Новосибирский Государственный Университет
Факультет Информационных Технологий

Техническое описание проекта по курсу ООАД

«Менеджер расписания»

Студенты ФИТ НГУ
Неретин Степан Иванович
Кондренко Кирилл Павлович
Группа 21203

Версия 1.7

Содержание

Глава 1

Введение

1.1 Цель

Данный документ представляет собой техническое описание проекта «Менеджер расписания»и содержит основные требования к разрабатываемой в рамках проекта программной системе и описание архитектуры программного решения.

1.2 Область действия

Документ разработан в рамках проекта «Менеджер расписания»на основе стандартного шаблона и предназначен для использования студентами ФИТ и преподавателями дисциплины ООАД.

1.3 Определения и сокращения

Термин	Описание
Предметы по выборы	Предметы, разделенные по блокам, причем в каждом блоке студент расставляет их по приоритетам, но в конечном итоге из каждого блока для студента определён **лишь один** предмет

1.4 Ссылки

1.5 Краткое описание

Содержание данного документа построено таким образом, чтобы дать ответ на следующие вопросы:

- Какие проблемы предметной области должен решать будущий программный продукт

- Посредством какой функциональности системы будут достигнуто решение проблем предметной области
- Какова архитектура программного решения

Описание предметной области и проблем, для решения которых предназначен будущий программный продукт, приведены в разделе 2.

Раздел 3 содержит описание требований к программному решению, раздел 4 — описание архитектуры выбранного решения.

Глава 2

Предметная область проекта

Надстройка в виде «телеграмм-бота» или «расширения для браузера», улучшение текущего представление расписания для студентов НГУ посредством гибкого редактирования (скрытия «лишних» предметов, добавления новых).

2.1 Существующие проблемы

У многих студентов НГУ в расписании стоят предметы, которые они не посещают (например, это «предметы по выбору», из которых студент выбрал лишь один) или же нет предметов, на которые они ходят (например, студент может по своему желанию посещать семинары других преподавателей дополнительно к своим).

Решением этого было бы составление университетом индивидуального расписания для каждого из студентов, однако данное решение не реализуется университетом. Поэтому единственным решением в данном случае остаётся создание такой системы, с помощью которой студент сам бы мог редактировать своё расписание, при этом все изменения бы оставались и отображались там, где студент чаще всего смотрит своё расписание.

2.2 Предполагаемое решение

Ввиду того что чаще всего студенты смотрят своё расписание через сайт в браузере и в приложении от университета для телефона, решением будет создать «расширение для браузера», работающее в любом современном браузере, и «телеграмм-бота», которые должны быть взаимозаменяемыми и работать друг с другом.

Глава 3

Требования к программному решению

Данный раздел описывает требования к программной системе, разрабатываемой в рамках проекта «Менеджер расписания».

3.1 Роли

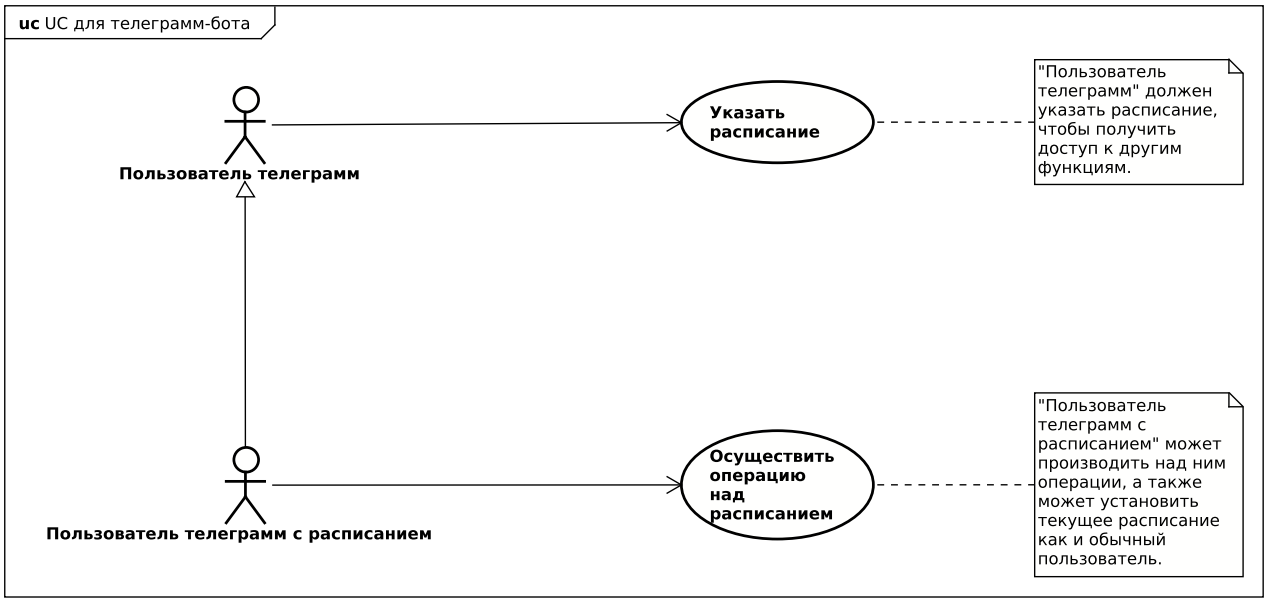
Роль — это что-то (например: другая система) или кто-то (например: человек) вне системы, которые взаимодействуют с ней. В предлагаемой к разработке системе идентифицированы следующие роли:

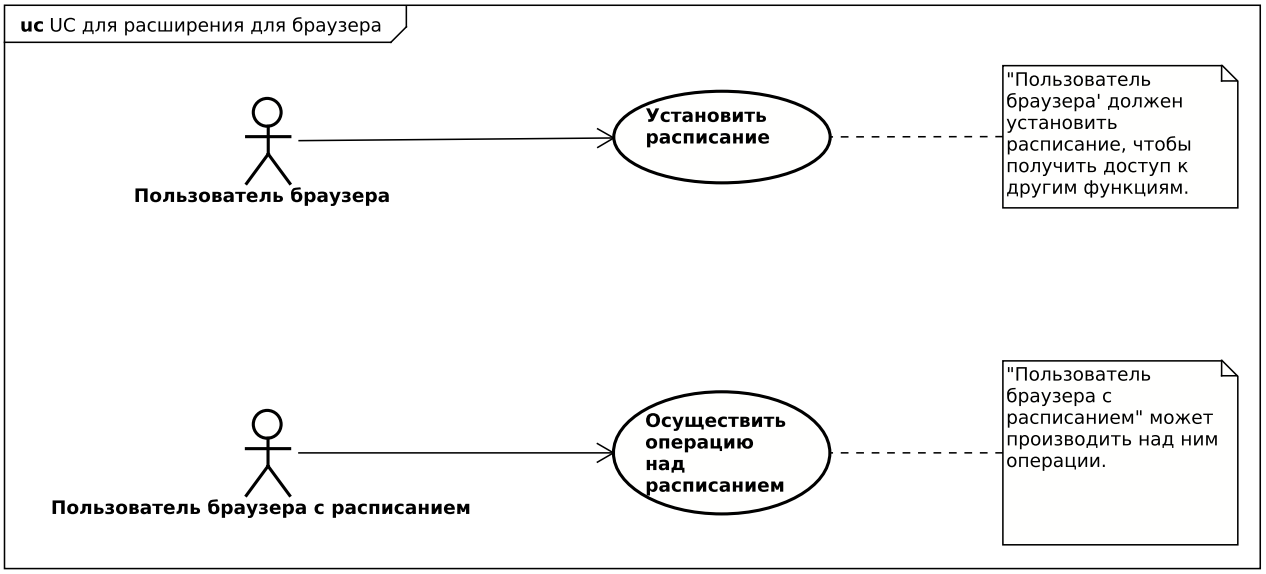
1. *«Пользователь телеграмм»* — человек, добавивший к себе бота в телеграмм;
2. *«Пользователь телеграмм с расписанием»* — пользователь телеграмм, указавший текущее расписание;
3. *«Пользователь браузера»* — человек, который зашёл на сайт с расписанием группы НГУ;
4. *«Пользователь браузера с расписанием»* – пользователь браузера, для которого определено текущее расписание.

3.2 Функциональные требования для роли *«Пользователь телеграмм»*

3.2.1 Указать расписание

«Пользователь телеграмм» должен указать расписание, чтобы получить доступ к другим функциям. Он может сделать несколькими способами:





- Импортировать расписание из файла с расписанием (подготовленного ранее с помощью телеграмм-бота или расширения для браузера);
- Указать номер группы;
- Оставить расписание пустым.

3.3 Функциональные требования для роли *«Пользователь телеграмм с расписанием»*

3.3.1 Осуществить операцию над расписанием

«Пользователь телеграмм с расписанием» может производить над ним операции, а именно он может:

- Показать расписание;
- Экспортировать расписание;
- Изменить расписание (добавить предмет, удалить предмет, изменить предмет).

«Пользователь телеграмм с расписанием» также может установить текущее расписание как и *«Пользователь телеграмм»*.

3.4 Функциональные требования для роли *«Пользователь браузера»*

3.4.1 Установить расписание

Как только *«Пользователь браузера»* зашёл на сайт с расписанием группы, текущее расписание устанавливается как расписание этой группы. Однако если по какой-то причине это не удалось сделать, то пользоваться расширением не получится. Если удалось установить расписание, то данное расписание сохраняется в кэш браузера.

Если *«Пользователь браузера»* ранее уже заходил на сайт с расписанием данной группы, то текущее расписание восстановится из кэша браузера.

3.5 Функциональные требования для роли *«Пользователь браузера с расписанием»*

3.5.1 Осуществить операцию над расписанием

«Пользователь браузера с расписанием» может производить над ним операции, а именно он может:

- Экспортировать расписание;
- Изменить расписанием (добавить предмет, удалить предмет, изменить предмет, установить расписание пустым, импортировать расписание).

При этом после любых изменений расписание сохраняется в кэш браузера, заменяя предыдущее, и страница браузера перерисовывается с учётом изменений.

3.6 Нефункциональные требования

Проект состоит из двух частей — телеграмм-бота и расширения для браузера, однако они должны иметь схожий пользовательский интерфейс и быть совместимы между собой, чтобы, например, можно было импортировать расписание, которое было экспортировано в другой части проекта.

Глава 4

Обзор архитектуры

Этот раздел описывает архитектуру проекта.

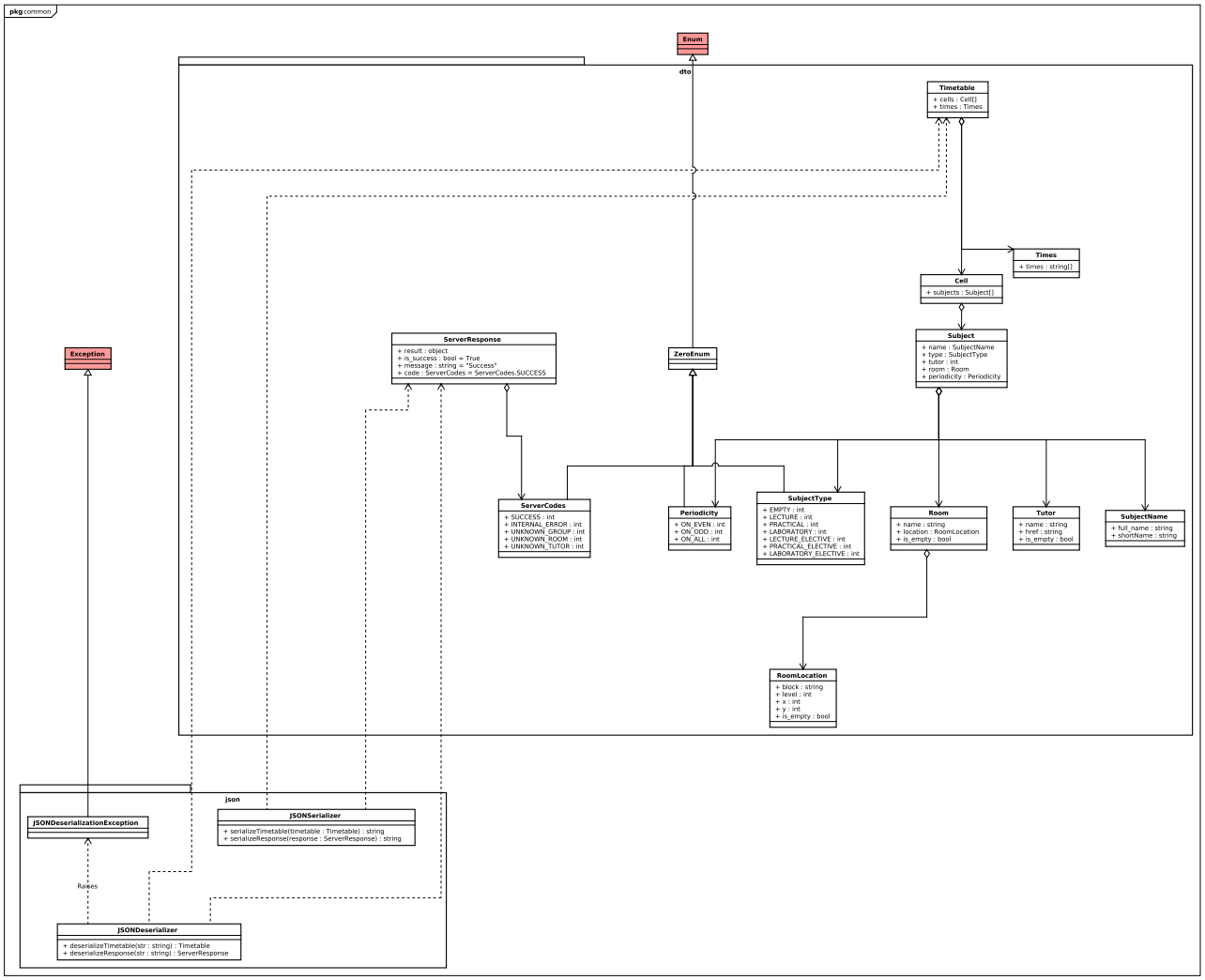
4.1 Подсистемы и компоненты проекта

Если не брать в расчёт компоненты сторонних производителей, то проект состоит из подсистем

- **Bot**;
- **Extension**;
- **Backend**.

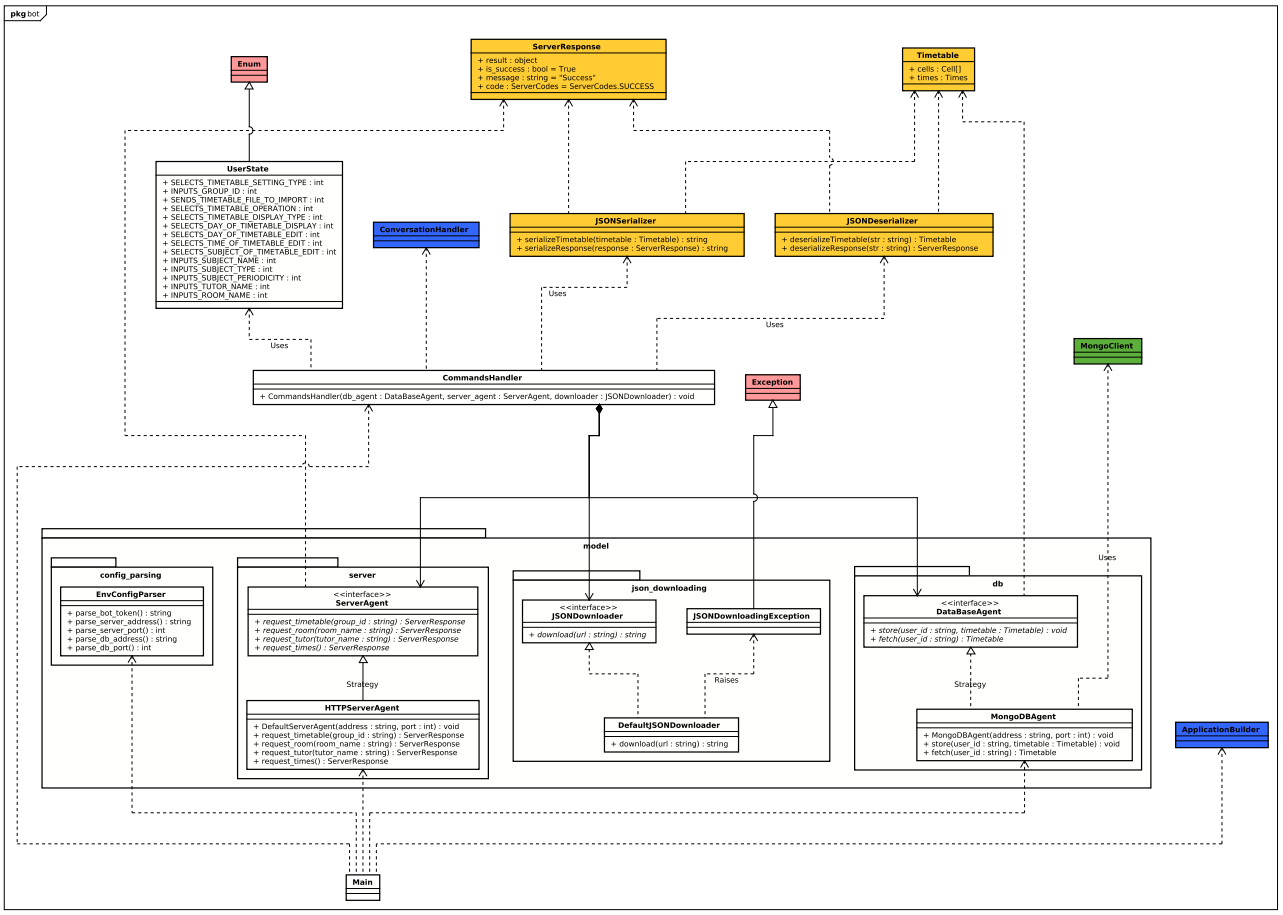
и компонента **Common**.

4.1.1 Common



Common является компонентом, который используют все подсистемы проекта. Он содержит основные DTO и способы их сериализации и десериализации в формат JSON и обратно.

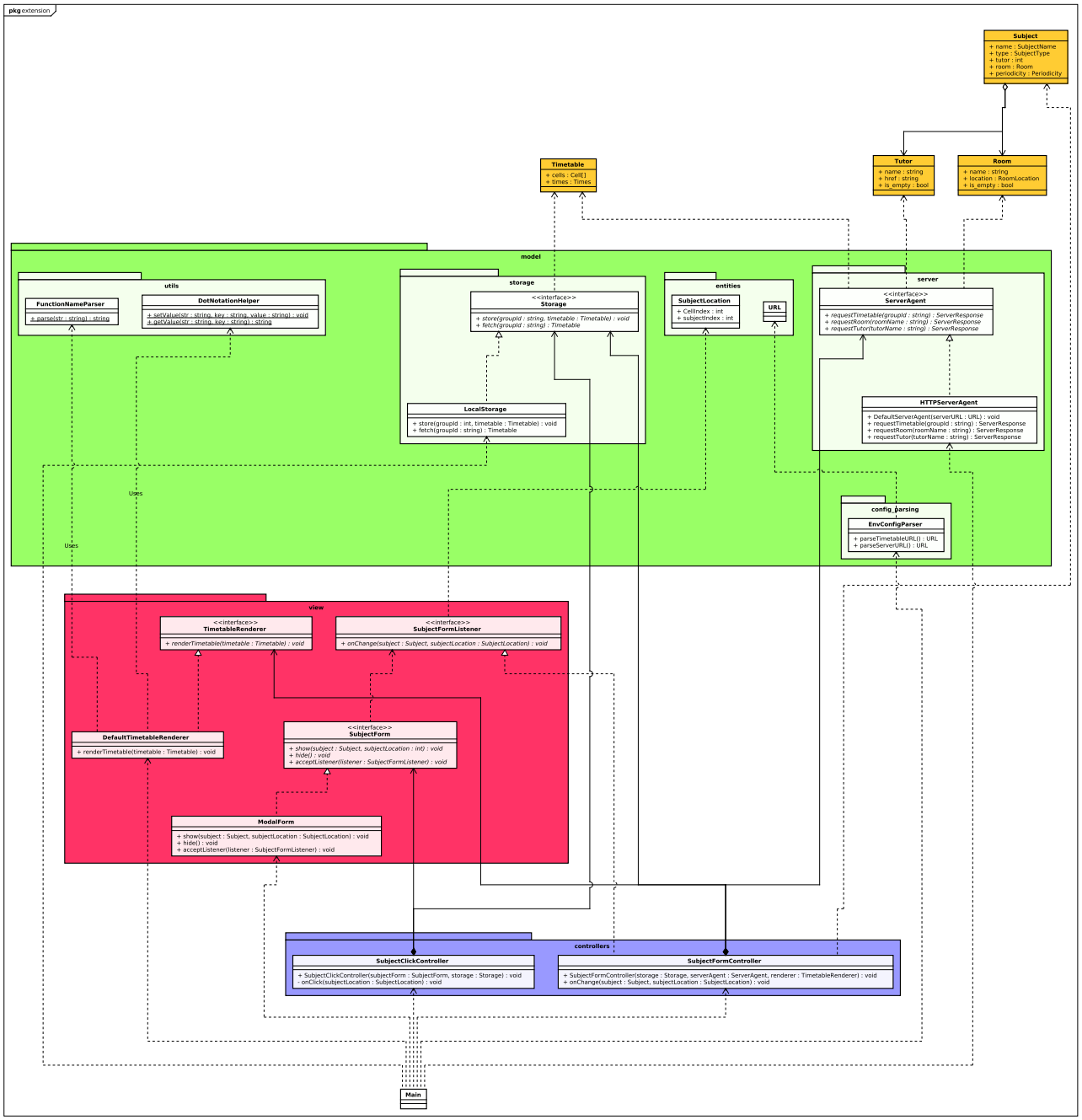
4.1.2 Bot



Bot является одним из фронтендов проекта, а именно фронтендом в мессенджере «Телеграмм». Выделим некоторые ключевые компоненты

- *EnvConfigParser* — класс для парсинга конфигурации из файла .env;
- *CommandsHandler* — класс для взаимодействия с пользователем телеграмм (поддержание диалога и обработка пользовательских сообщений).
- *HTTPServerAgent* — класс для взаимодействия с бэкендом;
- *MongoDBAgent* — класс для взаимодействия с нереляционной базой данных MongoDB.

4.1.3 Extension

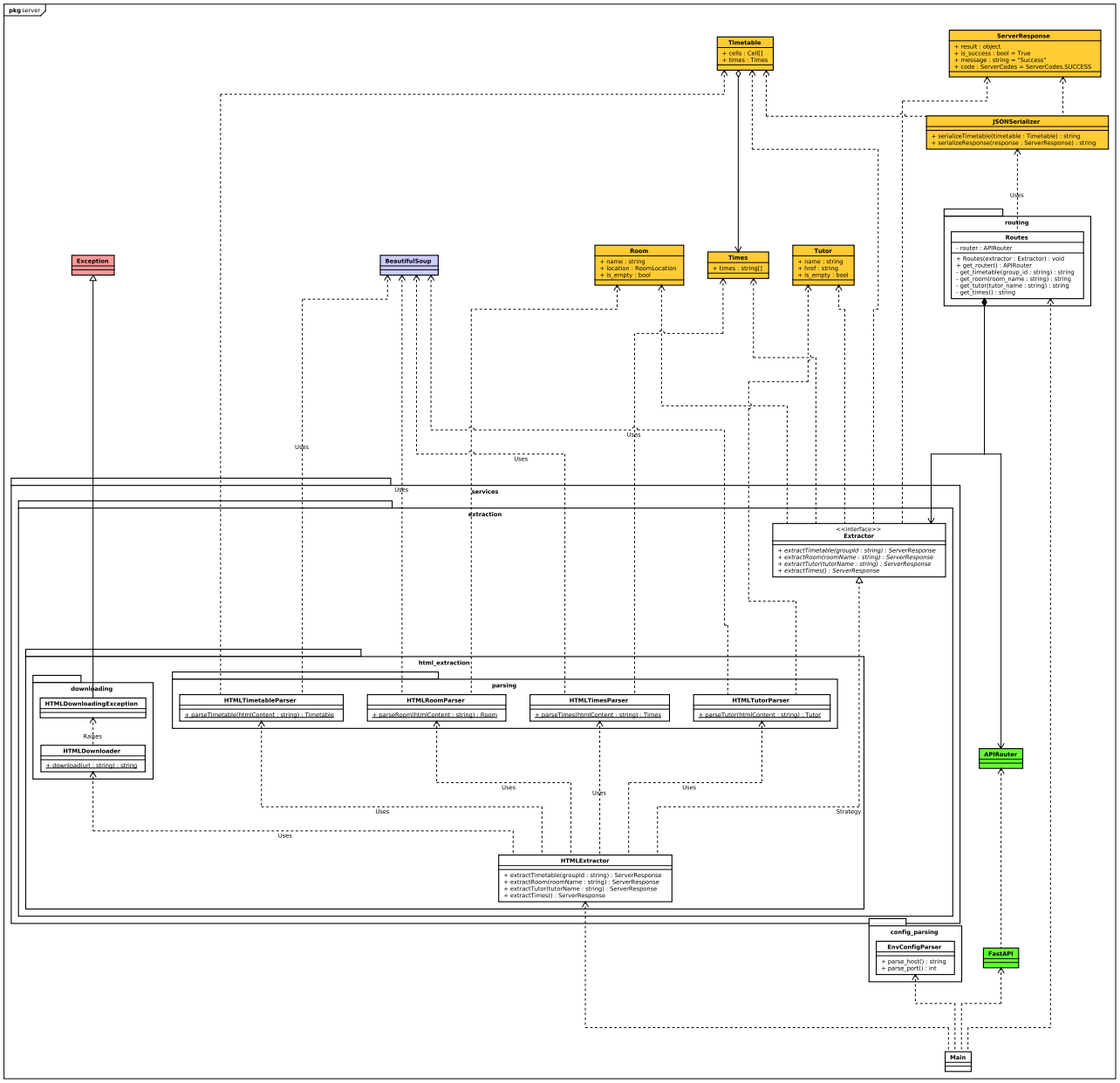


Extension является одним из фронтендов проекта, а именно фронтендом для браузера. Выделим некоторые ключевые компоненты

- *EnvConfigParser* — класс для парсинга конфигурации из файла .env;
- *HTTPServerAgent* — класс для взаимодействия с бэкендом;
- *LocalStorage* — класс для взаимодействия с кэшем браузера (сохранение данных в кэш и извлечение данных из кэша);
- *ModalForm* — класс, представляющий собой форму для изменения предмета в расписании;

- *DefaultTimetableRenderer* — класс, перерисовывающий страницу с расписанием при изменении предметов;
- *SubjectClickController* — класс-контроллер, детектирующий нажатие пользователя на предмет в расписании;
- *SubjectFormController* — класс-контроллер, детектирующий изменение предмета в форме.

4.1.4 Backend



Backend представляет собой REST-сервер с API, который по запросу парсит сайт с расписанием НГУ и возвращает информацию о расписании группы, аудитории, преподавателе или временах начала пар. Выделим некоторые ключевые компоненты

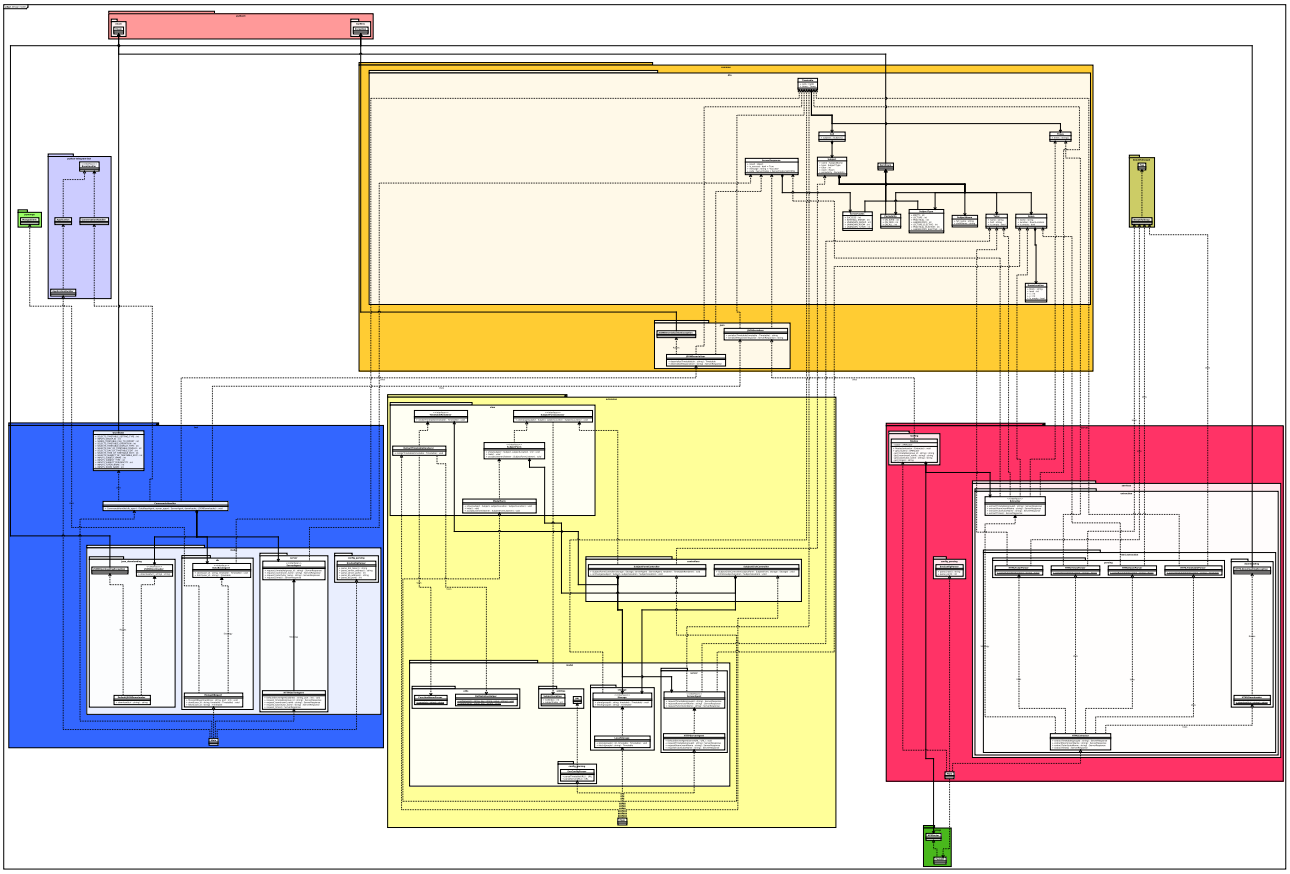
- *EnvConfigParser* — класс для парсинга конфигурации из файла .env;

- *Routes* — класс, инкапсулирующий создание конечных точек для API;
- *HTMLDownloader* — класс для скачивания HTML-страниц из интернета по URL;
- *HTMLTimetableParser* — класс для парсинга расписания из HTML-страницы;
- *HTMLRoomParser* — класс для парсинга аудитории из HTML-страницы;
- *HTMLTutorParser* — класс для парсинга преподавателя из HTML-страницы;
- *HTMLTimesParser* — класс для парсинга времён начала пар из HTML-страницы;
- *HTMLExtractor* — класс, извлекающий информацию с сайта с расписанием НГУ. Он использует класс *HTMLDownloader* для скачивания HTML-страниц и классы *HTMLTimetableParser*, *HTMLRoomParser*, *HTMLTutorParser*, *HTMLTimesParser* для парсинга скачанных страниц.

4.2 Компоненты сторонних производителей

- **python-telegram-Botbf** — библиотека под Python для работы с API-телеграмма. Используется в подсистеме **Bot**.
- **python-dotenv** — библиотека под Python для парсинга файлов .env. Используется в подсистемах **Bot** и **Backend** для парсинга начальной конфигурации.
- **beautifulsoup4** —- библиотека под Python для парсинга текстовых документов. Используется в подсистеме **Backend** для парсинга информации из HTML-документов.
- **fastapi** — библиотека под Python для создания своего API. Используется в подсистеме **Backend** для создания конечных точек.
- **pymongo** — библиотека под Python, представляющая собой драйвер для работы с нереляционной базой данных MongoDB. Используется в подсистеме **Bot** для сохранения и получения текущего расписания пользователя.

4.3 Диаграмма аналитических классов проекта



4.4 Схема развёртывания проекта

Развёртывание Extension

Сборка расширения в файлы bundle.js и manifest.json и дальнейшее добавление в каталог расширений современных браузеров.

Развёртывание Bot и Backend

При изменениях в коде GitHub Actions запускает предварительно настроенный workflow. Виртуальная машина на GitHub Actions создает Docker-образы из актуального кода, собранные образы помещаются в Docker Hub. После этого останавливаются и удаляются старые Docker-контейнеры на VPS, а контейнеры с новыми Docker-образами создаются и запускаются, после чего **Bot** и **Backend** автоматически перезапускаются.

Глава 5

Допущения и ограничения

При разработке проекта были принятые *следующие допущения*

- К бэкенду будут обращаться лишь фронтенды, но не сторонние пользователи сети интернет (ведь они могут это сделать, зная адрес фронтенда в сети интернет);
- На бэкенд не будет совершаться никаких DOS или DDOS-атак в сети интернет;
- Количество пользователей телеграмм-бота не будет значительным, чтобы их состояние могло сохраняться локально на том устройстве, на котором запущен бот (а не в базе данных);
- Бэкенд будет работать всегда;
- Вёрстка сайта с расписанием НГУ никогда не будет меняться.

Глава 6

Известные проблемы

6.1 Отсутствие поддержки JWT

Проблема	Для доступа к данным API бэкенда не требуется никакая авторизация
Ранг	3 (низкий)
Влияние на проект	Любой пользователь интернета может получать данные с бэкенда, что увеличивает нагрузку на него. Как следствие, оба фронтенда могут начать работать медленнее
Пути решения	Добавить поддержку JWT, чтобы доступ к бэкенду имелся лишь у авторизованных приложений (только у расширения, и только у бота), а не у любого пользователь интернет, знающего адрес бэкенда в сети интернет

6.2 Отсутствие защиты от DOS и DDOS-атак

Проблема	У бэкенда нет никакой защиты от DOS или DDOS-атак, поэтому при любой такой атаке он перестанет обрабатывать запросы фронтендов
Ранг	5 (средний)
Влияние на проект	При любой DOS или DDOS-атаке, оба фронтенда перестанут полноценно работать
Пути решения	Использовать интернет сервисы для защиты от DOS и DDOS-атак. Например, CloudFlare

6.3 Сохранение состояние пользователей телеграмм не в базе данных

Проблема	Текущее состояние пользователей (не их расписание) телеграмм не сохраняется в базу данных, а хранится локально на том устройстве, на котором запущен телеграмм-бот
Ранг	5 (средний)
Влияние на проект	При большом количестве пользователей телеграмм-бота, может закончится память на устройстве, на котором запущен телеграмм-бот, а значит он перестанет обрабатывать запросы новых пользователей или вовсе прекратит работу
Пути решения	Сохранять состояние пользователей в какую-нибудь удалённую базу данных

6.4 Невозможность работы фронтендов без бэкенда

Проблема	Если не работает бэкенд, то оба фронтенда не могут полноценно работать
Ранг	7 (Высокий)
Влияние на проект	При критических проблемах на стороне фронтенда оба фронтенда не могут полноценно работать
Пути решения	Избавиться от бэкенда и осуществлять парсинг сайта с расписанием НГУ на каждом фронтенде. Однако это приведёт к дублированию логики и потенциальному расхождении спецификации. Также при добавлении нового фронтенда, эту логику придётся заново реализовывать на нём (если он будет написан на языке программирования, отличном от уже существующих на фронтендах)

6.5 Зависимость бэкенда и расширения браузера от вёрстки сайта с расписанием НГУ

Проблема	Бэкенд и расширение браузера напрямую зависят от вёрстки сайта с расписанием НГУ
Ранг	8 (Высокий)
Влияние на проект	Если у сайта с расписанием НГУ поменяется вёрстка, то для полноценной работы системы придётся изменять бэкенд и отображение страниц в расширении
Пути решения	Использовать API, которое используют сайты с расписанием НГУ, однако данное API если и существует, то не является открытым

Лист регистрации изменений

Дата	Версия	Описание	Автор
06.09.2023	1.0	Заполнены разделы 1 и 2	Неретин Степан, Кондренко Кирилл
30.09.2023	1.1	Смоделирована usecase-модель телеграмм-бота	Кондренко Кирилл
02.10.2023	1.2	Смоделирована usecase-модель расширения для браузера	Неретин Степан, Кондренко Кирилл
02.10.2023	1.3	Заполнен раздел 3	Неретин Степан, Кондренко Кирилл
07.10.2023	1.4	Уточнён раздел 3	Кондренко Кирилл
22.10.2023	1.5	Уточнён раздел 3, и смоделирована аналитическая модель для телеграмм-бота	Кондренко Кирилл
27.10.2023	1.6	Уточнён раздел 3, и смоделирована аналитическая модель для расширения для браузера	Неретин Степан, Кондренко Кирилл
22.11.2023	1.7	Изменён раздел 1, смоделирована дизайн-модель и заполнены разделы 4, 5, 6	Неретин Степан, Кондренко Кирилл

Лист регистрации проверок

Дата	Версия	Описание	Автор