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

# Топология системы

На рис. 1 приведена топология разрабатываемой системы.

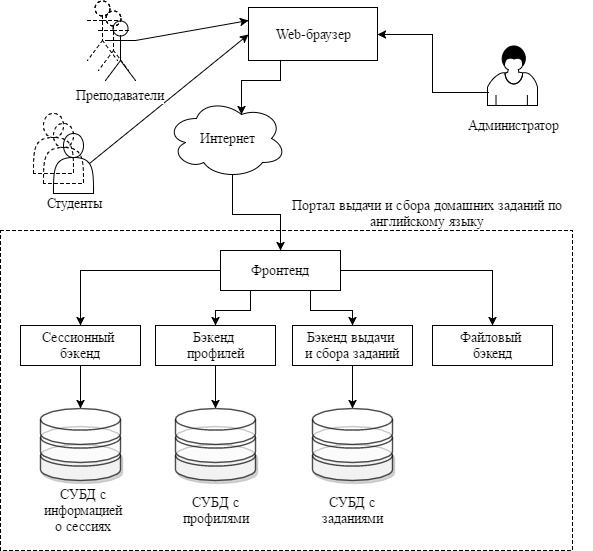


Рисунок 1. Топология системы

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 2 приведена концептуальная модель системы в нотации IDEF0.

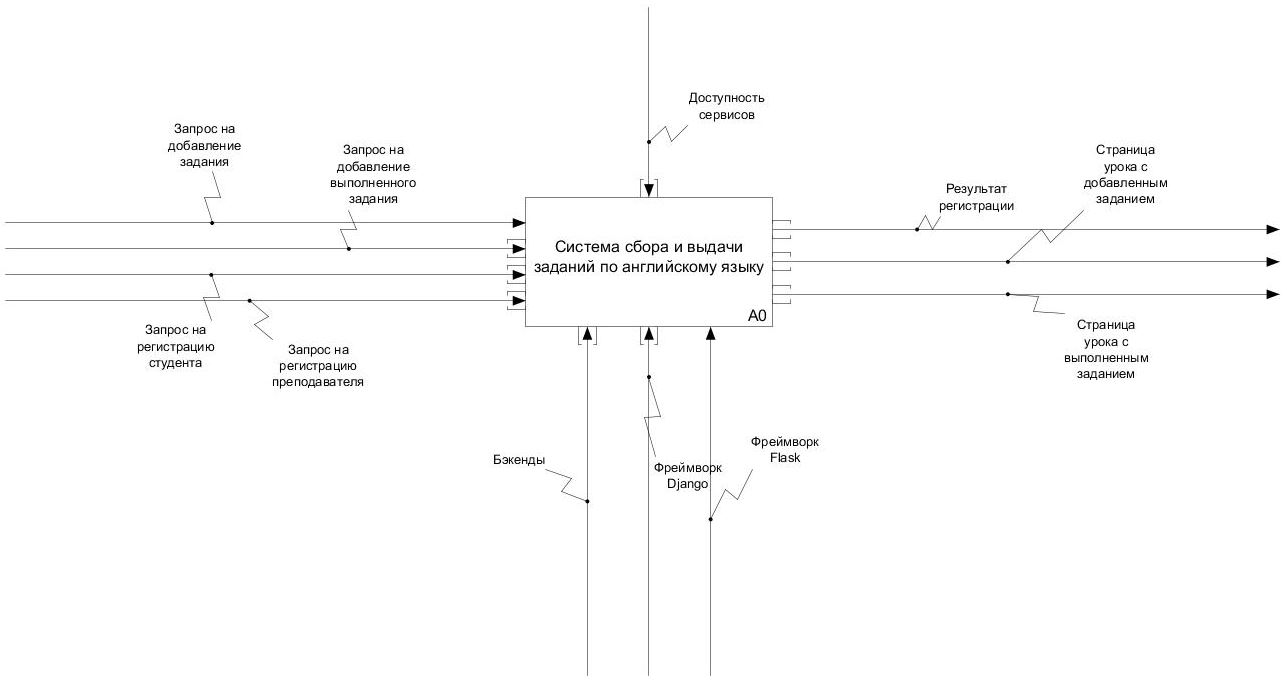


Рисунок 2. Концептуальная модель системы в нотации IDEF0

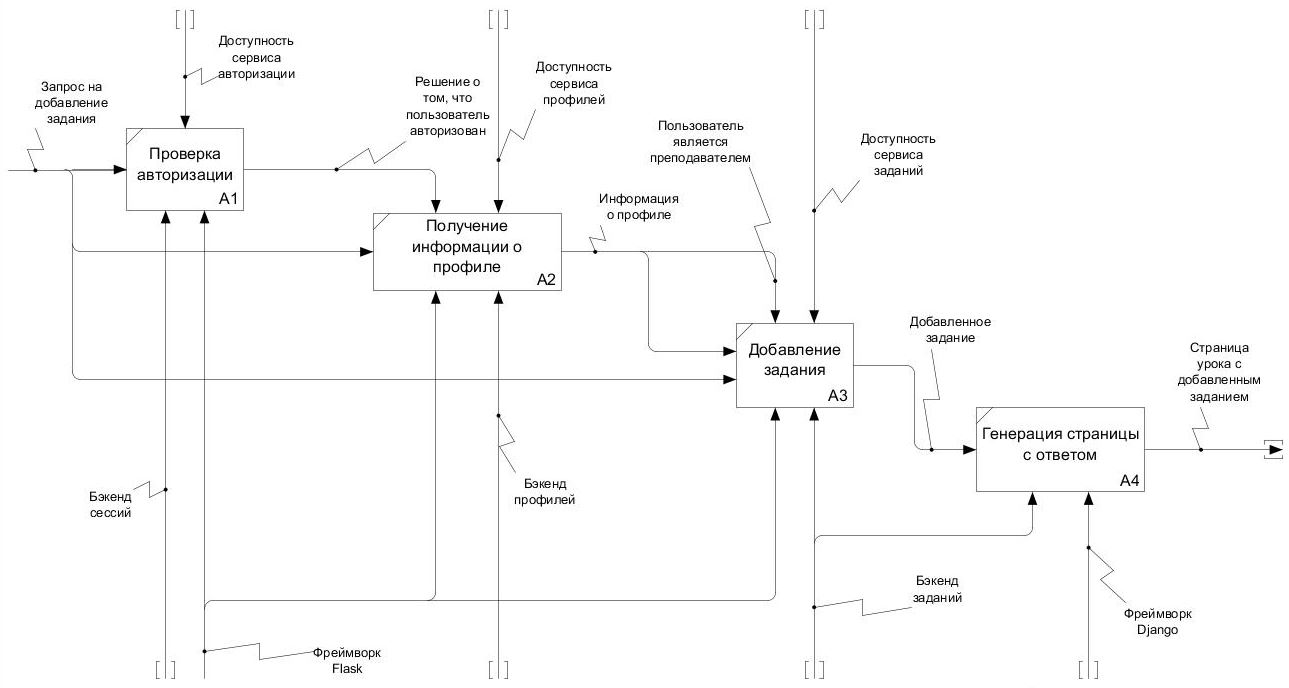
На рисунке 3 приведена детализированная концептуальная модель.

Рисунок 3. Детализированная концептуальная модель системы в нотации IDEF0.

# Варианты использования системы

На рисунке 4 представлена диаграмма прецедентов.

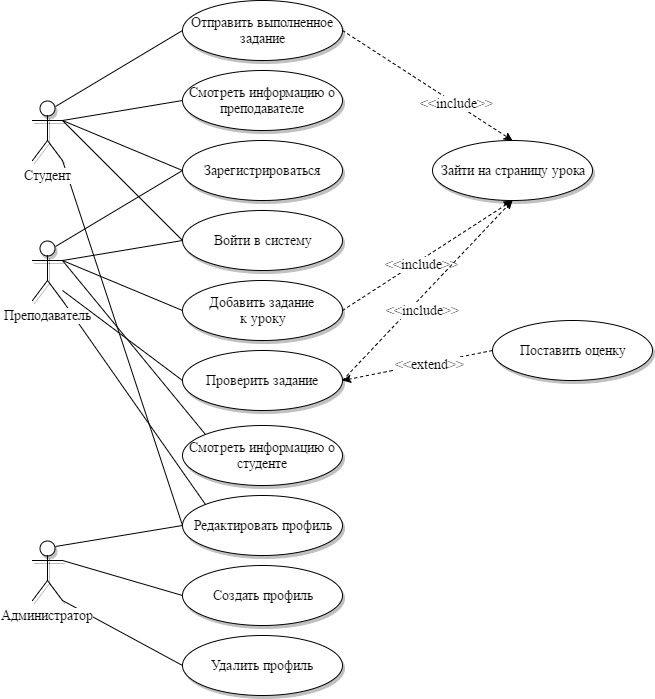


Рисунок 4. Диаграмма прецедентов

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».

# Логическая структура

На рисунке 5 представлена диаграмма классов системы.

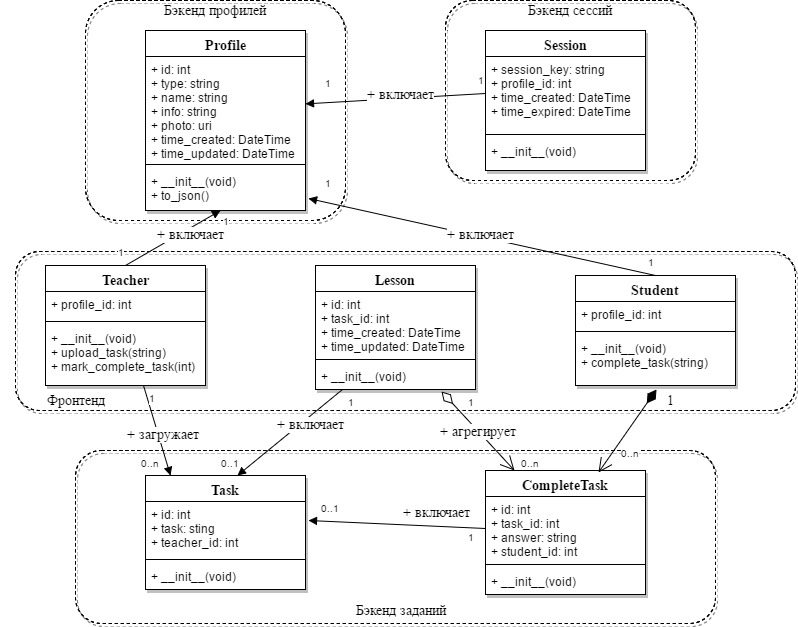


Рисунок 5. Диарамма классов

## 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 |