Московский государственный технический университет

имени Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Программное обеспечение ЭВМ и

информационные технологии»

**Технический проект**

**«Системы выдачи и сбора заданий по английскому языку»**

Пахомов А.А., студент ИУ7-27

Москва, 2017

Оглавление

[Глоссарий 3](#_Toc482440569)

[Техническое задание 4](#_Toc482440570)

[Раздел 1. Общие сведения 4](#_Toc482440571)

[Раздел 2. Назначение и цели создания системы 4](#_Toc482440572)

[Раздел 3. Требования к системе 5](#_Toc482440573)

[3.1. Требования к системе в целом. 5](#_Toc482440574)

[3.2. Технические требования. 5](#_Toc482440575)

[3.3. Требования к порталу с точки зрения пользователя 6](#_Toc482440576)

[3.4. Входные параметры системы 6](#_Toc482440577)

[3.5. Выходные параметры системы 7](#_Toc482440578)

[3.6. Требования к надежности 7](#_Toc482440579)

[Раздел 4. Порядок контроля и приёмки системы 7](#_Toc482440580)

[Раздел 5. Требования к документированию 7](#_Toc482440581)

[Топология системы 9](#_Toc482440582)

[Функциональные требования по подсистемам 10](#_Toc482440583)

[Концептуальный дизайн 12](#_Toc482440584)

[Варианты использования системы 14](#_Toc482440585)

[Отправка выполненного задания 14](#_Toc482440586)

[Добавление задания к уроку 15](#_Toc482440587)

[Проверка задания 15](#_Toc482440588)

[Логическая структура 16](#_Toc482440589)

[Спецификации классов 17](#_Toc482440590)

# Глоссарий

|  |  |
| --- | --- |
| **Термин** | **Определение** |
| Валидация  данных | Проверка на корректность, полноту и непротиворечивость входных, выходных и обрабатываемых данных |
| REST | [Архитектурный стиль](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) взаимодействия компонентов распределённого приложения в [сети](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C). |
| UUID | Уникальный идентификатор пользователя. |
| WEB-интерфейс | Интерфейс пользователя, предоставляемой системой через Web-браузер. В разрабатываемой системе только один веб-интерфейс. |
| СОА (SOA) | Сервис-ориентированная архитектура (Service Oriented Architecture), [модульный](https://ru.wikipedia.org/wiki/Модульность_(программирование)) подход к разработке [программного обеспечения](https://ru.wikipedia.org/wiki/Программное_обеспечение), основанный на использовании [распределённых](https://ru.wikipedia.org/wiki/Распределённые_вычисления), [слабо связанных](https://ru.wikipedia.org/w/index.php?title=Слабое_связывание&action=edit&redlink=1) заменяемых компонентов, оснащённых стандартизированными [интерфейсами](https://ru.wikipedia.org/wiki/Интерфейс_программирования_приложений) для взаимодействия по стандартизированным [протоколам](https://ru.wikipedia.org/wiki/Сетевой_протокол). |
| Internet | Всемирная система объединённых компьютерных сетей для хранения и передачи информации. |
| CI (Непрерывная интеграция) | Практика разработки программного обеспечения, которая заключается в слиянии рабочих копий в общую основную ветвь разработки несколько раз в день и выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем. |
| Docker | Программное обеспечение для автоматизации развёртывания и управления приложениями в среде [виртуализации на уровне операционной системы](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%B0_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B). |
| Git | Система управления версиями. |
| GitHub | Крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. |
| Профиль пользователя | Информация о пользователе портала, хранящаяся в портале, в частности, имя, фамилия, фотография и др. |
| Сессия | Сессия на сайте - серия запросов к порталу, сделанных одним пользователем в заданный промежуток времени, в данном документе - в течение 30 минут. |
| Фронтенд | Серверное приложение, принимающее запросы от пользователя портала. На каждый из типов запросов от пользователя (показать профиль, вывести список заданий и др.) фронтенд определяет, как организовать выполнение запроса. Фронтенд принимает запросы от пользователя, анализирует их и в соответствии с заложенным алгоритмом выполняет запросы к бэкендам. |
| Бэкенд | Серверное приложение, выполняющее определенную задачу, например, взаимодействие с СУБД. Бэкенды принимают запросы от фронтенда. |
| Сервер | Компьютер, выполняющий функции обслуживания пользователей при доступе к информационным ресурсам в вычислительных системах. |
| Проект, портал, система | В данной работе термины «проект», «портал» и «система» взаимозаменяемы. |

## 

# Техническое задание

## Раздел 1. Общие сведения

Данное техническое задание составлено для разработки проекта «Система выдачи и сбора заданий по иностранному языку». Техническое задание выполнено на основе ГОСТ 19.201—78 «ЕСПД. Техническое задание. Требования к содержанию и оформлению».

Разработка ведется в рамках выполнения лабораторных работ по курсу Методология программной инженерии на кафедре «Программное обеспечение ЭВМ и информационные технологии» факультета «Информатика и системы управления» МГТУ им. Н. Э. Баумана.

## Раздел 2. Назначение и цели создания системы

Система разрабатывается для преподавателей английского языка в технических университетах. Предполагается, что студенты с помощью описываемой системы будут получать домашние задания по английскому языку, выполнять их и загружать на сайт. Преподаватель сможет проверять и выдавать новые задания. Все задания выдаются и собираются в текстовом виде.

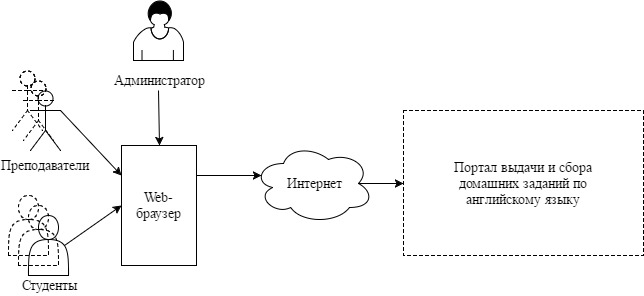


Рисунок 1 Схема предметной области

## Раздел 3. Требования к системе

### 3.1. Требования к системе в целом.

При разработке Системы должны быть учтены следующие принципы:

1. многопользовательский режим работы;
2. однократный ввод информации в Систему и многократное ее использование в различных подсистемах;
3. обеспечение Web-интерфейса для доступа ко всем пользовательским функциям Системы, позволяющего работать с документами посредством сети Internet;
4. информирование пользователей о событиях, требующих их внимания;
5. модульность – структурирование решения на функциональные блоки, отвечающие за выполнение отдельных задач с возможностью поэтапной реализации;
6. масштабируемость – возможность увеличения производительности при возрастании числа пользователей и объемов информационных потоков без внесения кардинальных изменений в архитектуру и логику функционирования;
7. функциональная адаптивность – возможность наращивания функциональных возможностей без внесения кардинальных изменений в архитектуру и логику функционирования платформы и ее составных частей;
8. надежность – использование технологий резервирования для обеспечения отказоустойчивости (по модели не менее N+1);
9. обработка и хранение информации должны производиться централизованно на серверах системы, в роли клиентского приложения должен выступать стандартный интернет-браузер, через которые должны быть доступны все основные функции системы для пользователей.

### 3.2. Технические требования.

1. Всё взаимодействие выполняется по REST.
2. Связь пользователей между системами выполняется с помощью UUID.
3. Все системы должны собираться и разворачиваться через CI.
4. Каждая система должна иметь свою собственную базу данных, прямой запрос к сторонней базе запрещен.
5. Предусмотреть возможность масштабирования сервисов.
6. Использовать Git и GitHub.

### 3.3. Требования к порталу с точки зрения пользователя

Портал должен обеспечивать реализацию следующих функций:

1. Система должна обеспечивать регистрацию пользователей

с валидацией вводимых данных.

1. Система должна обеспечивать аутентификацию пользователей.
2. Система должна обеспечивать разделение пользователей на три роли:
   * студент;
   * преподаватель;
   * администратор.
3. Система должна предоставлять **студенту** следующие функции:

* получение задания;
* отправка выполненного задания;
* редактирование профиля;
* просмотр информации о преподавателе: ФИО, фотография, учёная степень, краткая информация.

1. Система должна предоставлять **преподавателю** следующие функции:
   * выдача заданий;
   * редактирование профиля;
   * просмотр информации о студенте: ФИО, фотография, краткая информация.
2. Система должна предоставлять **администратору** неограниченные полномочия по изменению контента на портале.

### 3.4. Входные параметры системы

**Студент**

* Фотография студента в формате JPEG, размером не более 500КБ

и разрешением не более 500x500 пикселей.

* Имя, фамилия и отчество (при наличии), максимальная длина каждого из них 64 символа.
* Краткая информация о себе. Максимальная длина текста – 1024 символа.
* Выполненное домашнее задание – текстовое поле. Максимальная длина текста – 8192 символа.

**Преподаватель**

* Фотография преподавателя в формате JPEG, размером не более 500кб

и разрешением не более 500x500 пикселей.

* Имя, фамилия и отчество (при наличии), максимальная длина каждого из них 64 символа.
* Краткая информация о себе. Максимальная длина текста – 1024 символа.
* Тема урока – текстовое поле. Максимальная длина текста – 256 символов.
* Домашнее задание – текстовое поле. Максимальная длина текста – 8192 символа.
* Оценка за домашнее задание.

### 3.5. Выходные параметры системы

Выходными параметрами системы являются веб-страницы. Они должны содержать

следующую информацию:

* страница профиля пользователя системы (студента или преподавателя);
* страница урока: тема, домашнее задание, фильтр по студентам для просмотра их ответа и выставления оценки.

### 3.6. Требования к надежности

Необходимо предусмотреть ситуацию недоступности систем, обработку таймаутов и ошибок сервисов. В случае ошибки/недоступности некритичного функционала выполнять деградацию функциональности.

## Раздел 4. Порядок контроля и приёмки системы

1. Показать процесс сборки и развёртывания проекта.
2. Продемонстрировать работоспособность в случае доступности всех сервисов и поведение системы в случае недоступности одного или нескольких сервисов.
3. Отдельно продемонстрировать обработку медленного ответа сервиса.
4. Продемонстрировать возможность масштабирования сервисов (устное объяснение, без самой демонстрации).
5. Расчетно-пояснительная записка с диаграммами, описывающими функционирование системы.

## Раздел 5. Требования к документированию

Исполнитель должен подготовить и передать Заказчику следующие документы:

* руководство по развертыванию Системы;
* руководство администратора Системы;
* информацию о обрабатываемых открытых данных (сведения о полях);
* руководство для клиента по использованию Системы.

# Топология системы

На рис. 2 приведена топология разрабатываемой системы.

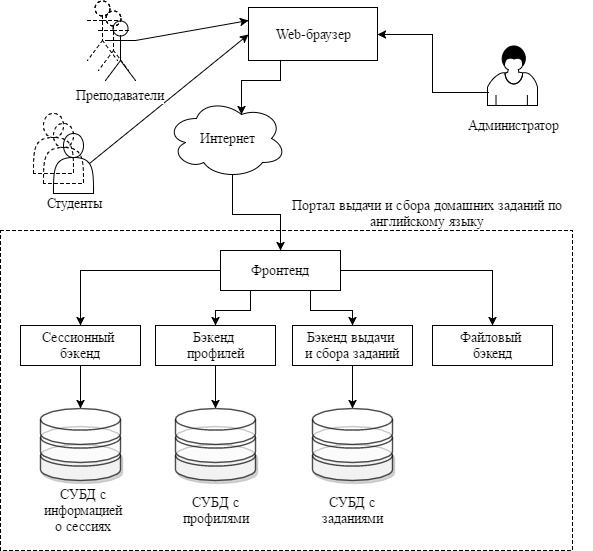


Рисунок 2. Топология системы

Система будет состоять из фронтенда и четырех бэкендов, что наиболее целесообразно для реализации ее основного назначения.

* **Сессионный бэкенд** отвечает за сессию пользователей портала и реализует

следующие функции:

* + регистрация пользователя (клиента или мастера);
  + аутентификация (проверка сессии) пользователя;
  + авторизация пользователя (вход, или «логин»);
  + выход из сессии («логаут»).
* **Файловый бэкенд** отвечает за хранение данных в файлах, которые содержат

фотографии пользователей и фотографии работ мастеров.

* **Бэкенд профилей** реализует следующие функции:
  + получение, изменение, удаление конкретного профиля;
  + добавление нового профиля.

В профиле хранится следующая информация о **студенте**:

* имя, фамилия, отчество;
* краткая информация о себе;
* фотография.

В профиле хранится следующая информация о **преподавателе**:

* имя, фамилия, отчество;
* краткая информация о себе;
* фотография.
* **Бэкенд выдачи и сбора заданий** реализует следующие функции:
  + добавление нового задания;
  + добавление выполненного задания;
  + получение задания к уроку;
  + получение ответа студента на задание.

Фронтенд принимает запросы от пользователей по протоколу HTTP и анализирует их. На основе проведенного анализа фронтенд выполняет запросы к бэкендам, агрегирует ответы бекендов и отсылает ответ пользователю.

## Функциональные требования по подсистемам

1. ***Фронтенд*** – это серверное приложение при разработки которого необходимо учитывать следующие факторы:

* фронтенд должен принимать запросы по протоколу HTTP и формировать ответы пользователям портала в формате HTML;
* в зависимости от типа запроса фронтенд должен отправлять последовательные запросы в соответствующие бэкенды;
* запросы к бэкендам осуществляются по протоколу HTTP. Данные необходимо передавать в формате JSON. Данный текстовый формат обмена данными удобен для чтения;
* целесообразно использовать библиотеку Bootstrap для создания HTML-верстки.

1. ***Сессионный бэкенд*** – это серверное приложение, которое должно отвечать следующим требованиям по разработке.

* Сессионный бекенд должен принимать и возвращать данные в формате JSON по протоколу HTTP.
* Выполнять авторизацию пользователей, проверять и удалять сессию, а также

регистрировать пользователей.

* Резервирование базы данных должно производиться по расписанию.
* Необходимо предусмотреть разработку скрипта для автоматического создания копий базы данных.

1. ***Файловый бэкенд*** должен быть серверным приложением, которое:

* должно принимать и отвечать на запросы в формате JSON по протоколу HTTP;
  + приложение должно уметь обрабатывать запросы на загрузку файлов в бэкенд и получение файлов из бекенда.

1. ***Бэкенд профилей*** должен быть серверным приложением, которое:

* должно принимать и отвечать на запросы в формате JSON по протоколу HTTP;
* обрабатывать запросы на создание, удаление, редактирование и получение профиля пользователя по ключу – идентификатору пользователя.

При разработке базы данных, содержащей информацию о профилях, требуется учитывать следующие требования:

* + необходимо разработать скрипт для автоматического создания резервной копий базы данных по расписанию;
  + первичным ключом является идентификатор пользователя;

1. ***Бэкенд выдачи и сбора заданий*** должен быть серверным приложением, которое:

* принимает и отвечает на запросы в формате JSON по протоколу HTTP;
* обрабатывает запросы **преподавателя** на добавление задания к уроку и получение ответа конкретного студента;
* обрабатывает запросы **студента** на добавление ответа к заданию;
* необходимо разработать скрипт для автоматического создания резервной копий базы данных по расписанию;

# Концептуальный дизайн

На рисунке 3 приведена концептуальная модель системы в нотации IDEF0.

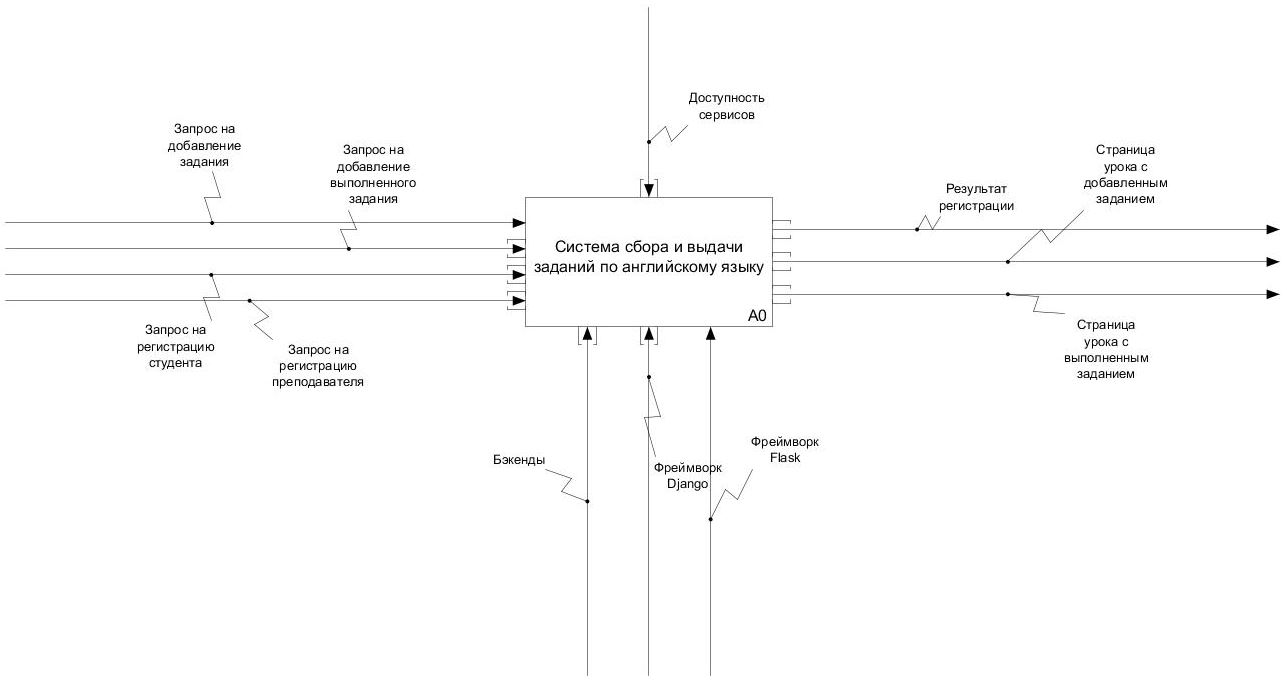


Рисунок 3. Концептуальная модель системы в нотации IDEF0

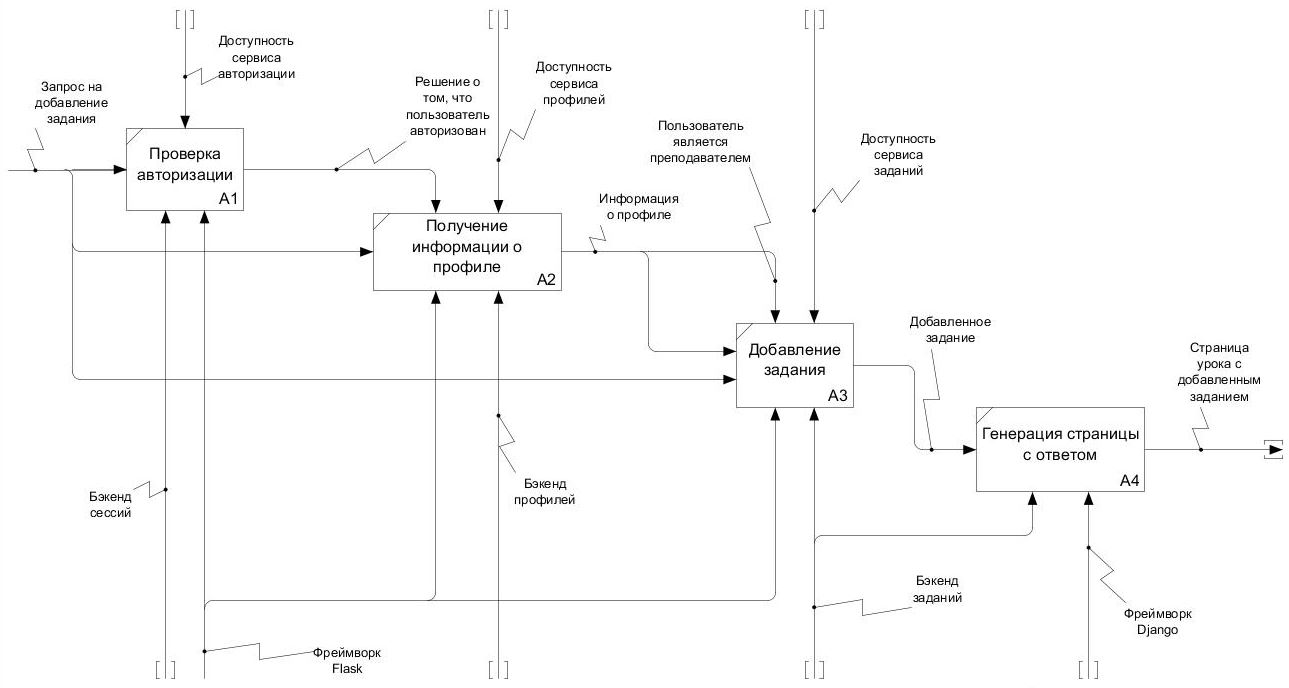
На рисунке 4 приведена детализированная концептуальная модель.

Рисунок 4. Детализированная концептуальная модель системы в нотации IDEF0.

# Варианты использования системы

На рисунке 5 представлена диаграмма прецедентов.

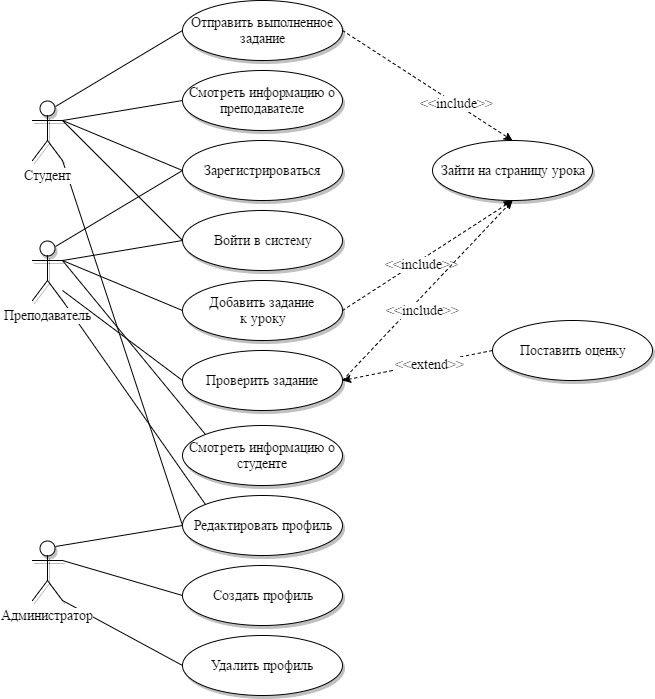


Рисунок 5. Диаграмма прецедентов

Далее представлены спецификации вариантов использования.

## Отправка выполненного задания

**Действующее лицо:**студент

**Цель:**отправить выполненное задание

**Предусловия:**студент осуществил вход в систему, задание к уроку добавлено

**Главная последовательность:**

1. студент заходит на страницу урока;
2. студент нажимает на кнопку «Ответить»;
3. система показывает текстовое поле для записи в него ответа на урок;
4. студент пишет ответ и нажимает кнопку «Отправить».

**Альтернативная последовательность**(студент уже отправлял ответ на это задание):

1. студент заходит на страницу урока;
2. студент видит свой предыдущий ответ и нажимает на кнопку «Редактировать»;
3. система показывает текстовое поле с предыдущим ответом на урок;
4. студент редактирует ответ и нажимает кнопку «Отправить».

## Добавление задания к уроку

**Действующее лицо:**преподаватель

**Цель:**добавить задание к уроку

**Предусловия:**преподаватель осуществил вход в систему

**Главная последовательность:**

1. преподаватель заходит на страницу урока;
2. преподаватель нажимает на кнопку «Добавить задание»;
3. система показывает текстовое поле для записи в него задания к уроку;
4. преподаватель пишет задание и нажимает кнопку «Добавить».

**Альтернативная последовательность**(задание уже было добавлено):

1. преподаватель заходит на страницу урока;
2. преподаватель нажимает на кнопку «Редактировать задание»;
3. система показывает текстовое поле для записи в него задания к уроку;
4. преподаватель пишет задание и нажимает кнопку «Добавить».

## Проверка задания

**Действующее лицо:**преподаватель

**Цель:**проверить задание к уроку

**Предусловия:**преподаватель осуществил вход в систему

**Главная последовательность:**

1. преподаватель заходит на страницу урока;
2. система показывает список студентов, оставивших ответы;
3. преподаватель нажимает на поле с записью о студенте и разворачивается ответ студента на задание к уроку;
4. ниже ответа студента есть выбор оценки за задание и кнопка «Поставить оценку».

**Альтернативная последовательность**(задание уже было оценено):

1. преподаватель заходит на страницу урока;
2. система показывает список студентов, оставивших ответы;
3. преподаватель нажимает на поле с записью о студенте и разворачивается ответ студента на задание к уроку;
4. ниже ответа студента написана оценка за задание и кнопка «Изменить оценку».

На рисунке 6 представлена диаграмма деятельности для прецедента «Проверить задание».

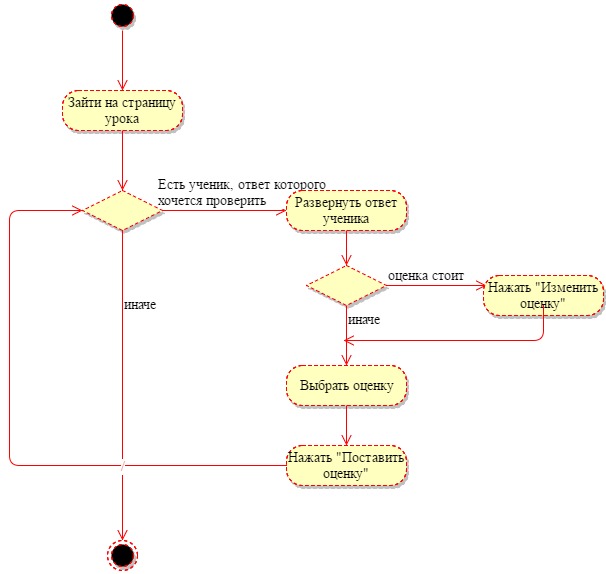
****

Рисунок 6. Диаграмма деятельности для прецедента "Проверить задание"

# Логическая структура

На рисунке 7 представлена диаграмма классов системы.

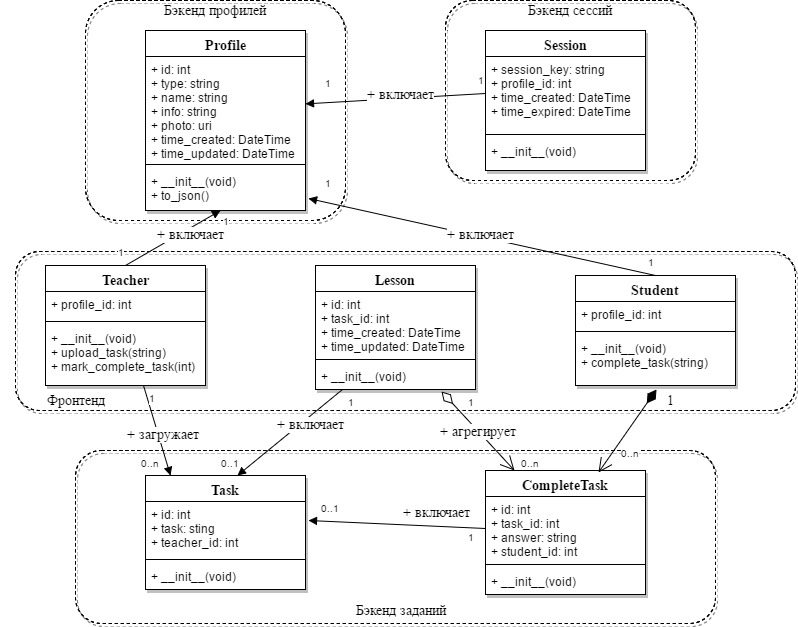


Рисунок 7. Диарамма классов

## Спецификации классов

Класс **Profile** представляет собой профиль преподавателя или студента. Класс используется в бэкенде профилей.

Методы классаProfile:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| \_\_init\_\_(void) | Конструирует объект профиля |
| to\_json(): string | Возвращает JSON-представление профиля. |

Атрибуты класса Profile:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| id | public: int | Идентификатор профиля |
| type | public: string | Тип профиля |
| name | public: string | ФИО |
| info | public: string | Краткая информация |
| photo | public: uri | Адрес расположения фотографии |
| time\_created | public: DateTime | Время создания |
| time\_updated | public: DateTime | Время последнего обновления |

Класс **Session** представляет собой запись о текущей сессии пользователя. Класс относится к бэкенду сессий.

Методы класса Session:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| \_\_init\_\_(void) | Конструирует объект сессии |

Атрибуты класса Session:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| session\_key | public: int | Ключ сессии |
| profile\_id | public: int | Идентификатор профиля |
| time\_created | public: DateTime | Время создания |
| time\_expired | public: DateTime | Время, когда текущий ключ истекает |

Класс **Teacher** представляет преподавателя на стороне фронтенда. Ссылается на профиль и реализует функционал, требуемый от преподавателя.

Методы класса Teacher:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| \_\_init\_\_(void) | Конструирует объект преподавателя |
| upload\_task(string) | param: task [ string - in ]  Загружает новую задачу с описанием task |
| mark\_complete\_task(int) | param: mark [ int - in ]  Проставляет оценку mark за задание |

Атрибуты класса Teacher:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| profile\_id | public: int | Идентификатор профиля |

Класс **Student** представляет студента на стороне фронтенда. Ссылается на профиль и реализует функционал, требуемый от студента.

Методы класса Student:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| \_\_init\_\_(void) | Конструирует объект студента |
| complete\_task(string) | param: answer [ string - in ]  Загружает ответ answer на задачу |

Атрибуты класса Student:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| profile\_id | public: int | Идентификатор профиля |

Класс **Lesson** представляет собой запись об уроке на стороне фронтенда. Ассоциируется с задачей и агрегирует выполенные задачи.

Методы класса Lesson:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| \_\_init\_\_(void) | Конструирует объект урока |

Атрибуты класса Lesson:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| id | public: int | Идентификатор урока |
| task\_id | public: int | Идентификатор задачи в уроке |
| time\_created | public: DateTime | Время создания |
| time\_updated | public: DateTime | Время последнего обновления |

Класс **Task** представляет задачу. Класс относится к бэкенду задач. Содержит ссылку на преподавателя, добавившего задачу.

Методы класса Task:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| \_\_init\_\_(void) | Конструирует объект сессии |

Атрибуты класса Task:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| id | public: int | Идентификатор задачи |
| task | public: string | Текст задачи |
| teacher\_id | public: int | Идентификатор учителя, добавившего задачу |

Класс **CompleteTask** представляет ответ на задание. Класс относится к бэкенду задач. Содержит ссылку на студента, выполнившего задание.

Методы класса CompleteTask:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| \_\_init\_\_(void) | Конструирует объект ответа на задание |

Атрибуты класса CompleteTask:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| id | public: int | Идентификатор ответа на задание |
| task\_id | public: int | Идентификатор задания |
| answer | public: string | Ответ |
| student\_id | public: int | Идентификатор ответившего студента |

## Диаграммы последовательности и потоков данных

На рисунке 8 представлена диаграмма последовательности для регистрации пользователя.

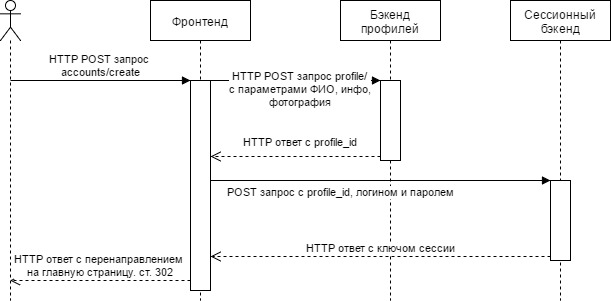


Рисунок 8. Диаграмма последовательности для регистрации пользователя

На рисунке 9 представлена диаграмма потоков данных при добавлении студентом ответа на задание.

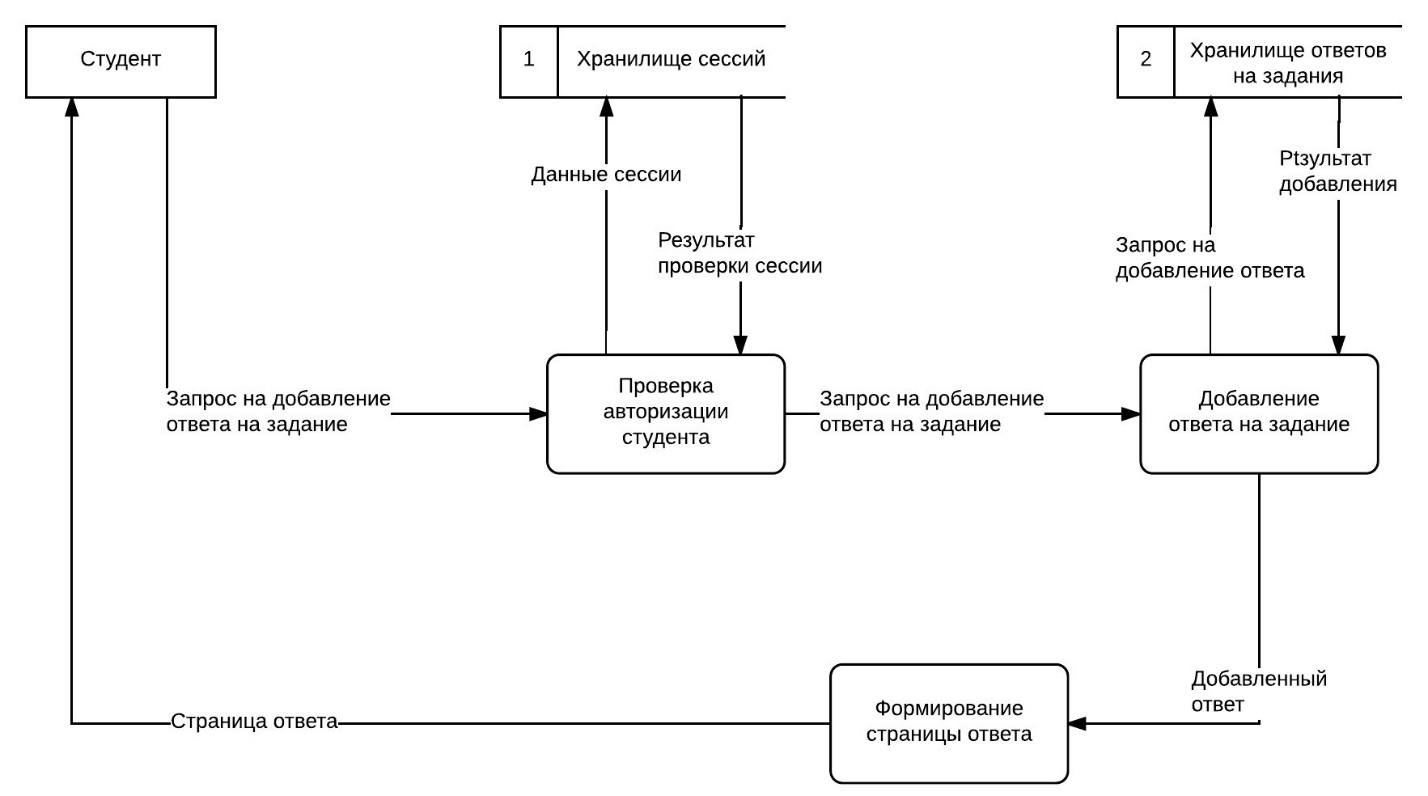


Рисунок 9. Диаграмма потоков данных при добавлении студентов ответа

# Описание пользовательского интерфейса

Пользовательский интерфейс в разрабатываемой системе представляет собой Web-интерфейс, доступ к которому осуществляется через браузер (тонкий клиент).

Страница портала состоит из «навбара» (верхней части страницы, в которой находится логотип и верхнее меню со ссылками на основные разделы портала), основной части и «футера» (нижней части страницы, в которой обычно размещают ссылки на редко посещаемые, но необходимые, страницы, например, страницы с пользовательским соглашением).

Обобщенно структуру страниц портала можно представить следующим образом:

* Главная страница
* Страница урока
  + Задание
  + Список студентов, выполнивших задание
  + Ответ студента на задание
  + Страница с информацией о студенте
* Страница с информацией о преподавателе
* Регистрация
  + Студент
  + Преподаватель
* Личный кабинет
  + Студент
  + Преподаватель

# Физический дизайн

На рисунке 10 представлена диаграмма развёртывания.

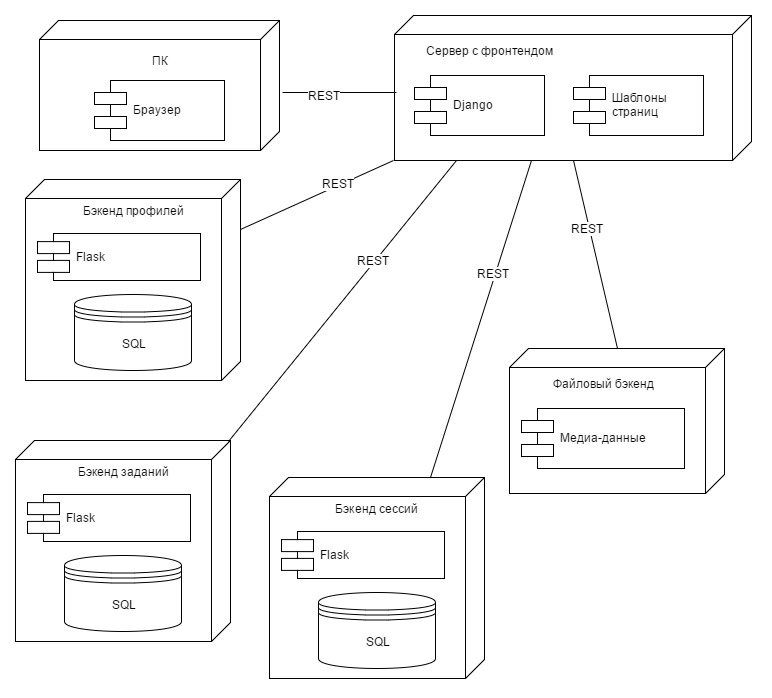


Рисунок 10. Диаграмма развёртывания системы

Исходя из того, что в техническом задании требуется, чтобы система была масштабируемой, необходимо предусмотреть возможность запуска серверов на отдельных машинах. На этапе разработки возможно использовать средства виртуализации для имитации запуска на отдельных машинах. Для этого принято решение использовать Docker, т.к. он полностью удовлетворяет следующим требованиям:

* **изоляция**: компоненты системы не должны иметь доступа друг к другу, за исключением доступа, предусмотренного протоколом взаимодействия;
* **управление ресурсами:** компонент системы должен потреблять ограниченное число ресурсов (процессорного времени, оперативной памяти, места на жестком диске);
* **отслеживание зависимостей:** компонент системы должен разворачиваться монолитно со всеми своими библиотеками. Это требование гарантирует, что компонент будет использоваться с теми же библиотеками, с которыми был протестирован;
* **низкие накладные расходы**: использование системы развертывания не должно нести высокие накладные расходы для распределенной системы.

## Выбор ОС

Согласно требованиям технического задания, разрабатываемый портал должен работать на типичных архитектурах ЭВМ (Intel x86, Intel x64), а так же быть экономически недорогим для сопровождения. Таким образом, можно сформулировать следующие требования к операционной системе:

* **Распространенность.** На рынке труда должно быть много специалистов, способных администрировать распределенную систему, работающую под управлением выбранной операционной системы.
* **Надежность**. Операционная система должна широко использоваться в стабильных проектах, таких как Yandex, Mail.ru, hh.ru. Эти компании обеспечивают высокую работоспособность своих сервисов, и на их опыт можно положиться.
* **Наличие требуемого программного обеспечения.** Выбор операционной системы не должен ограничивать разработчиков в выборе программного обеспечения, библиотек.
* **Цена.**

Под данные требования лучше всего подходит ОС XUbuntu. XUbuntu – это дистрибутив, использующий ядро Linux и графическую оболочку Xfce, которая намного менее требовательна к производительности, чем Unity, используемая в Ubuntu, о при этом не уступает в эргономичности и удобстве использования. XUbuntu поставляется с современными версиями ПО, в отличие от дистрибутивов Debian, CentOS. Преимуществом Ubuntu по сравнению с ArchLinux, Slackware и Gentoo являются низкие требования к квалификации системных администраторов.

## Выбор СУБД

В виду отсутствия жёстких требований к хранению данных целесообразно использовать легковесную встраиваемую библотеку SQLite, которая компонуется с исполняемым процессом и предоставляет удобное API для работы с БД. Данные хранятся в одном файле, что упрощает создание резервной копии.

## Выбор языка программирования и фреймворков

Язык разработки должен быть совместим с ОС Linux, поддерживать работу с SQLite, а также иметь популярные фреймворки для разработки Web-приложений. Среди таких языков можно выделить Python, Java, Go. Исходя из того, что скорость разработки данной системы важнее, чем количество запросов в секунду, которые она способна обработать, языком для разработки выбран Python.

В языке Python есть два наиболее популярных фреймворка для разработки Web-приложений: Django и Flask. Фреймворк Django целесообразно использовать для разработки сервиса, выполняющего функции фронтенда, т.к. он построен по принципу Model-View-Controller, что позволяет легко связать внутреннюю модель данных (состоящую в данном случае из данных, получаемых от бэкендов) со страницами, отображаемыми в браузере. Также фреймворк Django имеет удобный язык шаблонов, что упростит создание html-страниц.

Для разработки бэкендов целесообразнее использовать фреймворк Flask, т.к. от них требуется только хранить данные и обрабатывать HTML/REST запросы, приходящие от фронтенда.

# Список литературы