, уже было разработано приложение[2] (инструмент для составления индивидуальных графиков), однако оно имеет ряд недостатков:

- при каждом использовании данные собираются с Timetable, что может занимать немало времени, а также влечет зависимость работоспособности приложения от Timetable;
- покрывает только часть возможных мероприятий и ограничений, нет выбора аудиторий;
- при текущей архитектуре для расширения функциональности нужно достаточно сильно изменять существующий код.

Для решения задачи поиска удобного времени для мероприятий было принято решение переписать существующее приложение, предусмотрев возможность добавления новых видов мероприятий с их особенностями/ограничениями, а также личных календарей участников, помимо учебного расписания. А для повышения стабильности работы приложения использовать скрейпер[5] Timetable, который будет периодически собирать в базу данных (БД) всю информацию с Timetable. Для получения необходимых данных о расписаниях из БД реализовать API, расширяющий возможности Timetable API.

3. Реализация

3.1. Timetable API

Для получения необходимых данных с Timetable из предварительно собранной базы данных был разработан API. Для этого был выбран FastAPI — асинхронный Python-фреймворк с валидацией с Pydantic, автоматически генерирующий документацию в формате OpenAPI. Для запросов к БД используется SQLAlchemy ORM.

Функциональность API:

- Адреса
 - GET /addresses — список адресов всех факультетов
 - GET /addresses/{address_id}/classrooms — список аудиторий по адресу (на факультете)
- Аудитории
 - GET /classrooms — список всех аудиторий
 - GET /classrooms/{classroom}/events — расписание аудитории
- Преподаватели
 - GET /educators — список преподавателей
 - GET /educators/find — получение id преподавателя по ФИО
 - GET /educators/{educator_id}/events — расписание преподавателя
- Группы
 - GET /groups — список групп
 - GET /groups/find — получение id группы по названию
 - GET /groups/{group_id}/events — расписание группы

Также были найдены и исправлены ошибки в скрейпере timetable:

- Не сохранялась связь событий и их участников;
- Приложение бесконечно отправляло повторные запросы при ошибке. Timetable иногда отправляет неактуальные данные (например, список групп, где некоторые из них не существуют, и при попытке получить подробную информацию по одной из таких групп, выдает ошибку). Для того чтобы исключить блокировку приложения было добавлено ограничение количества попыток на запрос.

3.2. Серверная часть

Бэкенд[1] приложения написан на go (rest api), придерживаясь принципов чистой архитектуры.

Основные компоненты приложения:

- Router — отвечает за маршрутизацию поступающих запросов в нужный handler, логирует все запросы с помощью middleware;
- TimetableHandler — обрабатывает запрос, валидирует входные данные, вызывает методы сервиса и отправляет ответ клиенту;
- TimetableService — объединение всех операций в рамках одного бизнес-контекста, получает данные из кэша, а если их там нет, то использует http-клиент для запросов к timetable api и кэширует полученные данные;
- RedisRepository — инкапсуляция логики работы с хранилищем данных. Redis был выбран по следующим причинам:
 - Высокая скорость работы, так как данные хранятся в оперативной памяти, при этом потенциальная утеря данных в данной задаче не страшна;
 - Удобно хранить разнородные данные (key-value хранилище);
 - Автоматическое удаление старых данных.
- TimetableClient — http-клиент для работы с timetable api (с расширенным api скрейпера timetable).

Реализованные API методы:

- /timetable/educators — список преподавателей
- /timetable/educators/{educator_id} — мероприятия преподавателя
- /timetable/groups — список групп
- /timetable/groups/{group_id} — мероприятия группы
- /timetable/classrooms — список аудиторий
- /timetable/classrooms?name={classroom_name} — мероприятия в аудитории

Может показаться, что данное API дублирует API скрейпера Timetable, но их разделение кажется хорошим решением, так как уменьшается зависимость этих двух приложений, таким образом:

- Скрейпер не привязан к данному приложению и может использоваться другими
- Работу с API скрейпера можно легко заменить на любую другую реализацию получения данных с Timetable при необходимости

Расширение функциональности реализуется просто добавлением нового эндпоинта, так как:

- Для добавления нового источника достаточно реализовать http-клиент для работы с ним, остальное во многом можно переиспользовать;
- Все ограничения для мероприятий можно рассматривать как функцию, модифицирующую календари (например, если мы хотим учитывать время на то, чтобы добраться с одного факультета на другой, можем добавлять событие «в дороге» в календарь);
- Все проверки на возможность поставить мероприятие на определенное время — это просто функции, которые анализируют календари участников и выдают ответ.

3.3. Клиентская часть

Для фронтенда[4] найдена библиотека на React для удобного отображения расписаний нескольких участников на одной странице.

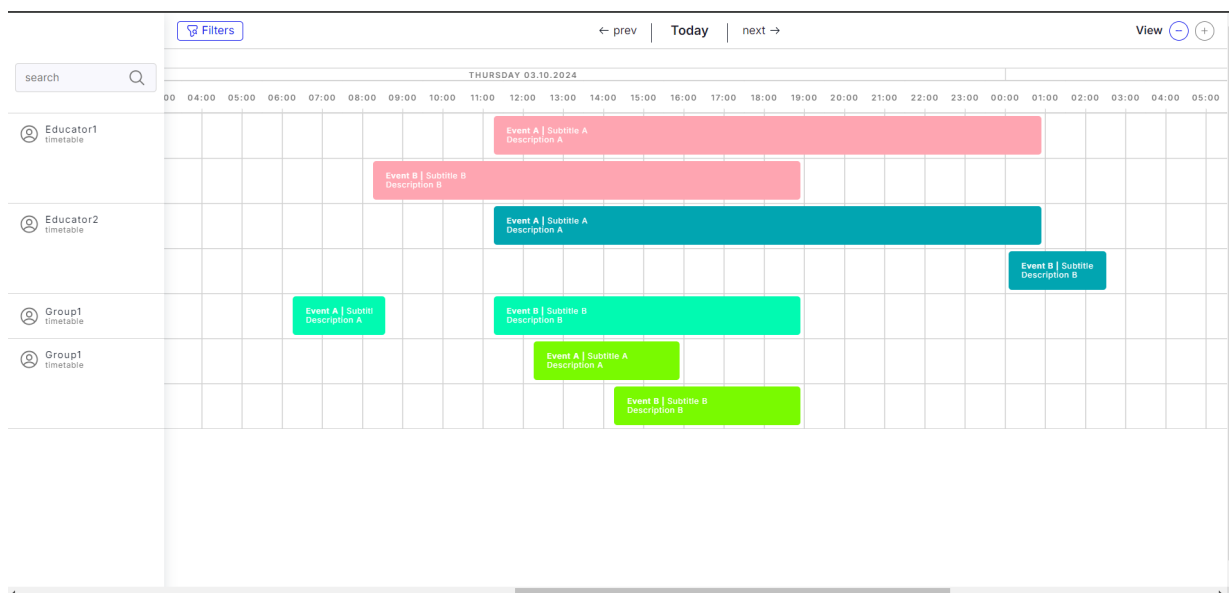


Рис. 1: Страница для планирования мероприятия

Заключение

В процессе работы были выполнены следующие задачи:

- Разработан API, расширяющий возможности Timetable API, для получения данных о расписании преподавателей, групп и аудиторий;
- Исправлены проблемы в работе скрейпера Timetable;
- Спроектировано приложение, предусмотрена возможность добавления новых мероприятий и календарей;
- Реализована функциональность для работы с расписанием из Timetable;
- Найдена библиотека для удобного отображения нескольких расписаний.

Код доступен в репозиториях[1, 4, 6] на GitHub.

Список литературы

- [1] Клиентская часть приложения для выбора времени и места проведения мероприятий. — URL: <https://github.com/Anton-Kraev/event-planner-frontend>.
- [2] Приложение для составления индивидуальных графиков. — URL: https://github.com/BadreevaPolina/individual_schedule.
- [3] Расписание СПбГУ (Timetable). — URL: <https://timetable.spbu.ru/>.
- [4] Серверная часть приложения для выбора времени и места проведения мероприятий. — URL: <https://github.com/Anton-Kraev/event-planner-backend>.
- [5] Скрейпер Timetable. — URL: <https://github.com/MinyazevR/timetable-database>.
- [6] Скрейпер Timetable с API. — URL: <https://github.com/Anton-Kraev/timetable-database>.