**Министерство образования и науки Российской федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный университет технологий и управления имени К.Г. Разумовского (Первый казачий университет)». Университетский колледж информационных технологий**

Специальность 09.02.03 «Программирование в компьютерных системах»

КУРСОВОЙ ПРОЕКТ

Модуль ПМ.01 Разработка программных модулей программного обеспечения

для компьютерных систем

МДК.01.02 Прикладное программирование

на тему Разработка информационно-образовательного портала

**Пояснительная записка**

**УКИТ 09.02.03.2016.304.06ПЗ**

|  |  |  |
| --- | --- | --- |
| Группа | П-304 |  |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (личная подпись) | Басыров С.А. |
| Руководитель проекта | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (личная подпись) | Глускер А. И. |

содержание

[ВВЕДЕНИЕ 3](#_Toc497645973)

[ОСНОВНАЯ ЧАСТЬ 5](#_Toc497645974)

[1 Введение в предметную область 5](#_Toc497645975)

[2 Спецификация 8](#_Toc497645976)

[3 Программа и методика испытания 15](#_Toc497645977)

[4 Технический проект 28](#_Toc497645978)

[5 Реализация программного изделия на языке программирования 41](#_Toc497645979)

[6 Тестирование программного продукта 43](#_Toc497645980)

[ЗАКЛЮЧЕНИЕ 45](#_Toc497645981)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc497645982)

[ПРИЛОЖЕНИЕ А Руководство оператора 48](#_Toc497645983)

[ПРИЛОЖЕНИЕ Б Исходный код 51](#_Toc497645984)

[ПРИЛОЖЕНИЕ В Протокол системы контроля версий 114](#_Toc497645985)

[ПРИЛОЖЕНИЕ Г Руководство программиста 122](#_Toc497645986)

[ПРИЛОЖЕНИЕ Д Доклад 127](#_Toc497645987)

# введение

С появлением компьютеров стало возможно автоматизировать различные сферы деятельности человека. Благодаря информационным технологиям стало возможно автоматизировать такие процессы, как сбор, обработка и передачи информации, что в свою очередь увеличивает эффективность работы.

На данный момент существует множество систем, которые предназначены для сбора, обработки и передачи информации, такие системы называются информационными.

Цель курсового проекта – разработать информационно-образовательный портал, который позволит обмениваться данными образовательной деятельности.

Основные задачи, которые были поставлены при разработке портала:

* Анализ предметной области;
* Проектирование базы данных;
* Реализация и защита базы данных в среде MySQL Workbench;
* Разработка клиентской части;
* Разработка серверной части;
* Тестирование портала.

Используемые методы при разработке портала:

1. Анализ;
2. Моделирование;
3. Сравнение;
4. Изучение дополнительной литературы;
5. Защита базы данных;
6. Объектно-ориентированный подход к реализации сереной части.

Структура курсового проекта:

1. Введение;

Данный раздел предназначен для того, чтобы описать текущее положение информационных систем в автоматизации процессов.

1. Основная часть;

Данный раздел описывает предметную область, для которой разрабатывается информационно-образовательный портал.

1. Спецификация;

Данный раздел описывает технического задание, которое устанавливает требования к информационно-образовательному порталу.

1. Программа и методика испытаний;

Данный раздел описывает применяемые методы тестирования и тестовые примеры к разрабатываемому информационно-образовательному порталу.

1. Технический проект;

Данный раздел описывает архитектурные решения, который применяются при разработке.

1. Реализация программного продукта;

Данный раздел описывает процесс разработки информационно-образовательного портала.

1. Тестирование программного продукта;

Данный раздел описывает результаты тестирования информационно-образовательного портала по программе и методике испытаний.

1. Заключение;

Данный раздел подводит итог по проделанной работе.

1. Список используемых источников;

В данном разделе приводиться список используемых источников, которые использовались при разработке информационно-образовательного портала.

1. Приложения.

В данном разделе приводятся дополнительная информация, которая включает в себя руководство оператора, исходный код приложения, протокол системы контроля версий, руководство программиста, доклад.

# основная часть

## Введение в предметную область

### Введение

Данный раздел описывает предметную область, которая представляет собой работу учебного отделения.

Главными сущностями предметной области являются:

* Студент;
* Староста;
* Преподаватель;
* Родитель;
* Заведующий отделением.

Каждый выполняет свои действия с точки зрения образовательного процесса.

**Основная деятельность студента** – это учёба. Он учится в соответствии с образовательной программой. Время от времени может происходить контроль знаний, который происходит различными методами, например, тестированием.

**Основная деятельность старосты** – это учёба и управления организационными моментами в группе, а конкретнее это контроль посещаемости, то есть староста отмечает в рапортичке, кто посещал занятие.

**Основная деятельность преподавателя** – это подача материала студентам, а так же контроль знаний. Контроль может происходить в виде тестов, по результатам которых можно оценить качество подготовки студентов и качество преподавания.

**Основная деятельность родителя** – это мониторинг результатов обучения студента и воздействие на студента с целью повышения уровня его образования. К мониторингу относятся: посещаемость, результаты его успеваемости.

**Основная деятельность заведующего учебного отделения** – это составление учебных групп, в которых будут обучаться студенты, контроль посещаемости, успеваемости и выдачи различных справок.

Другими сущностями предметной области являются:

* Предметы;
* Специальности;
* Группы;
* Расписание.

**Предмет** – учебная дисциплина, которая относится к конкретной области человеческой деятельности.

**Специальность** – группа предметов, которые необходимы в той или иной сфере деятельности. Включает в себя стандарт специальности, который в себе содержит основную информацию по обучению на данной специальности и её профессиональные предметы для подготовки.

Группа состоит из определённого числа студентов, которые учатся вместе по заданной специальности. Группы обязательно закреплены за специальностями.

**Расписание** – представляет собой график, расписывающий по дням и часам время проведения занятий в группах.

Для сбора требований к выполняемым функциям портала, были смоделированы UML-диаграммы использования для пользователей различных уровней доступа на (Рисунок 1 – 5).



Рисунок 1 – Диаграмма использования для студента



Рисунок 2 – Диаграмма прецедентов для старосты



Рисунок 3 – Диаграмма прецедентов для преподавателя



Рисунок 4 – Диаграмма прецедентов для родителя

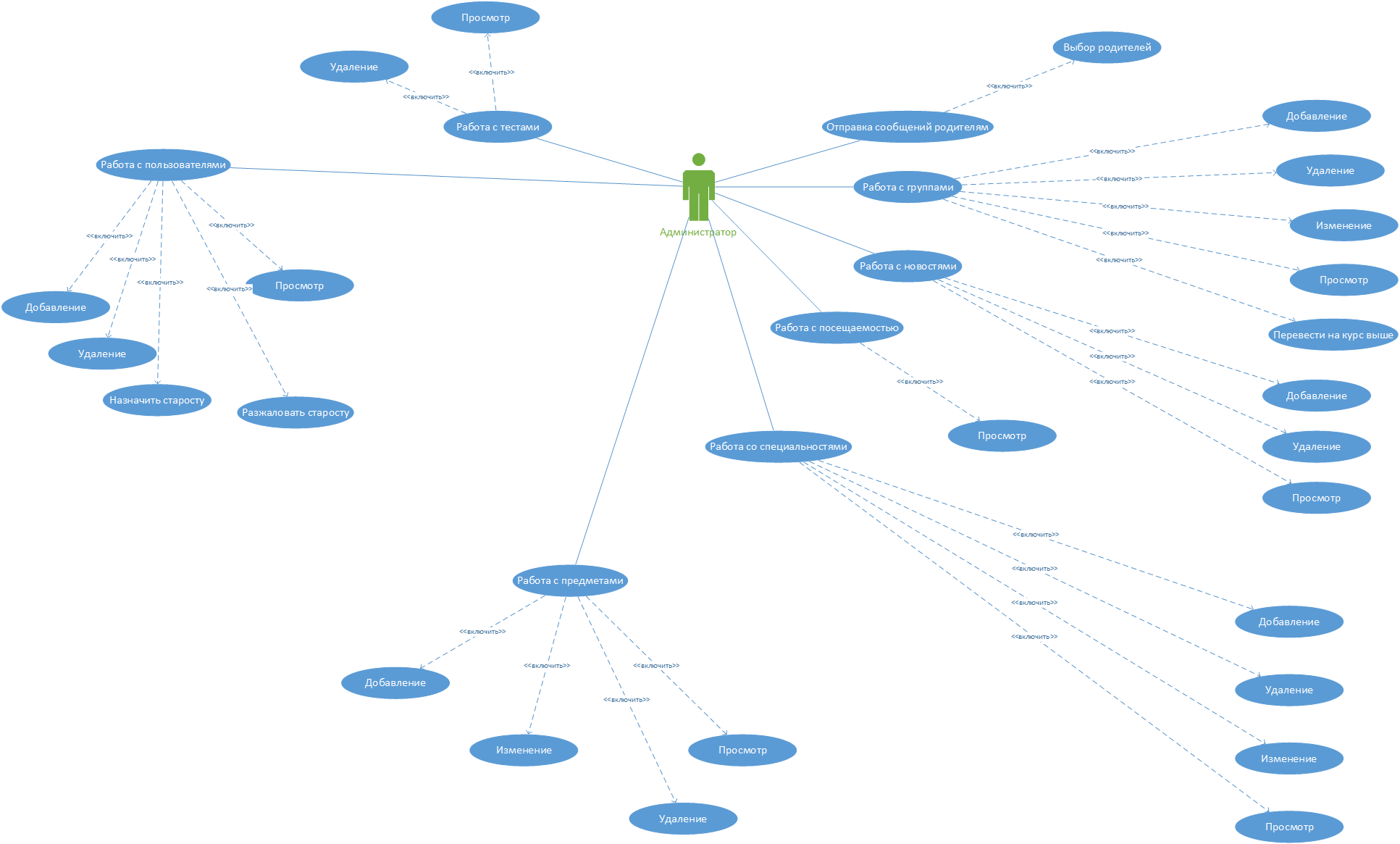


Рисунок 5 – Диаграмма прецедентов для администратора

## Спецификация

### Введение

В данном разделе описана основная информация о программном продукте, требования по эксплуатации и разработке, а также выполняемый им функционал.

### Спецификация

### Введение

### Наименование программы

Информационно образовательный портал EDUKIT

### Область применения

Образовательный процесс ФГБОУ ВО «Московский государственный университет технологий и управления имени К.Г. Разумовского (Первый казачий университет)». Университетский колледж информационных технологий.

### Объект, в котором используется программа

ФГБОУ ВО «Московский государственный университет технологий и управления имени К.Г. Разумовского (Первый казачий университет)». Университетский колледж информационных технологий.

### Основание для разработки

### Документ, на основании которого ведётся разработка

Техническое задание на курсовой проект.

### Наименование и (или) условное обозначение темы для разработки

EDUKIT.

### Назначение разработки

### Функциональное назначение

Обеспечение свободного доступа студентам, преподавателям и родителям к данным образовательной деятельности.

### Эксплуатационное назначение

Информационно-образовательный портал предназначен для использования лицами ФГБОУ ВО «Московский государственный университет технологий и управления имени К.Г. Разумовского (Первый казачий университет)». Университетский колледж информационных технологий.

### Требования к программе или программному изделию

### Требования к информационную обеспечению

### Информация о родителе

* Возраст;
* Образование;
* Место работы;
* Должность;
* Рабочий телефон;
* Сотовый телефон.

### Информация о студенте

* Группа;
* Домашний адрес;
* Сотовый телефон.

### Информация о пользователях

* Фамилия;
* Имя;
* Отчество;
* E-mail;
* Пароль;
* Тип пользователя.

### Информация о преподавателях

* Предмет/предметы;
* Информация (например, где работал ранее, достижения и т.д.)

### Новости

* Заголовок;
* Содержание (текст, списки, картинки, таблицы, ссылки);
* Автор;
* Дата публикации.

### Типы пользователей

* Администратор;
* Преподаватель;
* Староста;
* Студент;
* Родитель.

### Учебный план и график учебного плана

Представляет собой PDF-файл

### Требования к функциональным характеристикам

### Требования к системе тестирования

Тесты представляют собой вопрос и прилежащие к нему варианты ответа на него с одним правильным ответом. Вариантов ответа может быть от 4 до 10.

Со стороны преподавателя, он имеет следующе возможности:

* Просмотр всех созданных им тестов;
* Создание нового теста;
* Добавление нового вопроса в выбранный тест;
* Изменение вопросов в выбранном тесте;
* Удаление вопроса/вопросов;
* Удаление теста/тестов.

Со стороны студента, он имеет следующие возможности:

* Просмотр доступных для его группы тестов;
* Выбор теста для тестирования;
* Просмотр результатов пройденных тестов.

### Требования к управлению посещаемостью

Староста указывает кол-во всего часов в день, отведённые на пары и кол-во посещённых часов для каждого студента своей группы.

### Требования к новостной ленте

Студент:

* Просмотр.

Преподаватель:

* Добавление;
* Изменение;
* Удаление;
* Просмотр.

Администратор:

* Добавление;
* Изменение;
* Удаление;
* Просмотр.

### Требования к составу выполняемых функций

Гость (незарегистрированный пользователь):

* Регистрация/аутентификация пользователей;
* Новостная лента (просмотр);
* Информация о преподавателях (просмотр);
* Информация о расписании (просмотр).

Студент:

* Новостная лента (просмотр);
* Информация о преподавателях (просмотр);
* Информация о расписании (просмотр);
* Система тестирования (тестирование, просмотр результатов);
* Редактирование своего профиля (пароль);
* Просмотр своей посещаемости.

Преподаватель:

* Новостная лента (добавление новостей, просмотр);
* Информация о расписании (просмотр);
* Система тестирования (создание, удаление, редактирование, просмотр).

Родитель:

* Новостная лента (просмотр);
* Просмотр информации о своих детях (результаты тестирования, посещаемость).

Администратор:

* Система оповещения родителей;
* Работа с предметами (Добавление, удаление);
* Работа со специальностями (Добавление, удаление);
* Работа с группами (Добавление, удаление);
* Работа с расписанием;
* Работа с пользователями (Назначение старост, добавление, удаление);
* Просмотр посещаемости студентов.

### Требования к надёжности

### Проверка входных данных

### Условия эксплуатации

### Требования к персоналу

Минимальное кол-во персонала – один человек, владеющий русским языком и обладающий практическими навыками работы в интернет-браузере.

### Требования к составу и параметрам технических средств

В состав технических средств должен входить компьютер клиента, включающий:

* Процессор Pentium 4 или выше;
* Клавиатура и мышь;
* Видеокарта;
* Монитор;
* HDD объёмом 80 ГБ и более.

В состав технических средств должен входить сервер, включающий:

* Процессор Intel Core i3 или выше;
* Клавиатура и мышь;
* HDD объёмом 100 ГБ и более;
* Подключение к интернету не менее 100 Мб/с.

### Требования к информационной и программной совместимости

### Требования к информационным структурам на входе и выходе не предъявляются

### Требования к методам решения

Сайт должен быть разработан с использованием следующий технологий:

* HTML5;
* CSS3;
* JavaScript;
* PHP 5.6 и выше;
* Фреймворк Semantic;
* Библиотека Smarty;
* Фреймворк jQuery.
* Схема базы данных должна быть приведена к 3 нормальной форме.

### Требования к программной документации

### Состав программной документации

* Состав программной документации должен включать:
* Техническое задание;
* Пояснительная записка;
* Текст приложения;
* Программа и методика испытания;
* Руководство пользователя;
* Руководство программиста.

### Специальные требования к пояснительной записке

Специальные требования не предъявляются.

### Требования к исходным кода

* Все страницы сайта должны успешно проходить валидацию HTML5 и CSS3 на сайте W3C Validator;
* Исходные коды на JavaScript должны удовлетворят требованиям исходным кодам Google;
* Исходные коды на PHP удовлетворяющие стандарту PSR.

### Стадии и этапы разработки

### Стадии разработки

Разработка осуществляется в три стадии:

* Техническое задание;
* Технический проект;
* Рабочий проект.

### Этапы разработки

На стадии техническое задание осуществляется разработка, согласование и утверждение технического задания.

На стадии технический проект осуществляется разработка, согласование и утверждение пояснительной записки.

На стадии рабочий проект осуществляется разработка текста программы.

### Порядок контроля и приёмки

### Приёмосдаточные испытания должно проводиться в соответствии с программой и методикой испытаний, разработанной, согласованной и утверждённой не позднее 31 декабря 2016 года

### Вывод

В данном разделе были изложены все основные особенности программного продукта и требования к нему.

## Программа и методика испытания

### Введение

Данный раздел описывает основные методы тестирования для проверки соответствия программного продукта функциональным требованиям.

### Программа и методика испытания

### Объект испытания

### Наименование

Информационно-образовательный портал EDUKIT

### Область применения

Образовательный процесс ФГБОУ ВО «Московский государственный университет технологий и управления имени К.Г. Разумовского» (Первый казачий университет). Университетский колледж информационных технологий.

### Обозначение программы

EDUKIT

### Цель испытаний

Проверка соответствия программного продукта требованиям технического задания.

### Требования к программе

### Требования к функциональным характеристикам

### Требования к составу выполняемых функций

Программа должна обеспечивать выполнение следующих функций – пункт 4.2.4 Технического задания.

### Требования к информационному обеспечению

Пункт 4.1.1 Технического задания.

### Требования к надёжности

Пункт 4.3 Технического задания

### Требования к информационной и программной совместимости

Пункт 4.6 Технического задания

### Требования программной документации

### Состав программной документации

Состав программной документации должен включать:

* Техническое задание;
* Пояснительная записка;
* Текст приложения;
* Программа и методика испытания;
* Руководство пользователя;
* Руководство программиста.

Пункт 5.1 Технического задания

### Специальные требования к пояснительной записке

Специальные требования не предъявляются.

Пункт 5.2 Технического задания

### Специальные требования к исходным кодам

* Все страницы сайта должны успешно проходить валидацию HTML5 и CSS3 на сайте W3C Validator;
* Исходные коды на JavaScript должны соответствовать Google JavaScript Code Style;
* Исходные коды на PHP должны соответствовать стандарту PSR.

Пункт 5.3 Технического задания

### Средства и порядок испытания

### Технические средства, используемые при проведении испытаний

В состав технических средств должен входить IBM-совместимый компьютер, включающий:

* Процессор;
* Видеокарту;
* Клавиатуру;
* Монитор;
* Жёсткий диск.

### Программные средства, используемые при проведении испытаний

В состав программных средств должно входить:

* Операционная система семейства Windows (Начиная с Windows 7) или дистрибутив Linux;
* Браузер Google Chrome, Mozilla Firefox, Opera.

### Порядок проведения испытаний

### Подготовка к проведению испытаний заключается в обеспечении наличия компьютера в п. 5.1, и программных средств, указанных в п. 5.2, установленных на этом компьютере

### Ход проведения испытаний документируется в протоколе, где указывается перечень проводимых испытаний, результат каждого испытания и возможно замечания.

### Состав испытания

### Проверка состава программной документации в соответствии с методом, описанном в п. 6.2

### Проверка требований к программе

### Проверка обеспечений требования к программе (п. 3) в соответствии с методом, описанным в п. 6.1

### Методы испытаний

### Метод проверки требований к программе

Проверка осуществляется путём запуска программы и сравнения ожидаемых результатов привидённых в (Таблица 1).

### Метод проверки требований к составу программной документации

Проверка состава программной документации осуществляется визуально путем сравнения набора предъявленных документов (в форме распечатки или в рукописной форме) списку, приведенному в п. 4.1. При этом исходные тексты программ должны быть предоставлены так же и в электронной форме.

В случае если набор предъявленных документов соответствует списку, а исходные тексты предоставлены также в электронной форме, то в протокол заносится запись: «Состав программной документации» – соответствует; в противном случае: «Состав программной документации» – не соответствует.

### Метод проверки требований к пояснительной записке

Проверка состоит из следующих этапов:

* Проверка наличия UML диаграмм;
* Проверка наличия тестовых примеров.

В случае если все вышеприведённые проверки прошли успешно, в протокол заносится запись: «Специальные требования к пояснительной записке» – соответствует; в противном случае «Специальные требования к пояснительной записке» – не соответствует.

### Метод проверки к исходным кодам

Изложенный ниже метод применяется ко всем файлам, содержащим исходный текст, и входящим в состав программной документации по отдельности. Для каждого файла вносится в протокол запись: «Требования к исходным кодам для файла ##### – соответствует/не соответствует (где вместо #### указывается название файла).

Проверка состоит из следующих этапов:

* Проверка исходных кодов написанных на HTML, CSS через W3C Validator;
* Проверка исходных кодов написанных на PHP стандартам PSR;
* Проверка исходных кодов написанных на JavaScript стилю написания Google.

В случае, если все перечисленные этапы пройдены, то в протокол о соответствии файла требованиям, в противном случае – о несоответствии.

Таблица 1 – Тестовые примеры

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 1 | Пользователь находиться на странице сайта, как гость | Нажимает на кнопку меню «Главная» | Открывается главная страница сайта |
| 2 | Нажимает на кнопку меню «Новости» | Открывается страница с новостями |
| 3 | Нажимает на кнопку меню «Расписание» | Открывается страница с расписанием |
| 4 | Нажимает на кнопку меню «Преподаватели» | Открывается страница с информаций о преподавателях |
| 5 | Нажимает на кнопку «Регистрация» | Всплывает модальное окно с формой регистрации студента |
| 6 | Нажимает на кнопку «Вход» | Всплывает модальное окно с формой входа |
| 7 | Открыто модальное окно с формой регистрации студента | Вводит в поле «Фамилия» свою фамилию | Если данные корректны, то поле выделяется зелёной рамкой, иначе красной |
| 8 | Вводит в поле «Имя» своё имя |
| 9 | Вводит в поле «Email» свою электронную почту | Если пользователь с такой почтой не был зарегистрирован ранее, то поле выделяется зелёной рамкой, иначе красной |
| 10 | Вводит в поле «Пароль» придуманный пароль | Если придуманный пароль удовлетворяет правилам создания пароля, то поле выделяется зелёной рамкой, иначе красной |
| 11 | Вводит в поле «Повторите пароль» тот же пароль, что ввёл ранее в поле «Пароль» | Если пароли совпадают и удовлетворяют правилам создания пароля, то оба поля («Пароль» и «Повторите пароль») выделяются зелёной рамкой, иначе красной |
| 12 | Выбирает в поле «Группа» свою группу | После выбора группы, поле выделяется зелёной рамкой |
| 13 | Вводит в поле «Адрес проживания» свой фактический адрес проживания | Если данные корректны, то поле выделяется зелёной рамкой, иначе красной |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 14 | Открыто модальное окно с формой регистрации студента | Вводит в поле «Сотовый телефон» свой рабочий сотовый телефон | Если данные корректны, то поле выделяется зелёной рамкой, иначе красной |
| 15 | Нажимает на кнопку «Сбросить» | Все поля должны быть очищены от введённых ранее данных |
| 16 | Нажимает на кнопку «Зарегистрироваться» | Если все данные корректны и введены, то после добавления пользователя всплывёт уведомление о том, что регистрация прошла успешно, иначе всплывёт уведомление о том, что не все поля заполнены |
| 17 | Нажимает на ссылку «Я родитель» | Открывается страница с формой регистрации родителя |
| 18 | Открыта страница с формой регистрации родителя | Вводит в поле «Фамилия» свою фамилию | Если данные корректны, то поле выделяется зелёной рамкой, иначе красной |
| 19 | Вводит в поле «Имя» своё имя |
| 20 | Вводит в поле «Возраст» своё отчество |
| 21 | Вводит в поле «Email» свою электронную почту | Если пользователь с такой почтой не был зарегистрирован ранее, то поле выделяется зелёной рамкой, иначе красной |
| 22 | Вводит в поле «Пароль» придуманный пароль | Если придуманный пароль удовлетворяет правилам создания пароля, то поле выделяется зелёной рамкой, иначе красной |
| 23 | Вводит в поле «Повторите пароль» тот же пароль, что ввёл ранее в поле «Пароль» | Если пароли совпадают и удовлетворяют правилам создания пароля, то оба поля («Пароль» и «Повторите пароль») выделяются зелёной рамкой, иначе красной |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 24 | Открыта страница с формой регистрации родителя | Вводит в поле «Домашний телефон» свой домашний телефон | Если данные корректны, то поле выделяется зелёной рамкой, иначе красной |
| 25 | Вводит в поле «Сотовый телефон» свой сотовый телефон |
| 26 | Вводит в поле «Место работы» название той организации, в которой работает |
| 27 | Вводит в поле «Должность» свою должность |
| 28 | Выбирает в поле «Образование» своё образование | После выбора образования, поле выделяется зелёной рамкой |
| 29 | Ставит галочку в поле «Я согласен(на) на обработку персональных данных» | Флажок переходит в активное состояние |
| 30 | Ставит галочку в поле «Я подтверждаю, что выбранные дети МОИ\*» |
| 31 | Нажимает на кнопку «Назад» | Открывается главная страница |
| 32 | Нажимает на кнопку «Зарегистрироваться» | В случае, если все данные корректны, то после регистрации выводиться уведомление о том, что регистрация прошла успешно, иначе выводиться сообщение о необходимости заполнить все поля или сообщение об ошибке |
| 33 | Нажимает на панель с наименованием группы | Панель раскрывается с таблицей, в которой указаны студенты |
| 34 | Ставит галочку в поле «Выбрать» | Флажок переходит в активное состояние |
| 35 | Открыто модальное окно с формой входа | Нажимает на ссылку «Забыл пароль» | Открывается страница с восстановлением пароля |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 36 | Открыто модальное окно с формой входа | Нажимает на кнопку «Войти» | Если введённые данные корректны, то производиться аутентификация и происходит перенаправление на главную страницу сайта |
| 37 | Пользователь произвёл аутентификацию в систему | Нажимает на кнопку меню «Профиль» | Открывается страница с его личным кабинетом |
| 38 | Нажимает на кнопку меню «Выход» | Производиться выход из системы |
| 39 | Пользователь произвёл аутентификацию в систему как родитель | Нажимает на панель «Результаты тестирования» | Раскрывается панель с таблицей, в которой отображены результаты тестирования ребёнка |
| 40 | Нажимает на панель «Посещаемость» | Раскрывается панель с таблицей, в которой отображены результаты посещаемости ребёнка |
| 41 | Пользователь произвёл аутентификацию в систему как студент | Нажимает на панель «Одногруппники» | Раскрывается панель с таблицей, в которой отображена информация об одногруппниках |
| 42 | Нажимает на панель «Доступные тесты» | Раскрывается панель с таблицей, в которой отображена информация об доступных тестах |
| 43 | Нажимает на название теста в открытой панели «Доступные тесты» | Открывается страница для прохождения теста |
| 44 | Нажимает на панель «Пройденные тесты» | Раскрывается панель с таблицей, в которой отображена информация об пройденных тестах |
| 45 | Нажимает на название теста в открытой панели «Пройденные тесты» | Открывается страница с информацией об пройденном тесте |
| 46 | Открыта страница для прохождения теста | Нажимает на кнопку «Назад» | Открывается страница с личным кабинетом студента |
| 47 | Выбирает вариант ответа на вопрос | Радиокнопка переходит в активное состояние |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 48 | Открыта страница для прохождения теста | Нажимает на кнопку «Сдать» | Если результаты успешно записаны в базу данных, то всплывает уведомление о том, что результаты были записаны и происходит перенаправление на страницу личного кабинета студента |
| 49 | Открыта страница с информацией об пройденном тесте | Нажимает на кнопку меню «Профиль» | Открывается страница с личным кабинетом студента |
| 50 | Пользователь произвёл аутентификацию в систему как преподаватель | Нажимает на кнопку «+», чтобы добавить тест | Всплывает модальное окно с формой для добавления нового теста |
| 51 | Нажимает на кнопку «-», чтобы удалить выбранные тесты | Выбранные тесты удаляются и выводиться соответствующее сообщение |
| 52 | Нажимает на панель «Предметы» | Раскрывается панель с предметами, которые ведёт преподаватель |
| 53 | Нажимает на панель «Новости» | Раскрывается панель с новостями, которые опубликовать преподаватель |
| 54 | Открыто модальное окно с формой для добавления нового теста | Вводит название теста | Если введены корректные данные, то поле подсвечивается зелёной рамкой |
| 55 | Выбирает предмет | Поле подсвечивается зелёной рамкой |
| 56 | Выбирает группы, которые могут проходить тест | Флажок переходит в активное состояние |
| 57 | Нажимает кнопку «Добавить» | Если тест был успешно создан, то выводиться сообщение о том, что тест был создан, иначе выводиться сообщение об ошибке |
| 58 | Раскрыта панель «Предметы» | Нажимает на кнопку «Добавить» | Всплывает модальное окно с формой для выбора предметов |
| 59 | Нажимает на кнопку «Удалить» | Выбранные предметы снимаются с преподавателя |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 60 | Открыто модальное окно с формой для выбора предметов | Открыто модальное окно с формой для выбора предметов | Открыто модальное окно с формой для выбора предметов |
| 61 | Открыто модальное окно с формой для выбора предметов | Открыто модальное окно с формой для выбора предметов |
| 62 | Раскрыта панель «Новости» | Нажимает на кнопку «Добавить» | Всплывает модальное окно с формой добавления новой новости |
| 63 | Нажимает на кнопку «Удалить» | Выбранные новости удаляются |
| 64 | Открыто модальное окно с формой для добавления новой новости | Вводит заголовок новости | Если введены корректные данные, то поле подсвечивается зелёной рамкой |
| 65 | Вводит содержание новости | - |
| 66 | Выбирает дату публикации | - |
| 67 | Нажимает кнопку «Опубликовать» | Если новость была успешно добавлена, то выводиться сообщение о том, что новость добавлена, иначе выводиться сообщение об ошибке |
| 68 | Открыта страница с формой входа в панель администратора системы | Нажимает на кнопку «Назад» | Открывается главная страница сайта |
| 69 | Нажимает на кнопку «Войти» | Если данные верны и пользователь существует, то происходит вход в панель управления, иначе выводиться сообщение об ошибке |
| 70 | Администратор вошёл в панель администратора | Нажимает на кнопку меню «Главная» | Открывается главная страница панели администратора |
| 71 | Нажимает на кнопку меню «Пользователи» | Открывается страница для работы с пользователями |
| 72 | Нажимает на кнопку меню «Группы» | Открывается страница для работы с группами |
| 73 | Нажимает на кнопку меню «Специальности» | Открывается страница для работы со специальностями |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 74 | Администратор вошёл в панель администратора | Нажимает на кнопку меню «Новости» | Открывается страница для работы с новостями |
| 75 | Нажимает на кнопку меню «Предметы» | Открывается страница для работы с предметами |
| 76 | Нажимает на кнопку меню «Расписание» | Открывается страница для работы с расписанием |
| 77 | Нажимает на кнопку меню «Оповещения» | Открывается страница для работы с оповещениями |
| 78 | Нажимает на кнопку меню «Посещаемость» | Открывается страница для работы с посещаемостью |
| 79 | Нажимает на кнопку меню «Настройки» | Открывается страница с настройками системы |
| 80 | Нажимает на кнопку меню «Выход» | Администратор выходит из панели администратора |
| 81 | Открыта страница для работы с пользователями | Нажимает на вкладку «Преподаватели» | Отображается таблица с зарегистрированными преподавателями |
| 82 | Нажимает на вкладку «Студенты» | Отображаются панели с названиями групп |
| 83 | Нажимает на панель с группой | Раскрывается панель с таблицей, в которой содержится информация о студентах |
| 84 | Нажимает на вкладку «Родители» | Отображается таблица с зарегистрированными родителями |
| 85 | Нажимает на вкладку «Старосты» | Отображается форма для назначения и разжалования старосты |
| 86 | Нажимает на кнопку «Назначить» | Назначается выбранный староста |
| 87 | Нажимает на кнопку «Разжаловать» | Происходит разжалование выбранного старосты |
| 88 | Нажимает на кнопку «Удалить» | Происходит удаление выбранного пользователя из системы |
| 89 | Нажимает на кнопку «Добавить» | Происходит добавление нового преподавателя с указанными данными |
| 90 | Открыта страница для работы с группами | Нажимает на кнопку «Удалить» | Удаляются выбранные группы |
| 91 | Нажимает на кнопку «Добавить» | Добавляет группу с указанными данными |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 92 | Открыта страница для работы со специальностями | Нажимает на кнопку «Удалить» | Происходит удаление выбранных специальностей |
| 93 | Нажимает на кнопку «Добавить» | Добавляет специальность с указанными данными |
| 94 | Открыта страница для работы с новостями | Нажимает на заголовок новости | В форму для добавления новости подставляются данные выбранной формы |
| 95 | Нажимает на кнопку «Изменить» | Изменяет выбранную ранее новость, нажатием на заголовок |
| 96 | Нажимает на кнопку «Удалить» | Удаляются выбранные новости |
| 97 | Нажимает на кнопку «Добавить» | Добавляется новая новость с указанными данными |
| 98 | Открыта страница для работы с предметами | Нажимает на кнопку «Удалить» | Удаляются выбранные новости |
| 99 | Нажимает на кнопку «Добавить» | Добавляется новый предмет |
| 100 | Открыта страница для работы с расписанием | Нажимает на вкладку «Основное» | Отображаются панели с группами, в которых содержатся таблицы с расписанием |
| 101 | Нажимает на вкладку «Изменения» | Отображаются панели с группами, в которых содержатся таблицы с изменениями в расписании |
| 102 | Нажимает на кнопку «Назначить» | Назначает пару выбранной группе |
| 103 | Нажимает на кнопку «Поставить изменения» | Назначает изменения в расписании в выбранной группе |
| 104 | Нажимает на кнопку «Удалить все изменения» | Удаляются все изменения в расписании в выбранной группе |
| 105 | Открыта страница для работы с оповещениями | Жмёт кнопку «Отправить» | Выбранным пользователям отправляется сообщение |
| 106 | Открыта страница с посещаемостью | Нажимает на панель с названием группы | Раскрывается панель с таблицей студентов |
| 107 | Нажимает на кнопку «Выбрать» | Отображается посещаемость выбранного студента |

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 108 | Открыта страница с настройками системы | Нажимает на кнопку «Удалить» | Удаляются выбранные администраторы |
| 109 | Нажимает на кнопку «Добавить» | Добавляется новый администратор |

### Вывод

В этом разделе были изложены методы, которые будут применяться к порталу для проверки соответствия требованиям технического задания.

## Технический проект

### Введение

В данном разделе изложены архитектурные решения и алгоритмы, используемые в разработке информационно-образовательного портала.

### Технический проект

### Введение

### Наименование программы

Информационно-образовательный портал EDUKIT.

### Условное обозначение темы разработки

EDUKIT.

### Документ, на основании которого ведётся разработка

Техническое задание на курсовой проект.

### Назначение и область применения

### Назначение программы

Обеспечение свободного доступа студентам, преподавателям и родителям к данным образовательной деятельности.

### Краткая характеристика области применения

Информационно-образовательный портал предназначен для использования лицами ФГБОУ ВО «Московский государственный университет технологий и управления имени К.Г. Разумовского» (Первый казачий университет). Университетский колледж информационных технологий для частичной автоматизации отделений колледжа, а так же предоставлении преподавателям возможности создания тестов для оценки знаний студентов.

### Технические характеристики

### Постановка задачи на разработку

Создание информационно-образовательного портала для колледжа, который позволит студентам, преподавателям и родителям обмениваться данными образовательной деятельности.

Обмен данными образовательной деятельности подразумевают под собой:

* Система тестирования;
* Система контроля посещаемости;
* Система управления расписанием;
* Система оповещения студентов и родителей по электронной почте.

### Аппаратные средства

В качестве сервера будет выступать сервер с операционной системой Windows Server 2008 R2, обладающий следующими техническими характеристиками:

* Процессор: Intel Core i5-4590 3.30GHz;
* ОЗУ: 8GB;
* HDD: 1TB;
* Ethernet 100 MB/s.

### Программные средства

В качестве локального сервера выступает OpenServer. Данная сборка была выбрана потому что в ней имеются все необходимые программные средства для функционирования портала:

* Веб-сервер Apache:

Является самым распространённым веб-сервером и был выбран по ряду следующих особенностей:

1. Кроссплатформенный;
2. Распространяется под свободной лицензией.

* PHP 7:

Данный язык был выбран основным для реализации серверной части портала. PHP является самым распространённым языком в веб-разработке при написании сложных веб-сайтов.

* MySQL:

Является бесплатной реляционной СУБД, что является главной её особенностью. Так же MySQL был выбран потому что удовлетворяет следующим характеристикам:

1. Максимальный размер базы данных зависит от максимального объёма жёсткого диска;
2. Максимальный размер таблицы: 256ТБ;
3. Максимальный размер строки: 65536 байт;
4. Максимальный размер поля: 4ГБ;
5. Максимальный кол-во строк: 18,446,744,073,709,551,615.

Для СУБД используется MySQL Workbench, так как он является бесплатным.

### Описание используемых технологий

Клиентская часть разрабатывается на языках HTML5, CSS3 и JavaScript с использованием следующих фреймворков и библиотек:

* Фреймворк Semantic UI:

Semantic UI обладает следующими особенностями:

1. Разметка, используемая для вёрстки проста в сопровождении;
2. Поддерживает разметку для адаптивного дизайна;
3. Поддерживает различные темы оформления.

* Библиотека jQuery:

jQuery является самой популярной библиотекой для JavaScript, так как позволяет сократить стандартный код на JavaScript и помимо этого обладает следующими особенностями:

1. DOM манипуляции;
2. Обработка событий;
3. Групповые операции над селекторами;
4. Дополнительные функции для работы с коллекциями.

Серверная часть разрабатывается на языке PHP с использованием следующих библиотек:

* Шаблонизатор Smarty:

Был выбран по причине того, что из существующих аналогов, его изучение оказалось самым быстрым.

* Библиотека PDO для взаимодействия с БД, в частности с MySQL;

PDO – это библиотека, которая обеспечивает интерфейс доступа к базам данных. Особенности PDO:

1. Объекто-ориентированный подход в работе с библиотекой;
2. Поддержка других баз данных, что позволит легко переходить на другие базы данных.

* Собственный класс CForm для получения данных форм;
* Собственный класс CTools для редиректа и вывода отладочных сообщений.

### Описание применяемых методов

Информационно-образовательный портал разработан с использованием шаблона проектирования MVC (Model View Controller). Так как MVC является самым распространённым архитектурным решением при разработке веб-сайтов.

В качестве Model выступают классы базы данных, в которых реализована логика работы с таблицами.

В качестве View выступают шаблоны с расширением, которые используются шаблонизатором для выдачи пользователю конечной страницы.

В качестве Controller выступают основные страницы, которые обрабатывают события пользователя. На таких страницах пользователь может взаимодействовать с сайтом, например, регистрация или аутентификация.

Схема работы MVC отображена на (Рисунок 6).



Рисунок 6 - Схема MVC

Вся бизнес-логика (запросы для манипуляции с данными в базе данных) были выполнены в виде хранимых процедур на стороне сервера.

При программировании серверной части используется объектно-ориентированная парадигма, так как она является наиболее популярной и уже устоявшейся парадигмой. ООП позволяет моделировать практически любую предметную области в виде совокупности объектов и их взаимодействии, что упрощает понимание работы программы.

В связи с этим были разработаны классы, которые описывают собой предметную область информационно-образовательного портала. Ниже на (Рисунок 7, 8 и 9) приведены диаграммы классов.

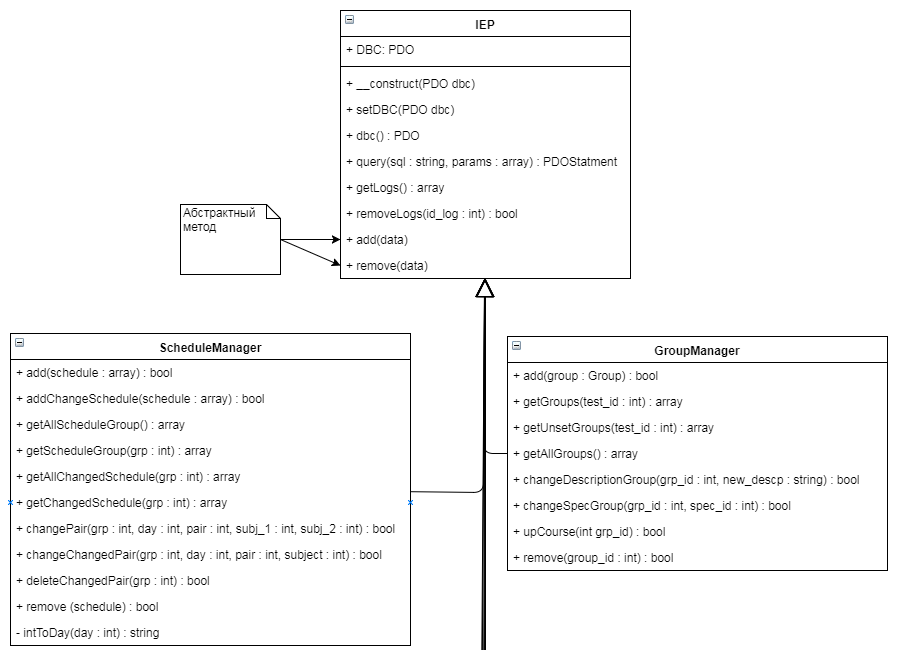


Рисунок 7 – Менеджеры 1

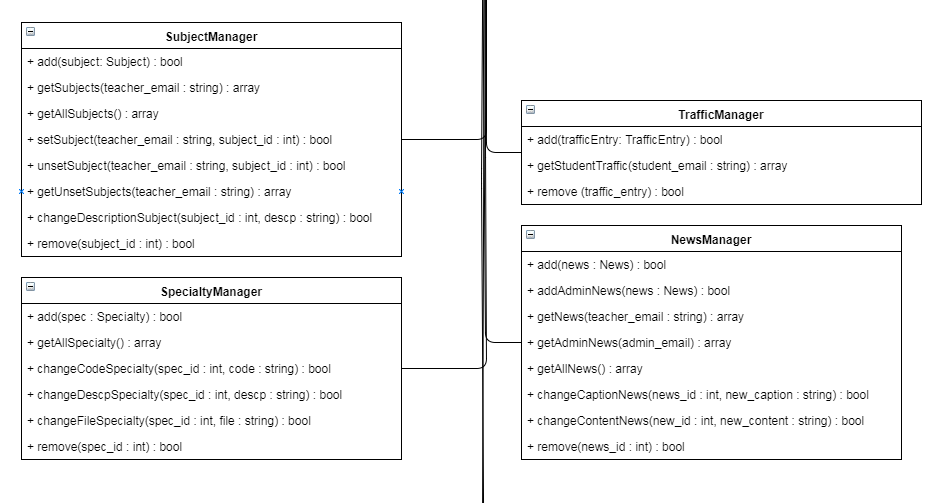


Рисунок 8 – Менеджеры 2

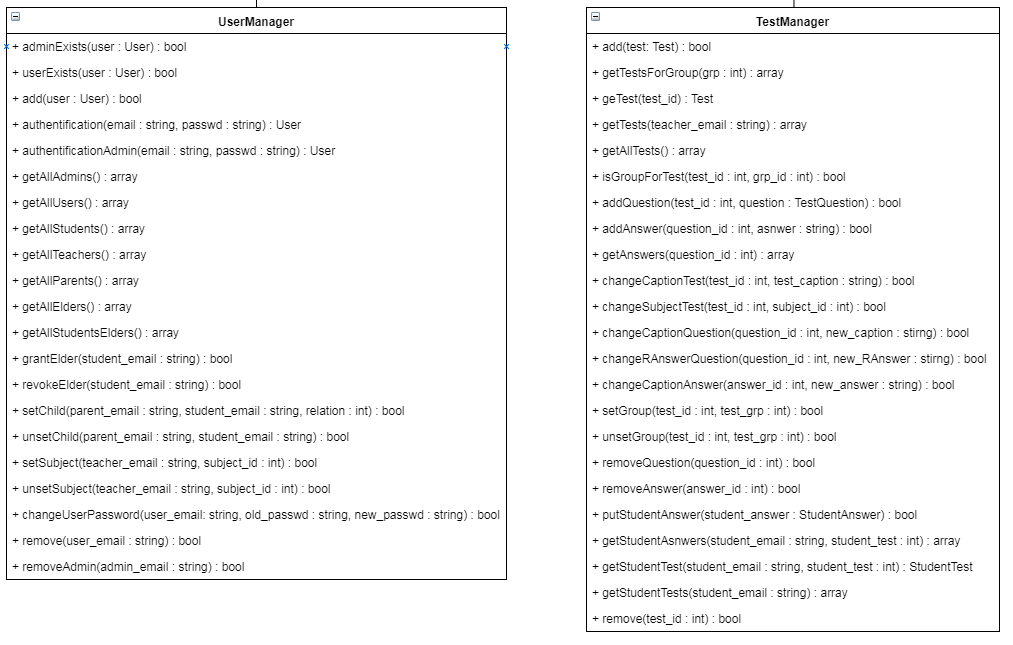


Рисунок 9 – Менеджеры 3

Ниже в (Таблица 2) приведено описание менеджеров.

Таблица 2 – Краткое описание менеджеров

|  |  |
| --- | --- |
| **Название менеджера** | **Назначение** |
| IEP | Базовый класс, который содержит общие методы для всех дочерних менеджеров |
| ScheduleManager | Работа с расписанием |
| GroupManager | Работа с группами |
| SubjectManager | Работа с предметами |
| TrafficManager | Работа с посещамостью |
| SpecialtyManager | Работа со специальностями |
| NewsManager | Работа с новостями |
| UserManager | Работа с пользователями |
| TestManager | Работа с тестами |

В СУБД были созданы следующие таблицы, которые описывают предметную область. Описание таблиц приведено в (Таблица 3 - 26).

Таблица 3 – Admins

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_admin | Идентификатор | int(11) | AI, PK |
| sn | Фамилия | varchar(30) |  |
| fn | Имя | varchar(30) |  |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| pt | Отчество | varchar(30) |  |
| email | Почта | varchar(30) | UNIQUE |
| passwd | Пароль | varchar(32) |  |

Таблица 4 – Admin-News

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_news | Идентификатор | int(11) | AI, PK |
| caption | Заголовок | varchar(255) |  |
| content | Содержание | text |  |
| id\_author | Идентификатор автора | int(11) | FK |
| date\_publication | Дата публикации | date |  |

Таблица 5 – Answers

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_answer | Идентификатор ответа | int(11) | AI, PK |
| id\_question | Идентификатор вопроса | int(11) | FK |
| answer | Ответ | varchar(255) |  |

Таблица 6 – Groups

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| grp | Идентификатор группы | int(11) | AI, PK |
| description | Название | varchar(10) |  |
| edu\_year | Год обучения | varchar(10) |  |
| spec\_id | Идентификатор специальности | int(11) | FK |
| is\_budget | Тип группы | int(11) |  |

Таблица 7 – Group-tests

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_test | Идентификатор теста | int(11) | FK |
| id\_group | Идентификатор группы | int(11) | FK |

Таблица 8 – News

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_news | Идентификатор новости | int(11) | AI, PK |
| caption | Заголовок | varchar(255) |  |
| content | Содержание | text |  |
| id\_author | Идентификатор автора | int(11) | FK |
| date\_publication | Дата публикации | date |  |

Таблица 9 – Parent-Child

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_parent | Идентификатор родителя | int(11) | FK |
| id\_children | Идентификатор студента | int(11) | FK |
| id\_type\_relation | Идентификатор отношения | int(11) | FK |

Таблица 10 – Parents

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_parent | Идентификатор родителя | int(11) | FK |
| age | Возраст | int(11) |  |
| education | Образование | varchar(50) |  |
| work\_place | Место работы | varchar(255) |  |
| post | Должность | varchar(255) |  |
| home\_phone | Домашний телефон | varchar(30) |  |
| cell\_phone | Сотовый телефон | varchar(30) |  |

Таблица 11 – Questions

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_question | Идентификатор вопроса | int(11) | AI, PK |
| id\_test | Идентификатор теста | int(11) | FK |
| question | Вопрос | varchar(255) |  |
| r\_answer | Правильный ответ | varchar(255) |  |

Таблица 12 – Relations

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_relation | Идентификатор отношения | int(11) | AI, PK |
| description | Наименование | varchar(255) |  |

Таблица 13 – Specialty

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_spec | Идентификатор специальности | int(11) | AI, PK |
| code\_spec | Код специальности | varchar(10) |  |
| description | Название | varchar(255) |  |
| pdf\_file | Файл специальности | varchar(255) |  |

Таблица 14 – Student-Answers

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_student\_answer | Идентификатор ответа студента | int(11) | AI, PK |
| id\_student\_test | Тест | int(11) | FK |
| question | Вопрос | varchar(255) |  |
| answer | Ответ | varchar(255) |  |

Таблица 15 – Students-Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_student\_test | Идентификатор пройденного теста | int(11) | AI, PK |
| id\_student | Идентификатор студента | int(11) | FK |
| caption | Заголовок | varchar(255) |  |
| subject | Предмет | subject(255) |  |
| date\_pass | Дата сдачи | date |  |
| mark | Оценка | int(11) |  |

Таблица 16 – Student-Traffic

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_traffic | Идентификатор | int(11) | AI, PK |
| id\_student | Идентификатор студента | int(11) | FK |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| date\_visit | Дата посещения | date |  |
| count\_passed\_hours | Кол-во пропущенных пар | int(11) |  |
| count\_all\_hours | Всего пар | int(11) |  |

Таблица 17 – Students

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_student | Идентификатор студента | int(11) | FK |
| home\_address | Домашний адрес | varchar(255) |  |
| cell\_phone | Сотовый телефон | varchar(12) |  |
| grp | Группа | int(11) | FK |

Таблица 18 – Subjects

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_subject | Идентификатор предмета | int(11) | AI, PK |
| description | Название | varchar(255) |  |

Таблица 19 – Teacher-Subjects

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_teacher | Идентификатор преподавателя | int(11) | FK |
| id\_subject | Идентификатор предмета | int(11) | FK |

Таблица 20 – Teachers

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_teacher | Идентификатор преподавателя | int(11) | AI, PK |
| info | Информация о преподавателе | text |  |

Таблица 21 - Tests

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_test | Идентификатор теста | int(11) | AI, PK |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_subject | Идентификатор предмета | int(11) | FK |
| id\_teacher | Идентификатор преподавателя | int(11) | FK |
| caption | Название | varchar(255) |  |

Таблица 22 – TypeUsers

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_type\_user | Идентификатор типа пользователя | int(11) | AI, PK |
| description | Наименование | varchar(30) |  |

Таблица 23 – Users

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_user | Идентификатор пользователя | int(11) | AI, PK |
| sn | Фамилия | varchar(30) |  |
| fn | Имя | varchar(30) |  |
| pt | Отчество | varchar(30) |  |
| email | Почта | varchar(30) |  |
| passwd | Пароль | varchar(32) |  |
| id\_type\_user | Идентификатор типа пользователя | int(11) | FK |

Таблица 24 – Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_grp | Идентификатор группы | int(11) | FK |
| \_day | День | int(11) | PK |
| pair | Пара | int(11) | PK |
| subj\_1 | Предмет на нижней неделе | int(11) | FK |
| subj\_2 | Предмет на верхней неделе | int(11) | FK |

Таблица 25 – Changed-Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_grp | Идентификатор группы | int(11) | FK |
| \_day | День | datetime | PK |
| pair | Пара по счёту | int(11) | PK |
| subject | Идентификатор предмета | int(11) | FK |

Таблица 26 – Logs

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id\_log | Идентификатор лога | int(11) | AI, PK |
| tbl | Таблица, в которой произошло что-то | varchar(225) |  |
| msg | Сообщение | text |  |
| date | Дата | date |  |

### Описание алгоритмов

Вычисление оценки пройденного теста:

|  |  |
| --- | --- |
|  | (1) |

Затем оценка получается в зависимости от полученного процента, который попадает в один из интервалов приведённых в (Таблица 27).

Таблица 27 – Интервалы

|  |  |
| --- | --- |
| **Процент** | **Оценка** |
| от 100 до 75 | 5 |
| от 75 до 50 | 4 |
| от 50 до 25 | 3 |
| от 25 до 0 | 2 |

### Источники, использованные при разработке

1 MySQL Documentation [Электронный ресурс]. – Режим доступа: https://dev.mysql.com/doc/;

2 PHP: Hypertext Preprocessor [Электронный ресурс]. – Режим доступа: <http://php.net/>;

3 W3 Markup Validation Service [Электронный ресурс]. – Режим доступа: https://validator.w3.org/.

### Вывод

Раздел описывает проектировочную часть разрабатываемого информационно-образовательного портала.

## Реализация программного изделия на языке программирования

### Введение

В этом разделе описывается разработка информационно-образовательного портала.

### Процесс реализации

Разработка портала началась в январе 2016 года. После реализации основных функций, т.е. системы тестирования, разработка портала временно прекратилась и портал до сих пор не был выпущен в релиз. Затем данный портал был взят как курсовой проект и ТЗ было составлено в августе 2016 года.

Срок сдачи курсового проекта был 31.12.2016, однако он не был сдан в силу того, что время разработки было очень растянуто и не продуктивно. В следствии этого могла пострадать архитектура приложения и появиться множество ошибок.

Сначала были свёрстаны основные страницы:

1. Новостей;
2. Расписания;
3. Преподавателей;
4. Личных кабинетов:
   1. Студента;
   2. Старосты;
   3. Преподавателя;
   4. Родителя.
5. Панели администратора (с разделами).
   1. Пользователи;
   2. Группы;
   3. Специальности;
   4. Новости;
   5. Предметы;
   6. Расписание;
   7. Оповещения;
   8. Посещаемость;
   9. Настройки.

После вёрстки основных страниц, была начата работа над разработкой основных объектов базы данных, в частности таблиц, ограничений, представлений, хранимых процедур, триггеров и пользователей.

Ниже приведена (Таблица 28) со статистикой по объектам базы данных.

Таблица 28 – Кол-во объектов базы дангых

|  |  |
| --- | --- |
| **Количество** | **Значение** |
| Таблиц | 24 |
| Столбцов | 189 |
| Индексов | 53 |
| Триггеров | 33 |
| Представлений | 13 |
| Хранимых процедур | 111 |
| Функций | 13 |
| Пользователей | 1 |

Затем были разработаны основные классы. Классы были сгруппированы по разным пространствам имён. На (Рисунок 10) отображена структура классов и их краткое описание.



Рисунок 10 – Структура классов

Из-за длительно разработки, интерес стал угасать и внедрять какие-либо новые архитектурные решения уже не хотелось. Рефакторинга кода не производилось и тестирование кода путём написания unit-тестов так же не выполнялось.

### Вывод

В этом разделе был описан процесс разработки информационно-образовательного портала.

## Тестирование программного продукта

### Введение

Тестирование исходного кода путём написания юнит-тестов не производилось.

Сдача информационного-портала происходит путём приёмо-сдаточных испытаний.

### Протоколы прохождения тестов

Все тестовые примеры были пройдены успешно. Ниже на (Рисунок 11 – 18) приведены результаты некоторых тестов.

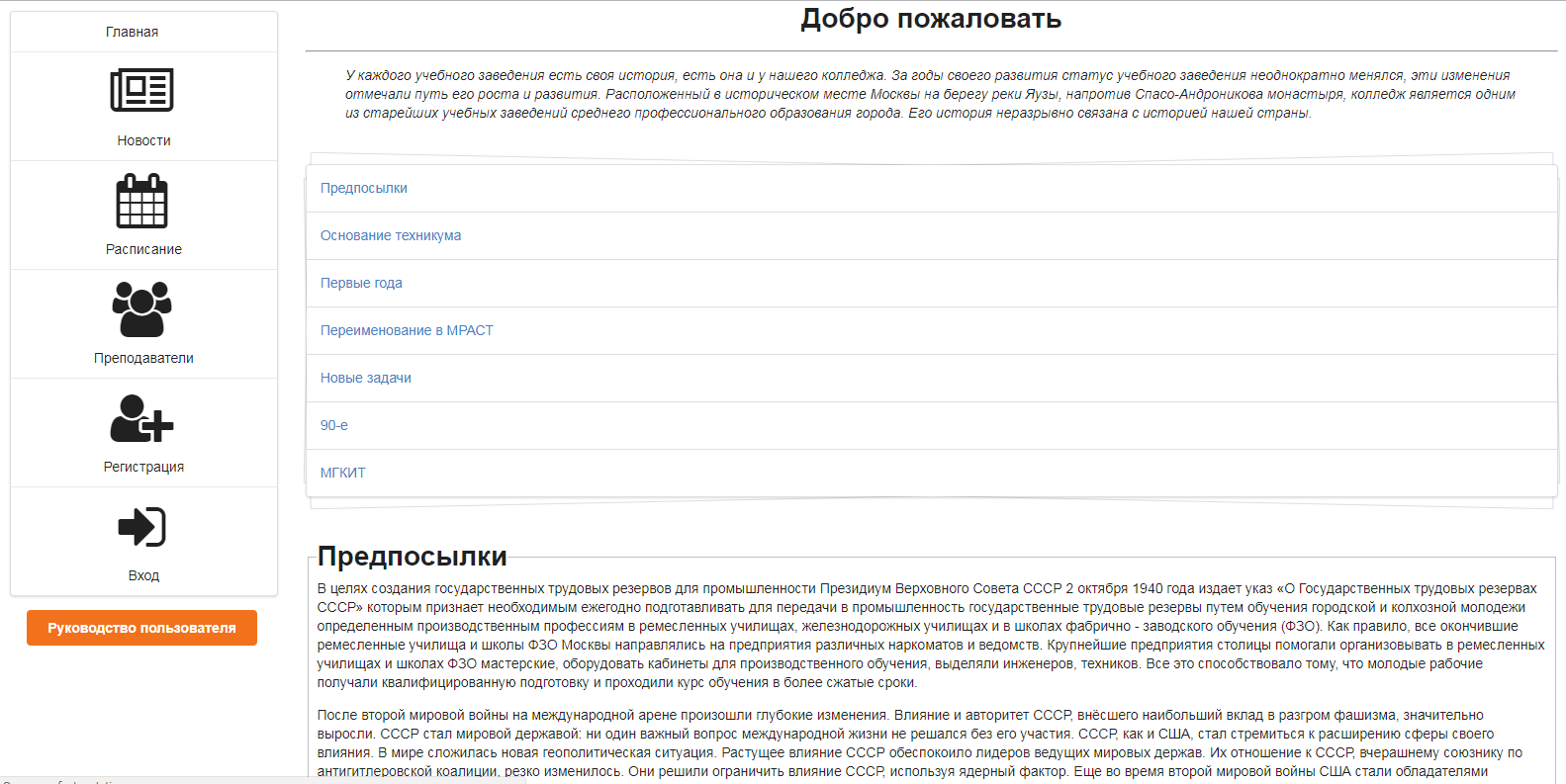


Рисунок 11 - Пользователь находиться на странице сайта, как гость

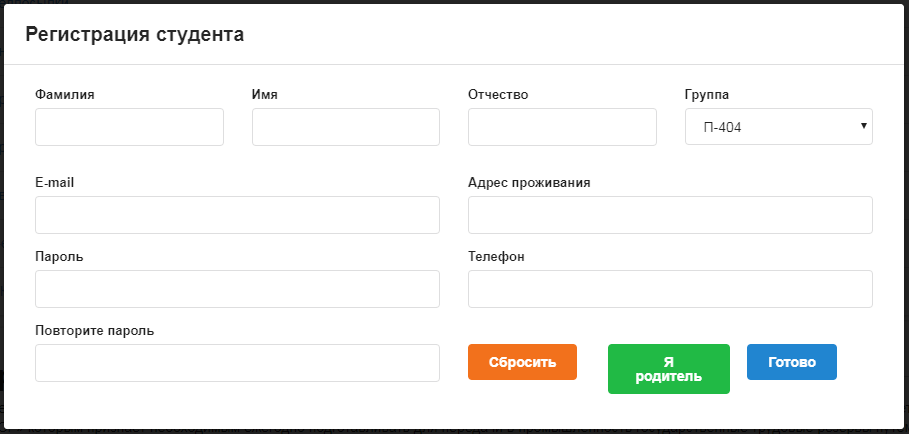


Рисунок 12 - Открыта страница с формой регистрации студента

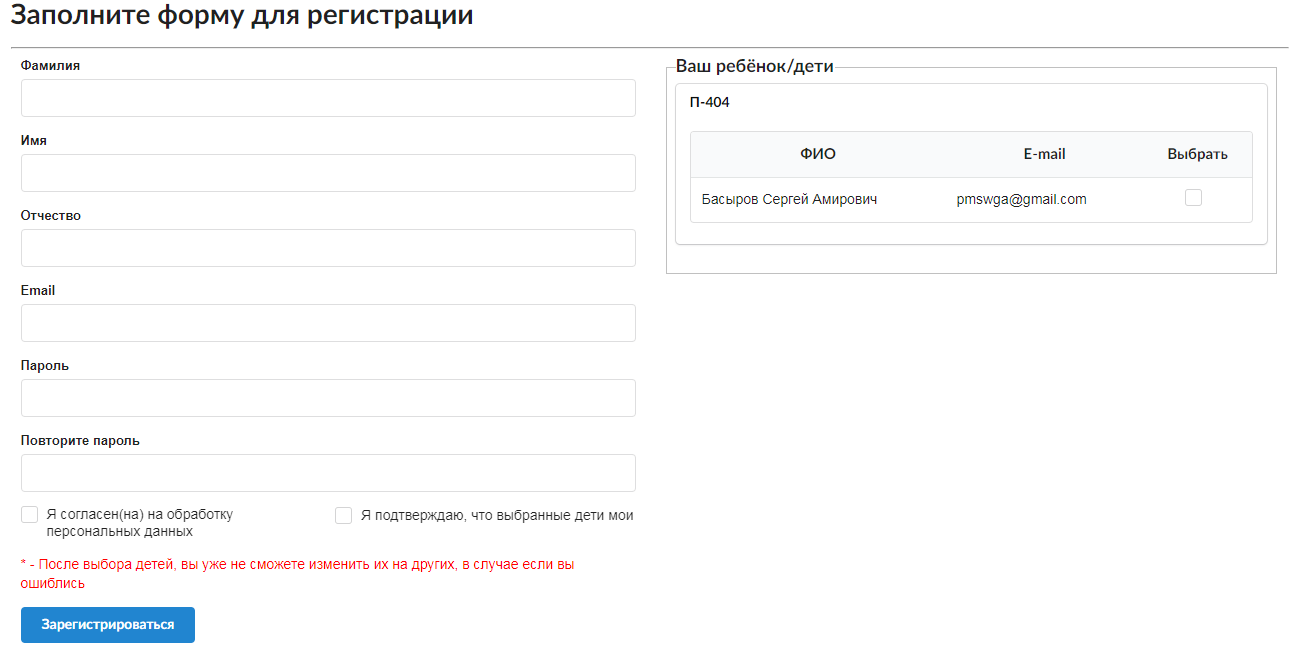


Рисунок 13 – Открыта страница с формой регистрации родителя

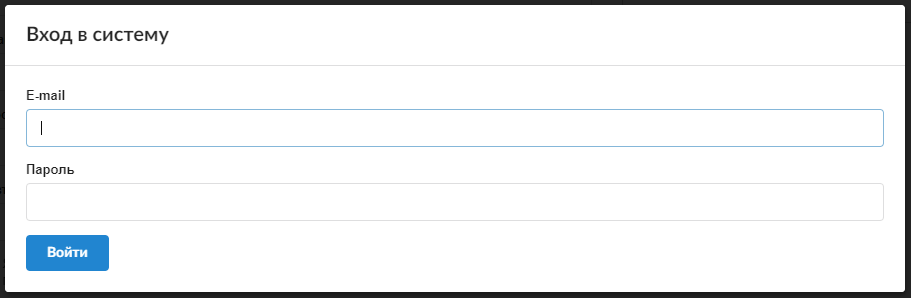


Рисунок 14 – Открыто модальное окно с формой входа

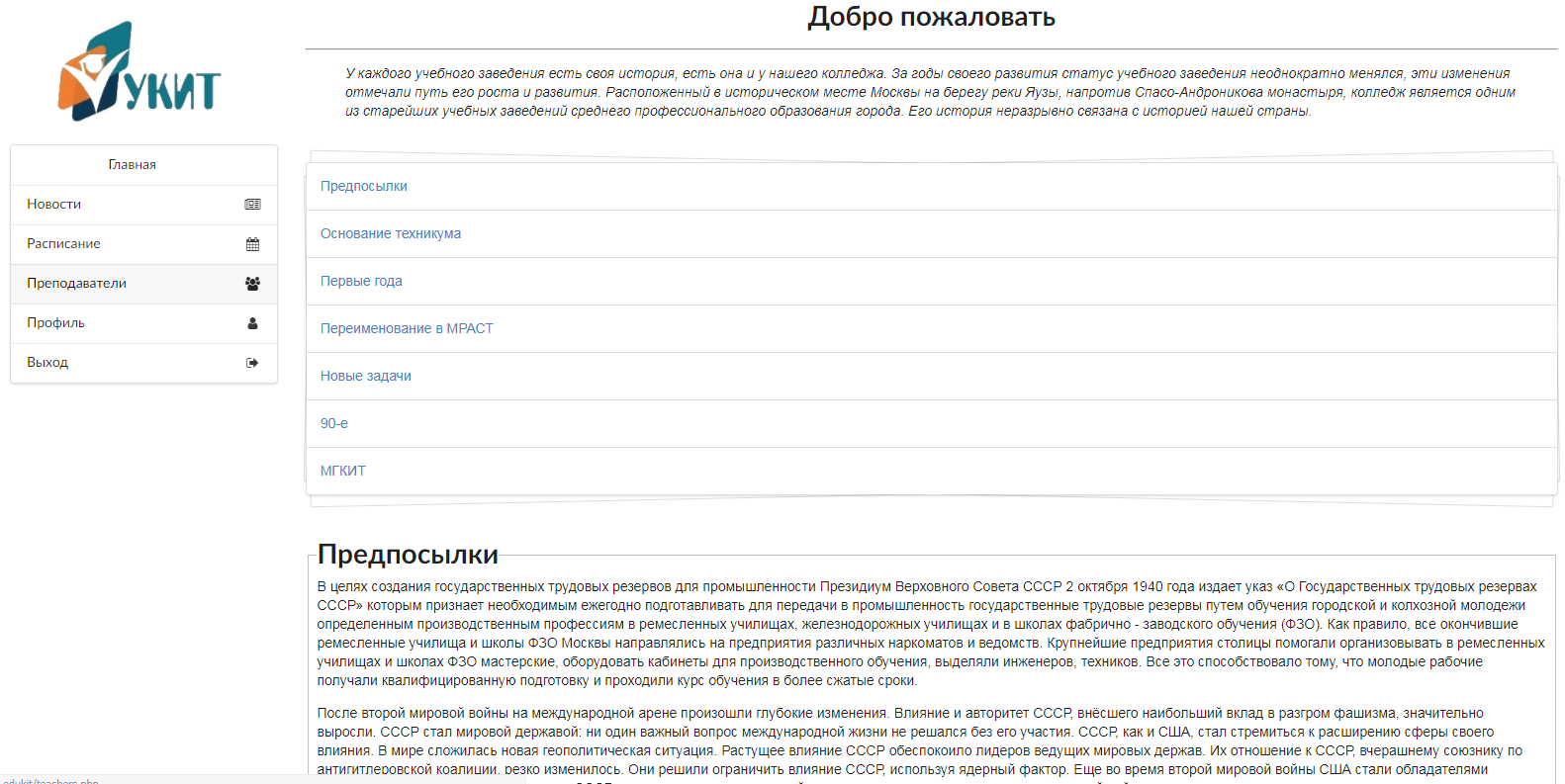


Рисунок 15 – Пользователь произвёл аутентификацию в систему

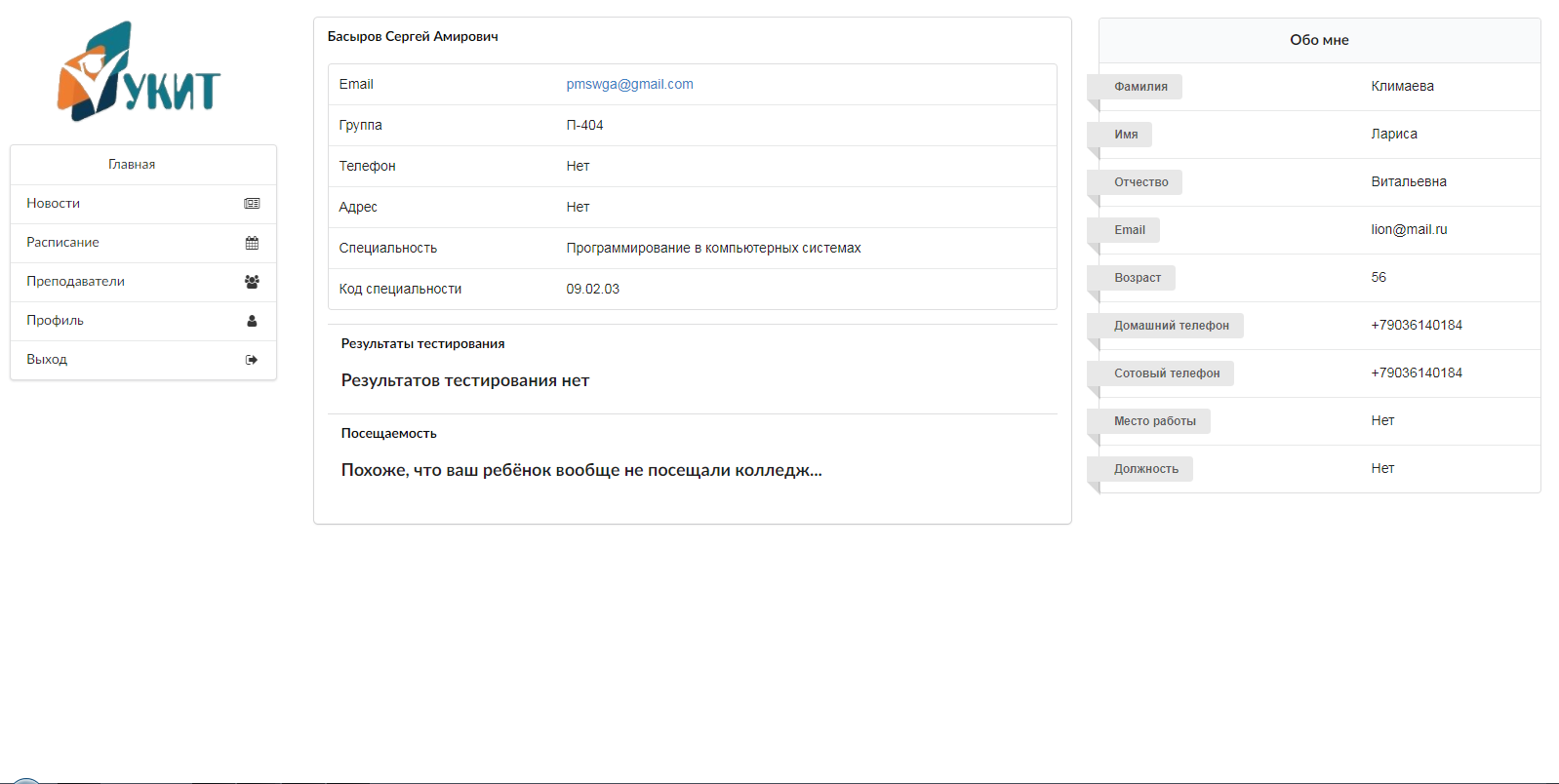


Рисунок 16 – Пользователь произвёл аутентификацию в систему как родитель

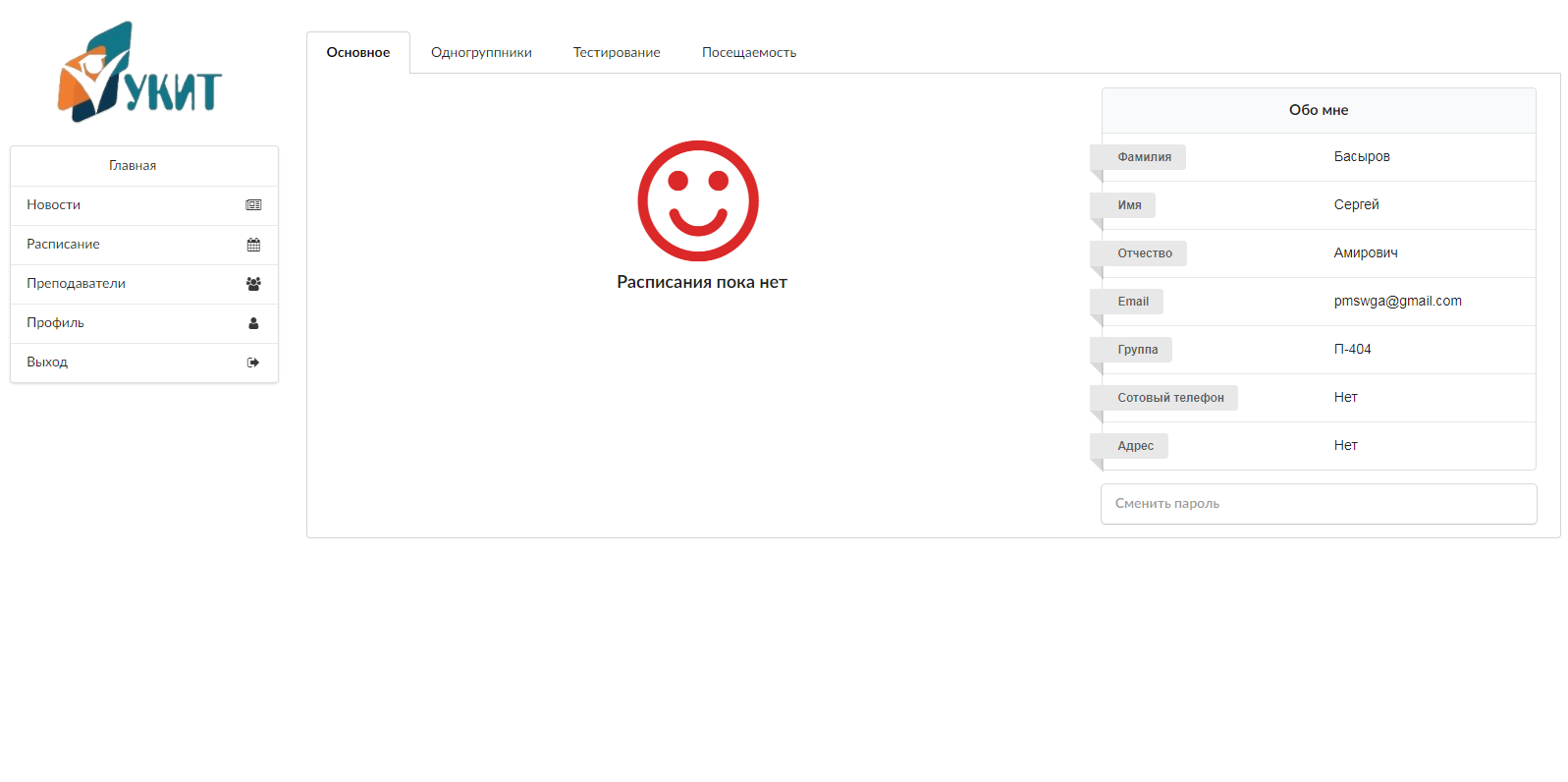


Рисунок 17 – Пользоваетль произвёл аутнетификацию как студент

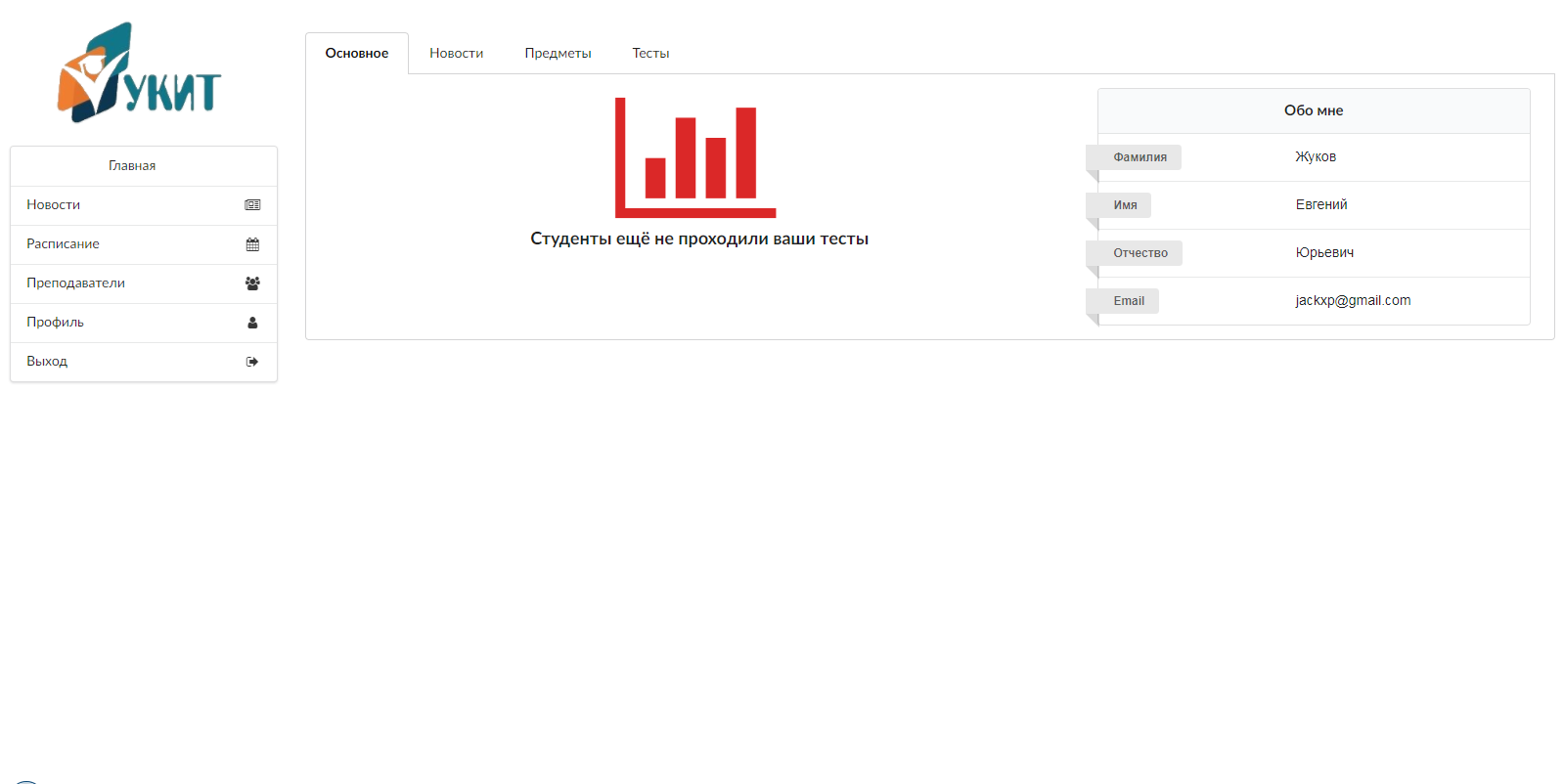


Рисунок 18 – Пользователь произвёл аутентификацию как преподаватель

### Вывод

В данном разделе были описаны методы тестирования, которые использовались для проверки соответствия требованиям технического задания.

# заключение

Результатом работы оказался разработанный информационно-образовательный портал, который включает в себя следующее:

1. Система тестирования;
2. Система управления расписанием;
3. Система оповещения;
4. Панель администратора:
   1. Работа с пользователями;
   2. Работа со специальностями;
   3. Работа с предметами;
   4. Работа с группами.

Достоинства разработанного портала:

1. Доступность, так как является веб-сайтом;
2. Централизованность хранения данных;
3. Интерфейс адаптирован как для настольных, так и для мобильных экранов.

Недостатки разработанного портала:

1. Слабоустойчивая архитектура приложения;
2. Достаточно сложна в сопровождении.

В итоге, было сделано заключение о том, что разработка подобных информационных порталов должна происходит в команде от 3 до n разработчиков, так как разрабатывать одному долго и неэффективно. Подход к разработке подобных систем давно выработан и заключается в написании модулей, интерфейсы которых унифицированы для совместимости друг с другом и непосредственного взаимодействия между ними.

Разработка подобных проектов требует учёта множества важных деталей, особенностей предметной области. Цель данного курсового проекта – это попытка разработки большой целостной системы, которая автоматизирует несколько учебных процессов, однако затянутый процесс разработки не позволил довести проект до нужного уровня.

Так же были закреплены навыки:

* Объекто-ориентированный анализ и проектирование;
* Проектирования базы данных и приведении к 3НФ;
* Работы с СУБД MySQL;
* Написания запросов SQL;
* Оформление документации удовлетворяющие стандартам ГОСТ;
* Программирование на PHP, JavaScript;
* Использования системы контроля версий.

# список Использованных источников

1 Форта, Бен. SQL за 10 минут, 4-е издание. [Текст]: / Б.Форта. – Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2015. – 288 с.;

2 Тейлор Аллен, Дж. SQL для чайников, 8-е издание. [Текст]: / А.Тейлор. – Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2016. – 416 с.;

3 Минник, Крис. JavaScript для чайников. [Текст]: / К. Минник, Е. Холланд. Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2016. – 320 с.;

4 Сырых, Ю.А. Современный веб-дизайн. Настольный и мобильный. 3-е издание. [Текст]: М. : ООО “И.Д. Вильямс”, 2014. – 384 с.;

5 Пайлон Д. UML для программистов. [Текст]: / Д. Пайлон, Н. Питмен. – СПб.: Питер, 2012. – 240 с.;

6 Котеров, Д.В. PHP 7. [Текст]: / Д.В. Котеров, И.В. Симдянов. – СПб: БХВ-Петербург, 2016. – 1088 с.;

7 Котеров, Д.В. PHP 5. [Текст]: / Д.В. Котеров, А.Ф. Костарев. – СПб.: БХВ-Петербург, 2014. – 1104 с.;

8 Колисниченко Д.Н. PHP и MySQL. Разработка веб-приложений. – 5-е издание., перераб. и доп. [Текст]: / Д.Н. Колисниченко. – СПб.: БХВ-Петербург, 2015. – 592 с.;

9 Локхарт, Джош. Современный PHP. Новые возможности и передовой опыт. [Текст]: / Д. Локхарт. – Пер. с англ. Рагимов Р.Н. – М.: ДМК Пресс, 2016. – 304 с.;

10 MySQL Documentation [Электронный ресурс]. – Режим доступа: https://dev.mysql.com/doc/;

11 PHP: Hypertext Preprocessor [Электронный ресурс]. – Режим доступа: http://php.net/;

12 The W3 Markup Validation Service [Электронный ресурс]. – Режим доступа: https://validator.w3.org/.

# приложение а

**Руководство оператора**

1. Назначение программы

### Наименование

Информационно-образовательный портал EDUKIT

### Назначение

Обеспечение свободного доступа студентам, преподавателям и родителям к данным образовательной деятельности.

### Характеристика области применения

Информационно-образовательный портал предназначен для использования лицами ФГБОУ ВО «Московский государственный университет технологий и управления имени К.Г. Разумовского (ПКУ)». Университетский колледж информационных технологий, владеющие русским языком и имеющие практические навыки в работе с интернет-браузером.

1. Условия выполнения программы

### Аппаратные средства

В состав аппаратных средств пользователя должен входит компьютер, который включает в себя:

* Процессор;
* Видеокарта;
* Жёсткий диск;
* Сетевая карта;
* Клавиатура;
* Мышь.

### Программные средства

В состав программных средств пользователя должен входит интернет браузер Google Chrome.

1. Выполнение программы

### Запуск программы

Для запуска программы необходимо открыть браузер и ввести в адресную строку URL адрес, на котором расположен информационно-образовательный портал.

### Регистрация пользователя

Для регистрации в качестве студента необходимо нажать на пункт меню «Регистрация» на главной панели меню. Перед вами откроется модальное окно с полями, которые необходимо заполнить, чтобы зарегистрироваться на портале.

Для регистрации в качестве родителя необходимо нажать на пункт меню «Регистрация» на главной панели меню. Перед вами откроется модальное окно с полями ввода, однако вы должны нажать на кнопку «Я родитель», которая расположена между кнопками «Сбросить» и «Готово». Вы будете перенаправлены на страницу с формой регистрации для родителя.

Для регистрации в качестве преподавателя необходимо обратиться к администратору портала, чтобы он вас зарегистрировал.

### Аутентификация пользователя

Для того чтобы пройти аутентификацию на портале (т.е. войти в личный кабинет), необходимо нажать на кнопку «Вход» на главной панели меню. Перед вами откроется модальное окно с полями для ввода логина и пароля. После ввода данных нажмите на кнопку «Войти». Если ваши данные присутствуют в базе данных портала, то страница перезагрузится и вы сможете зайти в свой личный кабинет.

### Руководство пользователя

Более подробную информацию вы можете узнать нажав на кнопку «Руководство пользователя» на главной панели меню. Вы будете перенаправлены на страницу с описанием основных разделов сайта и личных кабинетов.

1. Сообщения оператору

### «Регистрация закрыта»

Возникает в том случае, если администратор портала не составил группы, в которые могут зарегистрироваться студенты.

### «Такого пользователя не существует»

Возникает тогда, когда пользователем были введены неверные данные для входа на портал.

### «Пароль должен иметь длину не менее 6-ти символов»

Возникает в том случае, если пользователем при регистрации был введён пароль, длина которого меньше 6-ти символов.

### «Пароли не совпадают»

Возникает в том случае, если пользователь при регистрации указал несовпадающий пароль в поле «Повторить пароль» с полем «Пароль» и наоборот.

### «Вы должны принять соглашение»

Возникает в том случае, если пользователь при регистрации не поставил галочку отвечающую за согласие на обработку персональных данных.

# ПРИЛОЖЕНИЕ Б

**ИСХОДНЫЙ КОД**

Таблица 29 – Исходный код информационно-образовательного портала

|  |
| --- |
| **typeusers.consts.php** |
| <?php      /\*!    \name Типы пользователей  \author pmswga  \version 1.0  \brief По этим константам проверяется уровень доступа к функциональным возможностям сайта    \*/    const USER\_TYPE\_ADMIN = 0; ///< Администратор  const USER\_TYPE\_TEACHER = 1; ///< Преподаватель  const USER\_TYPE\_ELDER = 2; ///< Староста  const USER\_TYPE\_STUDENT = 3; ///< Студент  const USER\_TYPE\_PARENT = 4; ///< Родитель    ?> |
| **gm.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Managers;    require\_once "iep.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/group.class.php";    use IEP\Managers\IEP;  use IEP\Structures\Specialty;  use IEP\Structures\Group;    /\*!    \class GroupManager gm.class.php "iep/managers/gm.class.php"  \extends IEP  \brief Менеджер для управления группами  \author pmswga  \version 1.0    Задачи менеджера групп:  1. Добавление/удаление групп  2. Изменение данных о группе (название, специальность)  3. Выборка групп    \*/    class GroupManager extends IEP  {    /\*!  \brief Добавляет группу  \param[in] $group - Группа  \note Объект класса Group  \return TRUE - успешно, FALSE - ошибка  \*/    public function add($group) : bool  {  if (  !empty($group->getNumberGroup()) &&  !empty($group->getSpec()) &&  !empty($group->getYearEducation()) &&  !empty($group->getStatus())  ) {  $add\_group\_query = $this->dbc()->prepare("call addGroup(:descp, :spec\_id, :year\_edu, :is\_budget)");    $add\_group\_query->bindValue(":descp", $group->getNumberGroup());  $add\_group\_query->bindValue(":spec\_id", $group->getSpec());  $add\_group\_query->bindValue(":year\_edu", $group->getYearEducation());  $add\_group\_query->bindValue(":is\_budget", $group->getStatus());    return $add\_group\_query->execute();  } else {  return false;  }  }    /\*!  \brief Возвращает группы, которые привязаны к тесту  \param[in] $test\_id - Идентификатор теста  \return Группы  \note Массив с объктами класса Group  \*/    public function getGroups(int $test\_id) : array  {  $db\_groups = $this->query("call getTestGroups(:test\_id)", [":test\_id" => $test\_id]);    $groups = array();  foreach ($db\_groups as $db\_group) {    $spec = new Specialty($db\_group['spec\_code'], $db\_group['spec\_descp']);  $spec->setSpecialtyID((int)$db\_group['spec\_id']);    $group = new Group(  $db\_group['grp\_descp'],  $spec,  $db\_group['grp\_edu\_year'],  (int)$db\_group['grp\_payment']  );  $group->setGroupID((int)$db\_group['grp\_id']);    $groups[] = $group;  }    return $groups;  }    /\*!  \brief Возвращает группы, которые непривязаны к тесту  \param[in] $test\_id - Идентификатор теста  \return Группы  \note Массив с объктами класса Group  \*/    public function getUnsetGroups(int $test\_id) : array  {  $db\_groups = $this->query("call getUnsetGroups(:test\_id)", [":test\_id" => $test\_id]);    $unset\_groups = array();  foreach ($db\_groups as $db\_group) {    $spec = new Specialty($db\_group['spec\_code'], $db\_group['spec\_descp']);  $spec->setSpecialtyID((int)$db\_group['spec\_id']);    $group = new Group(  $db\_group['grp\_descp'],  $spec,  $db\_group['grp\_edu\_year'],  (int)$db\_group['grp\_payment']  );  $group->setGroupID((int)$db\_group['grp\_id']);    $unset\_groups[] = $group;  }    return $unset\_groups;  }    /\*!  \*/  public function getGroupsOfCurrentYear()  {  $db\_groups = $this->query("call getGroupsOfCurrentYear()");  $groups = array();  foreach ($db\_groups as $db\_group) {  $spec = new Specialty(  $db\_group['spec\_code'],  $db\_group['spec\_descp'],  $db\_group['spec\_file']  );  $spec->setSpecialtyID((int)$db\_group['spec\_id']);    $group = new Group($db\_group['number'], $spec, $db\_group['edu\_year'], (int)$db\_group['budget']);  $group->setGroupID((int)$db\_group['id\_grp']);    $groups[] = $group;  }  return $groups;  }    /\*!  \brief Возвращает все группы  \return Группы  \note Массив с объктами класса Group  \*/    public function getAllGroups() : array  {  $db\_groups = $this->query("call getAllGroups()");    $groups = array();  foreach ($db\_groups as $db\_group) {  $spec = new Specialty(  $db\_group['spec\_code'],  $db\_group['spec\_descp'],  $db\_group['spec\_file']  );  $spec->setSpecialtyID((int)$db\_group['spec\_id']);    $group = new Group($db\_group['number'], $spec, $db\_group['edu\_year'], (int)$db\_group['budget']);  $group->setGroupID((int)$db\_group['id\_grp']);    $groups[] = $group;  }    return $groups;  }    /\*!  \brief Изменяет название группы  \param[in] $grp\_id - Идентификатор группы  \param[in] $new\_descp - Новое название группы  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeDescriptionGroup(int $grp\_id, string $new\_descp) : bool  {  $change\_descp\_grp\_query = $this->dbc()->prepare("call changeDescriptionGroup(:grp, :descp)");    $change\_descp\_grp\_query->bindValue(":grp", $grp\_id);  $change\_descp\_grp\_query->bindValue(":descp", $new\_descp);    return $change\_descp\_grp\_query->execute();  }    /\*!  \brief Изменяет специальность группы  \param[in] $grp\_id - Идентификатор группы  \param[in] $spec\_id - Новый идентификатор группы  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeSpecGroup(int $grp\_id, int $spec\_id) : bool  {  $change\_spec\_grp\_query = $this->dbc()->prepare("call changeSpecGroup(:grp, :spec)");    $change\_spec\_grp\_query->bindValue(":grp", $grp\_id);  $change\_spec\_grp\_query->bindValue(":spec", $spec\_id);    return $change\_spec\_grp\_query->execute();  }    /\*!  \brief Переводит группы на курс выше  \param[in] $grp\_id - Идентификатор группы  \return TRUE - успешно, FALSE - ошибка  \*/    public function upCourse(int $grp\_id) : bool  {  $up\_course\_grp\_query = $this->dbc()->prepare("call upCourse(:grp)");    $up\_course\_grp\_query->bindValue(":grp", $grp\_id);    return $up\_course\_grp\_query->execute();  }    /\*!  \brief Удаляет группу  \param[in] $group\_id - Идентификатор группы  \return TRUE - успешно, FALSE - ошибка  \*/    public function remove($group\_id) : bool  {  if (!empty($group\_id) && ($group\_id > 0)) {  $remove\_group\_query = $this->dbc()->prepare("call removeGroup(:grp\_id)");    $remove\_group\_query->bindValue(":grp\_id", $group\_id);    return $remove\_group\_query->execute();  } else {  return false;  }  }    }    ?> |
| **iep.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Managers;    /\*!    \class IEP iep.class.php "iep/managers/iep.class.php"  \brief Базовый класс для все остальных менеджеров  \author pmswga  \version 1.0    Все запросы к базе данных осуществляются через PDO    Базовые функций, выполняемые менеджером:  1. Добавление данных  2. Удаление данных  3. Выполнение запроса к БД напрямую  4. Работа с логами (Выборка, удаление)      Логи ведутся в базе данных:  1. На добавление  2. На удаление  3. На изменение    С помощью специальных методов можно получить доступ к логам    \*/    abstract class IEP  {  private $DBC; ///< Контроллер базы данных (PDO)    /\*!  \brief Задаёт контроллер базы данных    Базовый конструктор для всех унаследованных менеджеров    \param[in] $dbc - Контроллер базы данных  \note Объект класса PDO  \*/    function \_\_construct(\PDO $dbc)  {  $this->DBC = $dbc;  }    /\*!  \brief Задаёт контроллер базы данных  \param[in] $dbc - Контроллер базы данных  \note Объект класса PDO  \*/    public function setDBC($dbc)  {  $this->DBC = $dbc;  }    /\*!  \brief Возвращает контроллер базы данных  \return Контроллер базы данных  \note Объект класса PDO  \*/    public function dbc()  {  return $this->DBC;  }    /\*!  \brief Выполняет запрос  \param[in] $sql - SQL код  \param[in] $params - Аргументы  \returns  TRUE или FALSE, если запрос не предполагает выборку данных, а иначе ассоциативный массив с данными  \*/    public function query(string $sql, array $params = array())  {  if (!empty($params)) {  $query = $this->dbc()->prepare($sql);  $result = $query->execute($params);  return ($result) ? $query->fetchAll(\PDO::FETCH\_ASSOC) : false;  } else {  return $this->dbc()->query($sql)->fetchAll(\PDO::FETCH\_ASSOC);  }  }    /\*!  \brief Возвращает все записи логов  \return Логи  \note Массив с записями  \*/    public function getLogs() : array  {  $logs = $this->DBC->query("call readLogs('all')")->fetchAll(\PDO::FETCH\_ASSOC);    return $logs;  }    /\*!  \brief Удаляет запись лога  \param[in] $log\_id - идентификатор записи логи  \*/    public function removeLogs(int $log\_id) : bool  {  $remove\_log\_query = $this->dbc()->prepare("call removeLog(:log)");    $remove\_log\_query->bindValue(":log", $log\_id);    return $remove\_log\_query->execute();  }    /\*!  \brief Добавление каких-либо данных в базу данных  \param[in] $data - Данные для добавления  \*/    abstract public function add($data);    /\*!  \brief Удаление каких-либо данными из базы данных  \param[in] $what - Данные для удаления  \*/    abstract public function remove($what);    }    ?> |
| **nm.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Managers;    require\_once "iep.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/news.class.php";    use IEP\Managers\IEP;  use IEP\Structures\News;    /\*!    \class NewsManager nm.class.php "iep/managers/nm.class.php"  \extends IEP  \brief Менеджер для работы с новостями  \author pmswga  \version 1.0    \*/    class NewsManager extends IEP  {    /\*!  \brief Добавление новой новости  \note Метод используется для добавления новостей преподавателем  \param[in] $news - новости  \return TRUE - успешно, FALSE - ошибка  \*/    public function add($news)  {  $add\_news\_query = $this->dbc()->prepare("call addNews(:caption, :content, :author, :date)");  $add\_news\_query->bindValue(":caption", $news->getCaption());  $add\_news\_query->bindValue(":content", $news->getContent());  $add\_news\_query->bindValue(":author", $news->getAuthor());  $add\_news\_query->bindValue(":date", $news->getDatePublication());    return $add\_news\_query->execute();  }    /\*!  \brief Добавление новой новости администратором  \note Метод используется для добавления новостей администратором  \param[in] $news - Новости  \note Массив с объектами класса News  \return TRUE - успешно, FALSE - ошибка  \*/    public function addAdminNews($news) : bool  {  $add\_news\_query = $this->dbc()->prepare("call addAdminNews(:caption, :content, :author, :date)");  $add\_news\_query->bindValue(":caption", $news->getCaption());  $add\_news\_query->bindValue(":content", $news->getContent());  $add\_news\_query->bindValue(":author", $news->getAuthor());  $add\_news\_query->bindValue(":date", $news->getDatePublication());    return $add\_news\_query->execute();  }    public function getAdminNewsByID(int $news\_id) : array  {  return $get\_news\_query = $this->query("SELECT \* FROM `admin\_news` WHERE `id\_news`=:news\_id", [":news\_id" => $news\_id]);  }    /\*!  \brief Возвращает все новости опубликованные преподавателем  \param[in] $teacher\_email - электронная почта преподавателя  \return Новости  \note Массив с объектами класса News  \*/    public function getNews(string $teacher\_email) : array  {  $db\_news = $this->query("call getNews(:t\_email)", [":t\_email" => $teacher\_email]);    $news = array();  foreach ($db\_news as $db\_new) {  $new = new News($db\_new['caption'], $db\_new['content'], $db\_new['author'], $db\_new['dp']);  $new->setNewsID((int)$db\_new['id\_news']);    $news[] = $new;  }    return $news;  }    /\*!  \brief Возвращает все новости опубликованные администратором  \param[in] $admin\_email - электронная почта администратора  \return Новости  \note Массив с объектами класса News  \*/    public function getAdminNews(string $admin\_email) : array  {  $db\_news = $this->query("call getAdminNews(:admin\_email)", [":admin\_email" => $admin\_email]);    $news = array();  foreach ($db\_news as $db\_new) {  $new = new News($db\_new['caption'], $db\_new['content'], $admin\_email, $db\_new['dp']);  $new->setNewsID((int)$db\_new['id\_news']);    $news[] = $new;  }    return $news;  }    /\*!  \brief Возвращает все опубликованные новости  \return Новости  \note Массив с объектами класса News  \*/    public function getAllNews() : array  {  $db\_news = $this->query("call getAllNews()");    $news = array();  foreach ($db\_news as $db\_new) {  $new = new News($db\_new['caption'], $db\_new['content'], $db\_new['author'], $db\_new['dp']);  $new->setNewsID((int)$db\_new['id\_news']);    $news[] = $new;  }    return $news;  }    /\*!  \brief Изменяет заголовок новости  \param[in] $news\_id - Идентификатор новости  \param[in] $new\_caption - Новое содержимое  \note Содержимое новости может быть представлено в формате HTML  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeCaptionNews(int $news\_id, string $new\_caption) : bool  {  $change\_caption\_news\_query = $this->dbc()->prepare("call changeCaptionNews(:news\_id, :caption)");  $change\_caption\_news\_query->bindValue(":news\_id", $news\_id);  $change\_caption\_news\_query->bindValue(":caption", $new\_caption);    return $change\_caption\_news\_query->execute();  }    /\*!  \brief Изменяет содержимое новости  \param[in] $news\_id - Идентификатор новости  \param[in] $new\_conent - Новое содержимое  \note Содержимое новости может быть представлено в формате HTML  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeContentNews(int $news\_id, string $new\_conent) : bool  {  $change\_conent\_news\_query = $this->dbc()->prepare("call changeContentNews(:news\_id, :conent)");  $change\_conent\_news\_query->bindValue(":news\_id", $news\_id);  $change\_conent\_news\_query->bindValue(":conent", $new\_conent);    return $change\_conent\_news\_query->execute();  }    /\*!  \brief Удаляет новость, добавленную администратором  \param[in] $news\_id - Идентификатор новости  \return TRUE - успешно, FALSE - ошибка  \*/    public function removeAdminNews(int $news\_id) : bool  {  $remove\_news\_query = $this->dbc()->prepare("call removeAdminNews(:news\_id)");  $remove\_news\_query->bindValue(":news\_id", $news\_id);    return $remove\_news\_query->execute();  }    /\*!  \brief Возвращает все опубликованные новости  \param[in] $news\_id - Идентификатор новости  \return TRUE - успешно, FALSE - ошибка  \*/    public function remove($news\_id) : bool  {  $remove\_news\_query = $this->dbc()->prepare("call removeNews(:news\_id)");  $remove\_news\_query->bindValue(":news\_id", $news\_id);    return $remove\_news\_query->execute();  }    }    ?> |
| **sbm.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Managers;    require\_once "iep.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/subject.class.php";    use IEP\Managers\IEP;  use IEP\Structures\Subject;    /\*!    \class SubjectManager sbm.class.php "iep/managers/sbm.class.php"  \extends IEP  \brief Менеджер для управления предметами  \author pmswga  \version 1.0    Задачи менеджера:  1.    \*/    class SubjectManager extends IEP  {    /\*!  \brief Добавление нового предмета  \param[in] $subject - Предмет  \note Объект класса Subject  \return TRUE - успешно, FALSE - ошибка  \*/    public function add($subject)  {  $add\_subject\_query = $this->dbc()->prepare("call addSubject(:descp)");    $add\_subject\_query->bindValue(":descp", $subject->getDescription());    return $add\_subject\_query->execute();  }    /\*!  \brief Возвращает предметы, которые ведёт преподаватель  \param[in] $teacher\_email - Электронная почта преподавателя  \return Предметы  \note Массив с объектами класса Subject  \*/    public function getSubjects(string $teacher\_email)  {  $db\_subjects = $this->query("call getSubjects(:t\_email)", [":t\_email" => $teacher\_email]);    $subjects = array();  foreach ($db\_subjects as $db\_subject) {  $subject = new Subject($db\_subject['description']);  $subject->setSubjectID((int)$db\_subject['id\_subject']);    $subjects[] = $subject;  }    return $subjects;  }    /\*!  \brief Возвращает все предметы  \return Предметы  \note Массив с объектами класса Subject  \*/    public function getAllSubjects() : array  {  $db\_subjects = $this->query("call getAllSubjects()");    $subjects = array();  foreach ($db\_subjects as $db\_subject) {  $subject = new Subject($db\_subject['descp']);  $subject->setSubjectID((int)$db\_subject['id\_subject']);    $subjects[] = $subject;  }    return $subjects;  }    /\*!  \brief Назначает предмет преподавателю  \param[in] $teacher\_email - Электронная почта преподавателя  \param[in] $subject\_id - Идентификатор предмета  \return TRUE - успешно, FALSE - ошибка  \*/    public function setSubject(string $teacher\_email, int $subject\_id) : bool  {  $set\_subject\_query = $this->dbc()->prepare("call setSubject(:t\_email, :subject\_id)");    $set\_subject\_query->bindValue(":t\_email", $teacher\_email);  $set\_subject\_query->bindValue(":subject\_id", $subject\_id);    return $set\_subject\_query->execute();  }    /\*!  \brief Снимает предмет с преподавателя  \param[in] $teacher\_email - Электронная почта преподавателя  \param[in] $subject\_id - Идентификатор предмета  \return TRUE - успешно, FALSE - ошибка  \*/    public function unsetSubject(string $teacher\_email, int $subject\_id) : bool  {  $unset\_subject\_query = $this->dbc()->prepare("call unsetSubject(:t\_email, :subject\_id)");    $unset\_subject\_query->bindValue(":t\_email", $teacher\_email);  $unset\_subject\_query->bindValue(":subject\_id", $subject\_id);    return $unset\_subject\_query->execute();  }    /\*!  \brief Возвращает все неназначенные предметы на преподавателя  \param[in] $teacher\_email - Электронная почта преподавателя  \return Предметы  \note Массив с объектами класса Subject  \*/    public function getUnsetSubjects(string $teacher\_email) : array  {  $db\_unset\_subjects = $this->query("call getUnsetSubjects(:t\_email)", [":t\_email" => $teacher\_email]);    $unset\_subjects = array();  foreach ($db\_unset\_subjects as $db\_unset\_subject) {  $subject = new Subject($db\_unset\_subject['description']);  $subject->setSubjectID((int)$db\_unset\_subject['id\_subject']);    $unset\_subjects[] = $subject;  }    return $unset\_subjects;  }    /\*!  \brief Изменяет название предмета  \param[in] $subject\_id - Идентификатор предмета  \param[in] $new\_descp - Новое название предмета  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeDescriptionSubject(int $subject\_id, string $new\_descp) : bool  {  $change\_subject\_query = $this->dbc()->prepare("call changeDescriptionSubject(:subject\_id, :new\_descp)");    $change\_subject\_query->bindValue(":subject\_id", $subject\_id);  $change\_subject\_query->bindValue(":new\_descp", $new\_descp);    return $change\_subject\_query->execute();  }    /\*!  \brief Удаляет предмет  \param[in] $subject\_id - Идентификатор предмета  \return TRUE - успешно, FALSE - ошибка  \*/    public function remove($subject\_id) : bool  {  $remove\_subject\_query = $this->dbc()->prepare("call removeSubject(:subject\_id)");    $remove\_subject\_query->bindValue(":subject\_id", $subject\_id);    return $remove\_subject\_query->execute();  }    }    ?> |
| **shm.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Managers;    require\_once "iep.class.php";    use IEP\Managers\IEP;    /\*!    \class ScheduleManager shm.class.php "iep/managers/shm.class.php"  \extends IEP  \brief Менеджер для работы с расписанием  \author pmswga  \version 1.0    Задачи менеджера:  1.    \*/    class ScheduleManager extends IEP  {    /\*!  \brief Форматирует числовое представление дня недели в строку  \param[in] $day - Номер дня недели  \return Строковое представление дня недели  \note 1 - ПН, 2 - ВТ, 3 - СР, 4 - ЧТ, 5 - ПТ, 6 - СБ  \*/    private function intToDay(int $day) : string  {  switch ($day)  {  case 1: return "Понедельник"; break;  case 2: return "Вторник"; break;  case 3: return "Среда"; break;  case 4: return "Четверг"; break;  case 5: return "Пятница"; break;  case 6: return "Суббота"; break;  default: return ""; break;  }  }    /\*!  \brief Добавление новой записи о расписании  \param[in] $schedule - Запись о расписании  \note Ассоциативный массив в формате  \code    $schedule\_entry = array(  "group" => ,  "day" => ,  "pair" => ,  "subject" =>  );    \endcode    \return TRUE - успешно, FALSE - ошибка  \*/    public function add($schedule) : bool  {  if (  !empty($schedule['group']) &&  !empty($schedule['day']) &&  !empty($schedule['pair']) &&  !empty($schedule['subj\_1']) &&  !empty($schedule['subj\_2'])  ) {  $add\_schedule = $this->dbc()->prepare("call addScheduleEntry(:grp, :day, :pair, :subj\_1, :subj\_2)");    $add\_schedule->bindValue(":grp", $schedule['group']);  $add\_schedule->bindValue(":day", $schedule['day']);  $add\_schedule->bindValue(":pair", $schedule['pair']);  $add\_schedule->bindValue(":subj\_1", $schedule['subj\_1']);  $add\_schedule->bindValue(":subj\_2", $schedule['subj\_2']);    return $add\_schedule->execute();  } else {  return false;  }    }    /\*!  \brief Добавление новой записи об изменениях в расписании  \param[in] $schedule - Запись об изменениях в расписании  \note Ассоциативный массив в формате  \code    $schedule\_entry = array(  "group" => ,  "day" => ,  "pair" => ,  "subject" =>  );    \endcode    \return TRUE - успешно, FALSE - ошибка  \*/    public function addChangeSchedule($schedule) : bool  {  $add\_schedule = $this->dbc()->prepare("call addChangeSchedule(:grp, :day, :pair, :subject)");    $add\_schedule->bindValue(":grp", $schedule['group']);  $add\_schedule->bindValue(":day", $schedule['day']);  $add\_schedule->bindValue(":pair", $schedule['pair']);  $add\_schedule->bindValue(":subject", $schedule['subject']);    return $add\_schedule->execute();  }    /\*!  \brief Возвращает расписание всех групп  \return Расписание  \note Ассоциативный массив в формате  \code    $schedule = array(  "П-304" => array(  "ПН" => array(  [0] => array(  [\_day] => ,  [group] => ,  [id\_grp] => ,  [pair] => ,  [subject] =>  ),  ...  )  ...  ),  ...  );    \endcode  \*/    public function getAllScheduleGroup() : array  {  $data = $this->query("call getAllScheduleGroup()");    $dataByGroup = array();  foreach ($data as $d) {  $dataByGroup[$d['group']." (".$d['edu\_year'].")"][] = $d;  }    $dataByGroupByDay = array();  foreach ($dataByGroup as $key => $value) {    foreach ($value as $day) {  $dataByGroupByDay[$key][$this->intToDay((int)$day['\_day'])][] = $day;  }    }    return $dataByGroupByDay;  }    /\*!  \brief Возвращает расписание группы  \param[in] $grp - Идентификатор  \return Расписание  \note Ассоциативный массив  \*/    public function getScheduleGroup(int $grp)  {  $data = $this->query("call getScheduleGroup(:g)", [":g" => $grp]);    $dataByGroup = array();  foreach ($data as $d) {  $dataByGroup[$d['group']." (".$d['edu\_year'].")"][] = $d;  }    $dataByGroupByDay = array();  foreach ($dataByGroup as $key => $value) {    foreach ($value as $day) {  $dataByGroupByDay[$key][$this->intToDay((int)$day['\_day'])][] = $day;  }    }    return $dataByGroupByDay;  }    /\*!  \brief Возвращает изменения во всех группах  \return Изменения в расписании  \note Ассоциативный массив в формате  \code    $schedule = array(  "П-304" => array(  "2017-09-02 00:00:00" => array(  [0] => array(  [\_day] => ,  [group] => ,  [id\_grp] => ,  [pair] => ,  [subject] =>  ),  ...  )  ...  ),  ...  );    \endcode  \*/  public function getAllChangedSchedule()  {  $data = $this->query("call getAllChangedSchedule()");  $dataByGroup = array();  foreach ($data as $d) {  $dataByGroup[$d['group']." (".$d['edu\_year'].")"][] = $d;  }  $dataByGroupByDay = array();  foreach ($dataByGroup as $key => $value) {  foreach ($value as $day) {  $dataByGroupByDay[$key][$day['\_day']][] = $day;  }  }  return $dataByGroupByDay;  }    /\*!  \brief Возвращает изменения в расписании группы  \param[in] $grp - Идентификатор  \return Расписание  \note Ассоциативный массив  \*/  public function getChangeScheduleGroup(int $grp)  {  $data = $this->query("call getChangeScheduleGroup(:g)", [":g" => $grp]);    $dataByGroup = array();  foreach ($data as $d) {  $dataByGroup[$d['group']][] = $d;  }    $dataByGroupByDay = array();  foreach ($dataByGroup as $key => $value) {    foreach ($value as $day) {  $dataByGroupByDay[$key][$day['\_day']][] = $day;  }    }    return $dataByGroupByDay;  }    /\*!  \brief Изменяет предмет  \param[in] $grp - Идентификатор группы  \param[in] $day - День недели  \param[in] $pair - Пара  \param[in] $subj\_1 - Идентфикатор предмета, который на нечётной неделе  \param[in] $subj\_2 - Идентфикатор предмета, который на чётной неделе  \return TRUE - успешно, FALSE - ошибка  \*/    public function changePair(int $grp, int $day, int $pair, int $subj\_1, int $subj\_2) : bool  {  $change\_query = $this->dbc()->prepare("UPDATE `schedule` SET `subj\_1`=:s1, `subj\_2`=:s2 WHERE `id\_grp`=:g AND `pair`=:p AND `\_day`=:d");  $change\_query->bindValue(":g", $grp);  $change\_query->bindValue(":d", $day);  $change\_query->bindValue(":p", $pair);  $change\_query->bindValue(":s1", $subj\_1);  $change\_query->bindValue(":s2", $subj\_2);    return $change\_query->execute();  }    /\*!  \brief Изменяет предмет в изменениях в расписании  \param[in] $grp - Идентификатор группы  \param[in] $day - День недели  \param[in] $pair - Пара  \param[in] $subject - Идентфикатор предмета  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeChangedPair(int $grp, string $day, int $pair, int $subject) : bool  {  $change\_query = $this->dbc()->prepare("UPDATE `changed\_schedule` SET `subject`=:s WHERE `id\_grp`=:g AND `pair`=:p AND `\_day`=:d");  $change\_query->bindValue(":g", $grp);  $change\_query->bindValue(":d", $day);  $change\_query->bindValue(":p", $pair);  $change\_query->bindValue(":s", $subject);    return $change\_query->execute();  }    /\*!  \brief Удаляет записи об изменениях в расписании  \param[in] $grp - Идентификатор группы  \return TRUE - успешно, FALSE - ошибка  \*/    public function deleteChangedPair(int $grp) : bool  {  $change\_query = $this->dbc()->prepare("DELETE FROM `changed\_schedule` WHERE `id\_grp`=:g");  $change\_query->bindValue(":g", $grp);    return $change\_query->execute();  }    /\*!  \brief Удаление записи о расписании  \param[in] $schedule - Запись о расписании  \warning Ожидает реализации  \bug Хз, с начала времён тут  \*/    public function remove($schedule)  {    }    }    ?> |
| **sm.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Managers;    require\_once "iep.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/specialty.class.php";    use IEP\Managers\IEP;  use IEP\Structures\Specialty;    /\*!    \class SpecialtyManager sm.class.php "iep/managers/sm.class.php"  \extends IEP  \brief Менеджер для работы со специальностями  \author pmswga  \version 1.0    \*/    class SpecialtyManager extends IEP  {    /\*!  \brief Добавляет новую специальность  \param[in] $spec - Специальность  \note Объект класс Specialty  \return TRUE - успешно, FALSE - ошибка  \*/    public function add($spec) : bool  {  if (  !empty($spec->getCode()) &&  !empty($spec->getDescription()) &&  !empty($spec->getFilepath())  ) {    $add\_spec\_query = $this->dbc()->prepare("call addSpecialty(:code, :descp, :file)");    $add\_spec\_query->bindValue(":code", $spec->getCode());  $add\_spec\_query->bindValue(":descp", $spec->getDescription());  $add\_spec\_query->bindValue(":file", $spec->getFilepath());    return $add\_spec\_query->execute();  } else {  return false;  }    }    /\*!  \brief Возвращает все специальности  \return Специальности  \note Массив с объектами класса Specialty  \*/    public function getAllSpecialty() : array  {  $db\_specs = $this->query("call getAllSpecialty()");    $specs = array();  foreach ($db\_specs as $db\_spec) {  $spec = new Specialty($db\_spec['code'], $db\_spec['descp'], $db\_spec['file']);  $spec->setSpecialtyID((int)$db\_spec['id\_spec']);    $specs[] = $spec;  }    return $specs;  }    /\*!  \brief Изменяет код специальности  \param[in] $spec\_id - Идентификатор специальности  \param[in] $code - Новый код специальности  \note Формат кода специальности "ХХ.ХХ.ХХ", где Х - число от 0 до 9  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeCodeSpecialty(int $spec\_id, string $code) : bool  {  $change\_code\_spec\_query = $this->dbc()->prepare("call changeCodeSpecialty(:spec\_id, :code)");    $change\_code\_spec\_query->bindValue(":spec\_id", $spec\_id);  $change\_code\_spec\_query->bindValue(":code", $code);    return $change\_code\_spec\_query->execute();  }    /\*!  \brief Изменяет название специальности  \param[in] $spec\_id - Идентификатор специальности  \param[in] $descp - Новое название специальности  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeDescpSpecialty(int $spec\_id, string $descp) : bool  {  $change\_descp\_spec\_query = $this->dbc()->prepare("call changeDescriptionSpecialty(:spec\_id, :descp)");    $change\_descp\_spec\_query->bindValue(":spec\_id", $spec\_id);  $change\_descp\_spec\_query->bindValue(":descp", $descp);    return $change\_descp\_spec\_query->execute();  }    /\*!  \brief Изменяет файл специальности  \param[in] $spec\_id - Идентификатор специальности  \param[in] $file - Путь до файла специальности  \note Файл должен быть в формате PDF  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeFileSpecialty(int $spec\_id, string $file) : bool  {  $change\_file\_spec\_query = $this->dbc()->prepare("call changeFileSpecialty(:spec\_id, :file)");    $change\_file\_spec\_query->bindValue(":spec\_id", $spec\_id);  $change\_file\_spec\_query->bindValue(":file", $file);    return $change\_file\_spec\_query->execute();  }    /\*!  \brief Удаляет специальность  \param[in] $spec\_id - Идентификатор специальности  \return TRUE - успешно, FALSE - ошибка  \*/    public function remove($spec\_id) : bool  {  $remove\_spec\_query = $this->dbc()->prepare("call removeSpecialty(:spec\_id)");    $remove\_spec\_query->bindValue(":spec\_id", $spec\_id);    return $remove\_spec\_query->execute();  }    }    ?> |
| **tm.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Managers;    require\_once "iep.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/test.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/testquestion.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/studentanswer.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/subject.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/group.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/studenttest.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/studentresult.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/student.class.php";    use IEP\Managers\IEP;  use IEP\Structures\Subject;  use IEP\Structures\Specialty;  use IEP\Structures\Group;  use IEP\Structures\Test;  use IEP\Structures\TestQuestion;  use IEP\Structures\StudentAnswer;  use IEP\Structures\StudentTest;  use IEP\Structures\StudentResult;  use IEP\Structures\Student;  use IEP\Structures\User;  /\*!  \class TestManager tm.class.php "iep/managers/tm.class.php"  \extends IEP  \brief Менеджер для работы с тестами  \author pmswga  \version 1.0    \*/    class TestManager extends IEP  {    /\*!  \brief Добавляет новый тест  \param[in] $test - Тест  \note Объект класса Test  \return TRUE - успешно, FALSE - ошибка  \*/    public function add($test)  {  try  { //< Блокировка таблиц !!!  $this->dbc()->beginTransaction();    $test\_add\_query = $this->dbc()->prepare("call addTest(:emailTeacher, :subject, :caption)");    $test\_add\_query->bindValue(":subject", $test->getSubject());  $test\_add\_query->bindValue(":emailTeacher", $test->getAuthor());  $test\_add\_query->bindValue(":caption", $test->getCaption());    if ($test\_add\_query->execute()) {    $for\_groups = $test->getGroups();    if (!empty($for\_groups)) {    $last\_id = $this->query("SELECT LAST\_INSERT\_ID() as last\_id FROM `tests`");  $last\_id = $last\_id[0]['last\_id'];    $result = true;  for ($i = 0; $i < count($for\_groups); $i++) {  $result \*= $this->setGroup((int)$last\_id, (int)$for\_groups[$i]);  }    if ($result) {  return $this->dbc()->commit();  }  else {  $this->dbc()->rollBack();  return false;  }    } else {  return $this->dbc()->commit();  }    }  else {  $this->dbc()->rollBack();  return false;  }    }  catch(PDOException $e)  {  $this->dbc()->rollBack();  return false;  }  }    /\*!  \brief Возвращает все тесты доступные группе  \param[in] $grp - Идентификатор группы  \return Тесты  \note Массив с объектами класса Test  \*/    public function getTestsForGroup(int $grp) : array  {  $db\_tests = $this->query("call getTestsForGroup(:grp)", [":grp" => $grp]);    $tests = array();  foreach ($db\_tests as $db\_test) {    $subject = new Subject($db\_test['subject\_caption']);  $subject->setSubjectID((int)$db\_test['subject\_id']);    $db\_questions = $this->query("call getQuestions(:test\_id)", [":test\_id" => $db\_test['id\_test']]);    $questions = array();  foreach ($db\_questions as $db\_question) {    $db\_answers = $this->query("call getAnswers(:question\_id)", [":question\_id" => $db\_question['id\_question']]);    $answers = array();  foreach ($db\_answers as $db\_answer) {  $answers[] = array(  "id" => $db\_answer['id\_answer'],  "answer" => $db\_answer['answer']  );  }    $question = new TestQuestion($db\_question['question'], $db\_question['r\_answer']);  $question->setQuestionID((int)$db\_question['id\_question']);  $question->setAnswers($answers);    $questions[] = $question;  }    $test = new Test($db\_test['caption'], $subject, $db\_test['author']);  $test->setTestID((int)$db\_test['id\_test']);  $test->setQuestions($questions);    $tests[] = $test;  }    return $tests;  }    /\*!  \brief Возвращает тест по его идентификатору  \param[in] $test\_id - Идентификатор теста  \return Тест  \note Объект класса Test  \*/    public function getTest(int $test\_id) : Test  {  $db\_test = $this->query("call getTest(:test\_id)", [":test\_id" => $test\_id])[0];    $subject = new Subject($db\_test['subject\_caption']);  $subject->setSubjectID((int)$db\_test['subject\_id']);    $db\_questions = $this->query("call getQuestions(:test\_id)", [":test\_id" => $db\_test['id\_test']]);    $questions = array();  foreach ($db\_questions as $db\_question) {    $db\_answers = $this->query("call getAnswers(:question\_id)", [":question\_id" => $db\_question['id\_question']]);    $answers = array();  foreach ($db\_answers as $db\_answer) {  $answers[] = array(  "id" => $db\_answer['id\_answer'],  "answer" => $db\_answer['answer']  );  }    $question = new TestQuestion($db\_question['question'], $db\_question['r\_answer']);  $question->setQuestionID((int)$db\_question['id\_question']);  $question->setAnswers($answers);    $questions[] = $question;  }    $test = new Test($db\_test['caption'], $subject, $db\_test['author']);  $test->setQuestions($questions);    return $test;  }    /\*!  \brief Возвращает все тесты, созданные преподавателем  \param[in] $teacher\_email - Электронная почта преподавателя  \return Тесты  \note Массив с объектами класса Test  \*/    public function getTests(string $teacher\_email) : array  {  $db\_tests = $this->query("call getTests(:t\_email)", [":t\_email" => $teacher\_email]);    $tests = array();  foreach ($db\_tests as $db\_test) {    $subject = new Subject($db\_test['subject\_caption']);  $subject->setSubjectID((int)$db\_test['subject\_id']);    $db\_groups = $this->query("call getTestGroups(:test\_id)", [":test\_id" => $db\_test['id\_test']]);    $groups = array();  foreach ($db\_groups as $db\_group) {    $spec = new Specialty($db\_group['spec\_code'], $db\_group['spec\_descp']);  $spec->setSpecialtyID((int)$db\_group['spec\_id']);    $group = new Group(  $db\_group['grp\_descp'],  $spec,  $db\_group['grp\_edu\_year'],  (int)$db\_group['grp\_payment']  );    $groups[] = $group;  }    $db\_questions = $this->query("call getQuestions(:test\_id)", [":test\_id" => $db\_test['id\_test']]);    $questions = array();  foreach ($db\_questions as $db\_question) {    $db\_answers = $this->query("call getAnswers(:question\_id)", [":question\_id" => $db\_question['id\_question']]);    $answers = array();  foreach ($db\_answers as $db\_answer) {  $answers[] = array(  "id" => $db\_answer['id\_answer'],  "answer" => $db\_answer['answer']  );  }    $question = new TestQuestion($db\_question['question'], $db\_question['r\_answer']);  $question->setQuestionID((int)$db\_question['id\_question']);  $question->setAnswers($answers);    $questions[] = $question;  }    $test = new Test($db\_test['caption'], $subject, $db\_test['author'], $groups);  $test->setTestID((int)$db\_test['id\_test']);  $test->setQuestions($questions);    $tests[] = $test;  }    return $tests;  }    /\*!  \brief Возвращает все созданные тесты  \return Тесты  \note Массив с объектами класса Test  \*/    public function getAllTests() : array  {  $db\_tests = $this->query("call getAllTests()");    $tests = array();  foreach ($db\_tests as $db\_test) {    $subject = new Subject($db\_test['subject\_caption']);  $subject->setSubjectID((int)$db\_test['subject\_id']);    $db\_groups = $this->query("call getTestGroups(:test\_id)", [":test\_id" => $db\_test['id\_test']]);    $groups = array();  foreach ($db\_groups as $db\_group) {    $spec = new Specialty($db\_group['code\_spec'], $db\_group['spec\_descp']);  $spec->setSpecialtyID((int)$db\_group['id\_spec']);    $group = new Group(  $db\_group['grp'],  $spec,  $db\_group['edu\_year'],  (int)$db\_group['is\_budget']  );    $groups[] = $group;  }    $db\_questions = $this->query("call getQuestions(:test\_id)", [":test\_id" => $db\_test['id\_test']]);    $questions = array();  foreach ($db\_questions as $db\_question) {    $db\_answers = $this->query("call getAnswers(:question\_id)", [":question\_id" => $db\_question['id\_question']]);    $answers = array();  foreach ($db\_answers as $db\_answer) {  $answers[] = array(  "id" => $db\_answer['id\_answer'],  "answer" => $db\_answer['answer']  );  }    $question = new TestQuestion($db\_question['question'], $db\_question['r\_answer']);  $question->setQuestionID((int)$db\_question['id\_question']);  $question->setAnswers($answers);    $questions[] = $question;  }    $test = new Test($db\_test['caption'], $subject, $db\_test['author'], $groups);  $test->setTestID((int)$db\_test['id\_test']);  $test->setQuestions($questions);    $tests[] = $test;  }    return $tests;  }    /\*!  \brief Проверяет привязанность группы к тесту  \param[in] $test\_id - Идентификатор теста  \param[in] $grp\_id - Идентификатор группы  \return TRUE, FALSE  \warning Возвращает не bool, а int  \*/    public function isGroupForTest(int $test\_id, int $grp\_id)  {  return $this->query("select isGroupForTest(:test\_id, :grp) as result",  [":test\_id" => $test\_id, ":grp" => $grp\_id]  )[0]['result'];  }    /\*!  \brief Добавляет новый вопрос в тест  \param[in] $test\_id - Идентификатор теста  \param[in] $question - Вопрос  \note Объект класса TestQuestion  \return TRUE - успешно, FALSE - ошибка  \*/    public function addQuestion(int $test\_id, TestQuestion $question)  {  try  {  $this->dbc()->beginTransaction();    $add\_question\_query = $this->dbc()->prepare("call addQuestion(:test\_id, :question, :r\_answer)");    $add\_question\_query->bindValue(":test\_id", $test\_id);  $add\_question\_query->bindValue(":question", $question->getQuestion());  $add\_question\_query->bindValue(":r\_answer", $question->getRAnswer());    if (!empty($question->getAnswers())) {    if ($add\_question\_query->execute()) {    $last\_id = $this->query("SELECT LAST\_INSERT\_ID() as last\_id FROM `tests` WHERE `id\_test`=:test\_id", [":test\_id" => $test\_id]);  $last\_id = $last\_id[0]['last\_id'];    $add\_answer\_query = $this->dbc()->prepare("call addAnswer(:question\_id, :answ)");  $add\_answer\_query->bindValue(":question\_id", $last\_id);    $result = true;  foreach ($question->getAnswers() as $answer) {  $add\_answer\_query->bindValue(":answ", $answer);    $result \*= $add\_answer\_query->execute();  }    if ($result) {  return $this->dbc()->commit();  } else {  $this->dbc()->rollBack();  return false;  }    } else {  $this->dbc()->rollBack();  return false;  }  }    }  catch(PDOException $e)  {  $this->dbc()->rollBack();  return false;  }  }    /\*!  \brief Добавляет ответ к вопросу  \param[in] $question\_id - Идентификатор вопроса  \param[in] $answer - Ответ  \return TRUE - успешно, FALSE - ошибка  \*/    public function addAnswer(int $question\_id, string $answer) : bool  {  $add\_answer\_query = $this->dbc()->prepare("call addAnswer(:question\_id, :answ)");    $add\_answer\_query->bindValue(":question\_id", $question\_id);  $add\_answer\_query->bindValue(":answ", $answer);    return $add\_answer\_query->execute();  }    /\*!  \brief Возвращает все ответы на вопрос  \param[in] $question\_id - Идентификатор теста  \return Ответы  \note Ассоциативнный массив  \*/    public function getAnswers(int $question\_id)  {  $answers\_query = $this->dbc()->prepare("call getAnswers(:question\_id)");    $answers\_query->bindValue(":question\_id", $question\_id);    return $answers\_query->execute();  }    /\*!  \brief Изменяет заголовок к тесту  \param[in] $test\_id - Идентификатор теста  \param[in] $test\_caption - Новый заголовок теста  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeCaptionTest(int $test\_id, string $test\_caption) : bool  {  $test\_change = $this->dbc()->prepare("call changeCaptionTest(:test\_id, :test\_caption)");    $test\_change->bindValue(":test\_id", $test\_id);  $test\_change->bindValue(":test\_caption", $test\_caption);    return $test\_change->execute();  }    /\*!  \brief Изменяет предмет, по которому идёт тестирование  \param[in] $test\_id - Идентификатор теста  \param[in] $subject\_id - Идентификатор предмета  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeSubjectTest(int $test\_id, int $subject\_id) : bool  {  $test\_change = $this->dbc()->prepare("call changeSubjectTest(:test\_id, :subject\_id)");    $test\_change->bindValue(":test\_id", $test\_id);  $test\_change->bindValue(":subject\_id", $subject\_id);    return $test\_change->execute();  }    /\*!  \brief Изменяет заголовок вопроса  \param[in] $question\_id - Идентификатор теста  \param[in] $new\_caption - Новый заголовок вопроса  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeCaptionQuestion(int $question\_id, string $new\_caption) : bool  {  $change\_question\_query = $this->dbc()->prepare("call changeCaptionQuestion(:question\_id, :new\_caption)");    $change\_question\_query->bindValue(":question\_id", $question\_id);  $change\_question\_query->bindValue(":new\_caption", $new\_caption);    return $change\_question\_query->execute();  }    /\*!  \brief Изменяет правильный ответ на вопрос  \param[in] $question\_id - Идентификатор теста  \param[in] $new\_RAnswer - Новый правильный ответ  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeRAnswerQuestion(int $question\_id, string $new\_RAnswer) : bool  {  $change\_question\_query = $this->dbc()->prepare("call changeRAnswerQuestion(:question\_id, :new\_RAnswer)");    $change\_question\_query->bindValue(":question\_id", $question\_id);  $change\_question\_query->bindValue(":new\_RAnswer", $new\_RAnswer);    return $change\_question\_query->execute();  }    /\*!  \brief Изменяет ответ на вопрос  \param[in] $answer\_id - Идентификатор ответа  \param[in] $new\_answer - Новый ответ  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeCaptionAnswer(int $answer\_id, string $new\_answer) : bool  {  $change\_answer\_query = $this->dbc()->prepare("call changeCaptionAnswer(:answer\_id, :new\_answer)");    $change\_answer\_query->bindValue(":answer\_id", $answer\_id);  $change\_answer\_query->bindValue(":new\_answer", $new\_answer);    return $change\_answer\_query->execute();  }    /\*!  \brief Назначает группу на тест  \param[in] $test\_id - Идентификатор теста  \param[in] $test\_grp - Идентификатор группы  \return TRUE - успешно, FALSE - ошибка  \*/    public function setGroup(int $test\_id, int $test\_grp) : bool  {  $set\_group\_query = $this->dbc()->prepare("call setGroup(:test\_id, :test\_grp)");    $set\_group\_query->bindValue(":test\_id", $test\_id);  $set\_group\_query->bindValue(":test\_grp", $test\_grp);    return $set\_group\_query->execute();  }    /\*!  \brief Снимает группу с теста  \param[in] $test\_id - Идентификатор теста  \param[in] $test\_grp - Идентификатор группы  \return TRUE - успешно, FALSE - ошибка  \*/    public function unsetGroup(int $test\_id, int $test\_grp) : bool  {  $set\_group\_query = $this->dbc()->prepare("call unsetGroup(:test\_id, :test\_grp)");    $set\_group\_query->bindValue(":test\_id", $test\_id);  $set\_group\_query->bindValue(":test\_grp", $test\_grp);    return $set\_group\_query->execute();  }    /\*!  \brief Удаляет вопрос  \param[in] $question\_id - Идентификатор вопроса  \return TRUE - успешно, FALSE - ошибка  \*/    public function removeQuestion(int $question\_id) : bool  {  $remove\_question\_query = $this->dbc()->prepare("call removeQuestion(:question\_id)");  $remove\_question\_query->bindValue(":question\_id", $question\_id);    return $remove\_question\_query->execute();  }    /\*!  \brief Удаляет ответ на вопрос  \param[in] $answer\_id - Идентификатор ответа  \return TRUE - успешно, FALSE - ошибка  \*/    public function removeAnswer(int $answer\_id) : bool  {  $remove\_answer\_query = $this->dbc()->prepare("call removeAnswer(:answer\_id)");    $remove\_answer\_query->bindValue(":answer\_id", $answer\_id);    return $remove\_answer\_query->execute();  }    /\*!  \brief Записывает результаты прохождения теста  \param[in] $student\_answer - ответы студента  \note Объект класса StudentAnswer  \return TRUE - успешно, FALSE - ошибка  \*/    public function putStudentAnswer(StudentAnswer $student\_answer) : bool  {  try  {  $this->dbc()->beginTransaction();    $add\_student\_answer\_query = $this->dbc()->prepare("call createStudentTest(:test\_id, :student\_email, :subject, :caption, :mark)");    $add\_student\_answer\_query->bindValue(":test\_id", $student\_answer->getTestID());  $add\_student\_answer\_query->bindValue(":student\_email", $student\_answer->getStudent()->getEmail());  $add\_student\_answer\_query->bindValue(":subject", $student\_answer->getSubject());  $add\_student\_answer\_query->bindValue(":caption", $student\_answer->getCaption());  $add\_student\_answer\_query->bindValue(":mark", $student\_answer->getMark());    if ($add\_student\_answer\_query->execute()) {    $answers = $student\_answer->getAnswers();    if (!empty($answers)) {  $result = true;  for ($i = 0; $i < count($answers); $i++) {    $add\_answer\_query = $this->dbc()->prepare("call putStudentAnswer(:student\_test, :question, :answer)");    $add\_answer\_query->bindValue(":student\_test", $student\_answer->getTestID());  $add\_answer\_query->bindValue(":question", $answers[$i]['question']);  $add\_answer\_query->bindValue(":answer", $answers[$i]['answer']);    $result \*= $add\_answer\_query->execute();  }    if ($result) {  return $this->dbc()->commit();  } else {  $this->dbc()->rollBack();  return false;  }    } else {  $this->dbc()->rollBack();  return false;  }    } else {  $this->dbc()->rollBack();  return false;  }    }  catch(PDOException $e)  {  $this->dbc()->rollBack();  return false;  }  }    /\*!  \brief Возвращает все ответы к пройденному тесту  \param[in] $student\_test\_id - Идентификатор пройденного теста  \return Ответы студента  \note Ассоциативнный массив  \*/    public function getStudentAnswers(int $student\_test\_id)  {  return $this->query("call getStudentAnswers(:student\_test)", [":student\_test" => $student\_test\_id]);  }    /\*!  \brief Возвращает тест, пройденный студентом  \param[in] $student\_email - Электронная почта студента  \param[in] $student\_test - Идентификатор пройденного теста  \return Результат пройденного теста  \note Объект класса StudentTest  \*/    public function getStudentTest(string $student\_email, int $student\_test)  {  $db\_test = $this->dbc()->prepare("call getStudentTest(:student\_email, :student\_test)");  $db\_test->bindValue(":student\_test", $student\_test);  $db\_test->bindValue(":student\_email", $student\_email);    if ($db\_test->execute()) {    $test = $db\_test->fetchAll(\PDO::FETCH\_ASSOC)[0];    if (!empty($test)) {  return new StudentTest(  $student\_email,  $test['caption'],  $test['subject'],  $test['date\_pass'],  (int)$test['mark']  );  } else {  return array();  }    } else {  return array();  }    }    /\*!  \brief Возвращает все пройденные тесты  \param[in] $student\_email - Электронная почта студента  \return Пройденные тесты  \note Массив с объектами класса StudentTest  \*/    public function getStudentTests(string $student\_email) : array  {  $db\_students\_tests = $this->query("call getStudentTests(:student)", [":student" => $student\_email]);    $student\_tests = array();  foreach ($db\_students\_tests as $db\_student\_test) {  $test = new StudentTest(  $student\_email,  $db\_student\_test['caption'],  $db\_student\_test['subject'],  $db\_student\_test['date\_pass'],  (int)$db\_student\_test['mark']  );  $test->setTestID((int)$db\_student\_test['id\_student\_test']);    $student\_tests[] = $test;  }    return $student\_tests;  }  /\*!  \*/  public function getStudentsResult(string $teacher\_email) : array  {    }    /\*!  \brief Удаляет тест  \param[in] $test\_id - Идентификатор теста  \return TRUE - успешно, FALSE - ошибка  \*/    public function remove($test\_id) : bool  {  $remove\_test\_query = $this->dbc()->prepare("call removeTest(:test\_id)");    $remove\_test\_query->bindValue(":test\_id", $test\_id);    return $remove\_test\_query->execute();  }    }    ?> |
| **trm.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Managers;    require\_once "iep.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/trafficentry.class.php";    use IEP\Managers\IEP;  use IEP\Structures\TrafficEntry;    /\*!  \class TrafficManager trm.class.php "iep/managers/trm.class.php"  \extends IEP  \brief Менеджер для работы с записями посещамости  \author pmswga  \version 1.0    Задачи менеджера:  1. Добавлять новые записи об посещамости студента  2. Получать список записей об посещамости конкретного студента  3. Получать весь список записей посещамости студентов    \*/    class TrafficManager extends IEP  {    /\*!  \brief Добавляет новую запись об посещамости студента  \param[in] $traffic\_entry - Запись об посещамости  \note Объект класса TrafficEntry  \return TRUE - успешно, FALSE - ошибка  \*/    public function add($traffic\_entry) : bool  {  $add\_traffic\_query = $this->dbc()->prepare("call addTrafficEntry(:s\_email, :date\_visit, :cph, :cah)");    $add\_traffic\_query->bindValue(":s\_email", $traffic\_entry->getStudentEmail());  $add\_traffic\_query->bindValue(":date\_visit", $traffic\_entry->getDateVisit());  $add\_traffic\_query->bindValue(":cph", $traffic\_entry->getCountPassedPair());  $add\_traffic\_query->bindValue(":cah", $traffic\_entry->getCountAllPair());    return $add\_traffic\_query->execute();  }    /\*!  \brief Возвращает информацию об посещамости студента  \param[in] $student\_email - электронная почта студента  \return Массив с записями об посещамости  \*/    public function getStudentTraffic(string $student\_email) : array  {  if (!empty($student\_email)) {  return $this->query("call getTrafficStudent(:s\_email)", [":s\_email" => $student\_email]);  } else {  return array();  }  }    /\*!  \brief Удаляет запись об посещамости студента  \param[in] $traffic\_entry -  \warning Ожидает реализации  \*/    public function remove($traffic\_entry) : bool  {  return false;  }    }    ?> |
| **um.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Managers;    require\_once "iep.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/consts/typeusers.consts.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/user.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/teacher.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/student.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/parent.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/group.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/specialty.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/structures/subject.class.php";    use IEP\Structures\User;  use IEP\Structures\Teacher;  use IEP\Structures\Student;  use IEP\Structures\Parent\_;  use IEP\Structures\Group;  use IEP\Structures\Specialty;  use IEP\Structures\Subject;    /\*!    \class UserManager um.class.php "iep/managers/um.class.php"  \extends IEP  \brief Менеджер для работы с пользователями  \author pmswga  \version 1.0    \*/    class UserManager extends IEP  {    /\*!  \brief Проверяет, существует ли такой администратор в базей данных  \param[in] $user - администратор  \note Объект класса User  \return TRUE - успешно, FALSE - ошибка  \*/    public function adminExists(User $user) : bool  {  $exists\_user = $this->query("SELECT \* FROM `admins` WHERE `email`=:email AND `passwd`=:passwd", [  ":email" => $user->getEmail(),  ":passwd" => $user->getPassword()  ])[0];    if (  $exists\_user['sn'] === $user->getSn() &&  $exists\_user['fn'] === $user->getFn() &&  $exists\_user['pt'] === $user->getPt() &&  $exists\_user['email'] === $user->getEmail() &&  $exists\_user['passwd'] === $user->getPassword()  ) {  return true;  } else {  return false;  }  }    /\*!  \brief Проверяет, существует ли такой пользователь в базей данных  \param[in] $user - пользователь  \note Объект класса User  \return TRUE - успешно, FALSE - ошибка  \*/    public function userExists(User $user) : bool  {  $exists\_user = $this->query("SELECT \* FROM `users` WHERE `email`=:email AND `passwd`=:passwd AND `id\_type\_user`=:user\_type", [  ":email" => $user->getEmail(),  ":passwd" => $user->getPassword(),  ":user\_type" => $user->getUserType()  ])[0];  if (  $exists\_user['sn'] === $user->getSn() &&  $exists\_user['fn'] === $user->getFn() &&  $exists\_user['pt'] === $user->getPt() &&  $exists\_user['email'] === $user->getEmail() &&  $exists\_user['passwd'] === $user->getPassword() &&  $exists\_user['id\_type\_user'] === $user->getUserType()  ) {  return true;  } else {  return false;  }  }    /\*!  \brief Добавляет нового пользователя  \param[in] $user - Новый пользователь  \note Объект класса User  \*/    public function add($user)  {  switch ($user->getUserType())  {  case USER\_TYPE\_TEACHER:  {  try  {    if (!empty($user->getSn()) &&  !empty($user->getFn()) &&  !empty($user->getPt()) &&  !empty($user->getEmail()) &&  !empty($user->getPassword()) &&  !empty($user->getInfo())  ) {  $this->dbc()->beginTransaction();    $add\_teacher\_query = $this->dbc()->prepare("call addTeacher(:sn, :fn, :pt, :email, :paswd, :info)");    $add\_teacher\_query->bindValue(":sn", $user->getSn());  $add\_teacher\_query->bindValue(":fn", $user->getFn());  $add\_teacher\_query->bindValue(":pt", $user->getPt());  $add\_teacher\_query->bindValue(":email", $user->getEmail());  $add\_teacher\_query->bindValue(":paswd", $user->getPassword());  $add\_teacher\_query->bindValue(":info", $user->getInfo());    if ($add\_teacher\_query->execute()) {    $subjects = $user->getSubjects();    if (!empty($subjects)) {    $set\_subject\_query = $this->dbc()->prepare("call setSubject(:email, :subject)");  $set\_subject\_query->bindValue(":email", $user->getEmail());    $result = true;  for ($i = 0; $i < count($subjects); $i++) {  $set\_subject\_query->bindValue(":subject", $subjects[$i]);    $result \*= $set\_subject\_query->execute();  }    if ($result) {  return $this->dbc()->commit();  } else {  $this->dbc()->rollBack();  return false;  }    }  else {  return $this->dbc()->commit();  }    } else {  $this->dbc()->rollBack();  return false;  }    } else {  return false;  }    }  catch(PDOException $e)  {  $this->dbc()->rollBack();  return false;  }  } break;  case USER\_TYPE\_STUDENT:  {  try  {  $this->dbc()->beginTransaction();    $add\_student\_query = $this->dbc()->prepare("call addStudent(:sn, :fn, :pt, :email, :paswd, :ha, :cp, :grp)");    $add\_student\_query->bindValue(":sn", $user->getSn());  $add\_student\_query->bindValue(":fn", $user->getFn());  $add\_student\_query->bindValue(":pt", $user->getPt());  $add\_student\_query->bindValue(":email", $user->getEmail());  $add\_student\_query->bindValue(":paswd", $user->getPassword());  $add\_student\_query->bindValue(":ha", $user->getHomeAddress());  $add\_student\_query->bindValue(":cp", $user->getCellPhone());  $add\_student\_query->bindValue(":grp", $user->getGroup());    $result = $add\_student\_query->execute();    if ($result) {  return $this->dbc()->commit();  } else {  $this->dbc()->rollBack();  return false;  }    }  catch(PDOException $e)  {  $this->dbc()->rollBack();  return false;  }  } break;  case USER\_TYPE\_PARENT:  {  try  {  echo "<pre>";  print\_r($user);  echo "</pre>";    $this->dbc()->beginTransaction();    $add\_parent\_query = $this->dbc()->prepare("call addParent(:sn, :fn, :pt, :email, :paswd, :age, :education, :wp, :post, :hp, :cp)");    $add\_parent\_query->bindValue(":sn", $user->getSn());  $add\_parent\_query->bindValue(":fn", $user->getFn());  $add\_parent\_query->bindValue(":pt", $user->getPt());  $add\_parent\_query->bindValue(":email", $user->getEmail());  $add\_parent\_query->bindValue(":paswd", $user->getPassword());  $add\_parent\_query->bindValue(":age", $user->getAge());  $add\_parent\_query->bindValue(":education", $user->getEducation());  $add\_parent\_query->bindValue(":wp", $user->getWorkPlace());  $add\_parent\_query->bindValue(":post", $user->getPost());  $add\_parent\_query->bindValue(":hp", $user->getHomePhone());  $add\_parent\_query->bindValue(":cp", $user->getCellPhone());    if ($add\_parent\_query->execute()) {    $childs = $user->getChilds();    if (!empty($childs)) {    $result = true;  for ($i = 0; $i < count($childs); $i++) {  $result \*= $this->setChild($user->getEmail(), $childs[$i], 6);  }    if ($result) {  return $this->dbc()->commit();  } else {  $this->dbc()->rollBack();  return false;  }    } else {  return $this->dbc()->commit();  }    } else {  $this->dbc()->rollBack();  return false;  }    }  catch(PDOException $e)  {  $this->dbc()->rollBack();  return false;  }  } break;  default:  {  $add\_admin\_query = $this->dbc()->prepare("call addAdmin(:sn, :fn, :pt, :email, :passwd)");    $add\_admin\_query->bindValue(":sn", $user->getSn());  $add\_admin\_query->bindValue(":fn", $user->getFn());  $add\_admin\_query->bindValue(":pt", $user->getPt());  $add\_admin\_query->bindValue(":email", $user->getEmail());  $add\_admin\_query->bindValue(":passwd", $user->getPassword());    return $add\_admin\_query->execute();  } break;  }    }    /\*!  \brief Производит аутентификацию пользователя  \param[in] $email - Электронная почта пользователя  \param[in] $passwd - Пароль  \return FALSE - пользователь не существует, объект класса User - пользователь существует  \*/    public function authentification(string $email, string $passwd)  {  $user = $this->query("call authentification(:email, :passwd)", [":email" => $email, ":passwd" => $passwd])[0];    switch($user['id\_type\_user'])  {  case USER\_TYPE\_TEACHER:  {  $teacher = $this->query("call getTeacherInfo(:email)", [":email" => $user['email']])[0];    if (!empty($teacher)) {    $teacher = new Teacher(  new User(  $teacher['sn'],  $teacher['fn'],  $teacher['pt'],  $teacher['email'],  $teacher['paswd'],  (int)$teacher['type\_user']  ),  $teacher['info']  );    return $teacher;    } else {  return false;  }    } break;  case USER\_TYPE\_STUDENT:  {  $student = $this->query("call getStudentInfo(:email)", [":email" => $user['email']])[0];  if (!empty($student)) {    $spec = new Specialty($student['spec\_code'], $student['spec\_descp'], "none");  $spec->setSpecialtyID((int)$student['spec\_id']);    $group = new Group($student['grp'], $spec, $student['edu\_year'], (int)$student['is\_budget']);  $group->setGroupID((int)$student['grp\_id']);    $s = new Student(  new User(  $student['sn'],  $student['fn'],  $student['pt'],  $student['email'],  $student['paswd'],  (int)$student['type\_user']  ),  $student['home\_address'],  $student['cell\_phone'],  $group  );    return $s;  } else {  return false;  }    } break;  case USER\_TYPE\_PARENT:  {  $parent = $this->query("call getParentInfo(:email)", [":email" => $user['email']])[0];    if (!empty($parent)) {    $db\_childs = $this->query("call getChilds(:email)", [":email" => $parent['email']]);    $childs = array();  foreach ($db\_childs as $db\_child) {  $childs[] = $this->authentification($db\_child["email"], $db\_child["passwd"]);  }    $p = new Parent\_(  new User(  $parent['sn'],  $parent['fn'],  $parent['pt'],  $parent['email'],  $parent['paswd'],  (int)$parent['type\_user']  ),  (int)$parent['age'],  $parent['education'],  $parent['work\_place'],  $parent['post'],  $parent['home\_phone'],  $parent['cell\_phone']  );    $p->setChilds($childs);    return $p;  } else {  return false;  }    } break;  case USER\_TYPE\_ELDER:  {  $elder = $this->query("call getElderInfo(:email)", [":email" => $user['email']])[0];  if (!empty($elder)) {    $spec = new Specialty($elder['spec\_code'], $elder['spec\_descp'], "none");  $spec->setSpecialtyID((int)$elder['spec\_id']);    $group = new Group($elder['grp'], $spec, $elder['edu\_year'], (int)$elder['is\_budget']);  $group->setGroupID((int)$elder['grp\_id']);    $s = new Student(  new User(  $elder['sn'],  $elder['fn'],  $elder['pt'],  $elder['email'],  $elder['paswd'],  (int)$elder['type\_user']  ),  $elder['home\_address'],  $elder['cell\_phone'],  $group  );    return $s;  } else {  return false;  }  } break;  default:  {  return false;  } break;  }  }    /\*!  \brief Производит аутентификацию администратора  \param[in] $email - Электронная почта администратора  \param[in] $passwd - Пароль  \return FALSE - пользователь не существует, объект класса User - пользователь существует  \*/    public function authentificationAdmin(string $email, string $passwd)  {  $admin = $this->query("call authentificationAdmin(:email, :passwd)", [":email" => $email, ":passwd" => $passwd])[0];    if (!empty($admin)) {  $a = new User(  $admin['sn'],  $admin['fn'],  $admin['pt'],  $admin['email'],  $admin['passwd'],  (int)$admin['type\_user']  );    return $a;  } else {  return false;  }    }    /\*!  \brief Возвращает всех администраторов  \return Администраторы  \note Массив с объектами класса User  \*/    public function getAllAdmins() : array  {  $db\_admins = $this->query("call getAllAdmins()");    $admins = array();  foreach ($db\_admins as $db\_admin) {  $admins[] = new User(  $db\_admin['sn'],  $db\_admin['fn'],  $db\_admin['pt'],  $db\_admin['email'],  $db\_admin['passwd'],  1  );  }    return $admins;  }    /\*!  \brief Возвращает всех пользователей  \return Пользователи  \note Массив с объектами класса User  \*/    public function getAllUsers() : array  {  $db\_admins = $this->query("call getAllUsers()");    $admins = array();  foreach ($db\_admins as $db\_admin) {  $admins[] = new User(  $db\_admin['sn'],  $db\_admin['fn'],  $db\_admin['pt'],  $db\_admin['email'],  $db\_admin['paswd'],  1  );  }    return $admins;  }    /\*!  \brief Возвращает всех студентов  \return Студенты  \note Массив с объектами класса Student  \*/    public function getAllStudents() : array  {  $db\_students = $this->query("call getAllStudents()");    $students = array();  foreach ($db\_students as $db\_student) {  $spec = new Specialty($db\_student['spec\_code'], $db\_student['spec\_descp'], "none");  $spec->setSpecialtyID((int)$db\_student['spec\_id']);    $group = new Group($db\_student['grp'], $spec, $db\_student['edu\_year'], (int)$db\_student['is\_budget']);  $group->setGroupID((int)$db\_student['grp\_id']);    $students[] = new Student(  new User(  $db\_student['sn'],  $db\_student['fn'],  $db\_student['pt'],  $db\_student['email'],  $db\_student['paswd'],  (int)$db\_student['type\_user']  ),  $db\_student['home\_address'],  $db\_student['cell\_phone'],  $group  );  }    return $students;  }    /\*!  \brief Возвращает всех преподавателей  \return Преподаватели  \note Массив с объектами класса Teacher  \*/    public function getAllTeachers() : array  {  $db\_teachers = $this->query("call getAllTeachers()");    $teachers = array();  foreach ($db\_teachers as $db\_teacher) {    $db\_subjects = $this->query("call getSubjects(:teacher\_email)", [ ":teacher\_email" => $db\_teacher['email']]);  $subjects = array();  foreach ($db\_subjects as $subject) {  $subjects[] = new Subject($subject['description']);  }  $teacher = new Teacher(  new User(  $db\_teacher['sn'],  $db\_teacher['fn'],  $db\_teacher['pt'],  $db\_teacher['email'],  $db\_teacher['paswd'],  (int)$db\_teacher['type\_user']  ),  $db\_teacher['info']  );    $teacher->setSubjects($subjects);  $teachers[] = $teacher;  }    return $teachers;  }    /\*!  \brief Возвращает всех родителей  \return Родители  \note Массив с объектами класса Parent\_  \*/    public function getAllParents() : array  {  $db\_parents = $this->query("call getAllParents()");    $parents = array();  foreach ($db\_parents as $db\_parent) {    $db\_childs = $this->query("call getChilds(:email)", [":email" => $db\_parent['email']]);    $childs = array();    foreach ($db\_childs as $db\_child) {  $childs[] = $this->authentification($db\_child['email'], $db\_child['passwd']);  }    $p = new Parent\_(  new User(  $db\_parent['sn'],  $db\_parent['fn'],  $db\_parent['pt'],  $db\_parent['email'],  $db\_parent['paswd'],  (int)$db\_parent['type\_user']  ),  (int)$db\_parent['age'],  $db\_parent['education'],  $db\_parent['work\_place'],  $db\_parent['post'],  $db\_parent['home\_phone'],  $db\_parent['cell\_phone']  );    $p->setChilds($childs);    $parents[] = $p;  }    return $parents;  }    /\*!  \brief Возвращает всех старост  \return Старосты  \note Массив с объектами класса Student  \*/    public function getAllElders()  {  $db\_students = $this->query("call getAllElders()");    $students = array();  foreach ($db\_students as $db\_student) {  $spec = new Specialty($db\_student['spec\_code'], $db\_student['spec\_descp'], "none");  $spec->setSpecialtyID((int)$db\_student['spec\_id']);    $group = new Group($db\_student['grp'], $spec, $db\_student['edu\_year'], (int)$db\_student['is\_budget']);  $group->setGroupID((int)$db\_student['grp\_id']);    $students[] = new Student(  new User(  $db\_student['sn'],  $db\_student['fn'],  $db\_student['pt'],  $db\_student['email'],  $db\_student['paswd'],  (int)$db\_student['type\_user']  ),  $db\_student['home\_address'],  $db\_student['cell\_phone'],  $group  );  }    return $students;  }    /\*!  \brief Возвращает всех студентов и старост  \return Студенты и старосты  \note Массив с объектами класса Student  \*/    public function getAllStudentsElders()  {  $db\_students = $this->query("call getAllStudentsElders()");    $students = array();  foreach ($db\_students as $db\_student) {  $spec = new Specialty($db\_student['spec\_code'], $db\_student['spec\_descp'], "none");  $spec->setSpecialtyID((int)$db\_student['spec\_id']);    $group = new Group($db\_student['grp'], $spec, $db\_student['edu\_year'], (int)$db\_student['is\_budget']);  $group->setGroupID((int)$db\_student['grp\_id']);    $students[] = new Student(  new User(  $db\_student['sn'],  $db\_student['fn'],  $db\_student['pt'],  $db\_student['email'],  $db\_student['paswd'],  (int)$db\_student['type\_user']  ),  $db\_student['home\_address'],  $db\_student['cell\_phone'],  $group  );  }    return $students;  }  /\*!  \brief  \return  \*/  public function getSoGroups(int $group\_id) : array  {  $sogroups = $this->query("SELECT \* FROM `v\_Students` WHERE `grp`=:grp\_id", [":grp\_id" => $group\_id]);  $elder = $this->query("SELECT \* FROM `v\_Elders` WHERE `grp`=:grp\_id", [":grp\_id" => $group\_id])[0];    if (!empty($elder)) {  $sogroups[] = $elder;  }  return $sogroups;  }    /\*!  \brief Назначает старосту  \param[in] $student\_email - Электронная почта студента  \return TRUE - успешно, FALSE - ошибка  \*/    public function grantElder(string $student\_email) : bool  {  $grant\_elder\_query = $this->dbc()->prepare("call grantElder(:email)");    $grant\_elder\_query->bindValue(":email", $student\_email);    return $grant\_elder\_query->execute();  }    /\*!  \brief Разжаловает старосту  \param[in] $student\_email - Электронная почта студента  \return TRUE - успешно, FALSE - ошибка  \*/    public function revokeElder(string $student\_email) : bool  {  $revoke\_elder\_query = $this->dbc()->prepare("call revokeElder(:email)");    $revoke\_elder\_query->bindValue(":email", $student\_email);    return $revoke\_elder\_query->execute();  }    /\*!  \brief Назначает детей родителю  \param[in] $parent\_email - Электронная почта родителя  \param[in] $student\_email - Электронная почта студента  \param[in] $relation - Тип отношения  \return TRUE - успешно, FALSE - ошибка  \*/    public function setChild(string $parent\_email, string $student\_email, int $relation)  {  $set\_child\_query = $this->dbc()->prepare("call setChild(:p\_email, :s\_email, :relation)");    $set\_child\_query->bindValue(":p\_email", $parent\_email);  $set\_child\_query->bindValue(":s\_email", $student\_email);  $set\_child\_query->bindValue(":relation", $relation);    return $set\_child\_query->execute();  }    /\*!  \brief Убирает детей от родителя  \param[in] $parent\_email - Электронная почта родителя  \param[in] $student\_email - Электронная почта студента  \return TRUE - успешно, FALSE - ошибка  \*/    public function unsetChild(string $parent\_email, string $student\_email) : bool  {  $unset\_child\_query = $this->dbc()->prepare("call unsetChild(:p\_email, :s\_email)");    $unset\_child\_query->bindValue(":p\_email", $parent\_email);  $unset\_child\_query->bindValue(":s\_email", $student\_email);    return $unset\_child\_query->execute();  }    /\*!  \brief Назначает предмет преподавателю  \param[in] $teacher\_email - Электронная почта родителя  \param[in] $subject\_id - Идентификатор предмета  \return TRUE - успешно, FALSE - ошибка  \*/    public function setSubject(string $teacher\_email, int $subject\_id) : bool  {  $set\_subject\_query = $this->dbc()->prepare("call setSubject(:t\_email, :subject\_id)");    $set\_subject\_query->bindValue(":t\_email", $teacher\_email);  $set\_subject\_query->bindValue(":subject\_id", $subject\_id);    return $set\_subject\_query->execute();  }    /\*!  \brief Убирает предмет у преподавателя  \param[in] $teacher\_email - Электронная почта родителя  \param[in] $subject\_id - Идентификатор предмета  \return TRUE - успешно, FALSE - ошибка  \*/    public function unsetSubject(string $teacher\_email, int $subject\_id) : bool  {  $unset\_subject\_query = $this->dbc()->prepare("call unsetSubject(:t\_email, :subject\_id)");    $unset\_subject\_query->bindValue(":t\_email", $teacher\_email);  $unset\_subject\_query->bindValue(":subject\_id", $subject\_id);    return $unset\_subject\_query->execute();  }    /\*!  \brief Изменяет пароль пользователя  \param[in] $user\_email - Электронная почта пользователя  \param[in] $old\_passwd - Старый пароль  \param[in] $new\_passwd - Новый пароль  \return TRUE - успешно, FALSE - ошибка  \*/    public function changeUserPassword(string $user\_email, string $old\_passwd, string $new\_passwd) : bool  {  $change\_passwod\_query = $this->dbc()->prepare("call changeUserPassword(:email, :old\_passwd, :new\_passwd)");    $change\_passwod\_query->bindValue(":email", $user\_email);  $change\_passwod\_query->bindValue(":old\_passwd", $old\_passwd);  $change\_passwod\_query->bindValue(":new\_passwd", $new\_passwd);    return $change\_passwod\_query->execute();  }    /\*!  \brief Удаляет пользователя  \param[in] $user\_email - почта пользователя  \return TRUE - успешно, FALSE - ошибка  \*/    public function remove($user\_email) : bool  {  $remove\_user\_query = $this->dbc()->prepare("call removeUser(:email)");    $remove\_user\_query->bindValue(":email", $user\_email);    return $remove\_user\_query->execute();  }    /\*!  \brief Удаляет администратора  \param[in] $admin\_email - почта пользователя  \return TRUE - успешно, FALSE - ошибка  \*/    public function removeAdmin(string $admin\_email) : bool  {  $remove\_admin\_query = $this->dbc()->prepare("call removeAdmin(:email)");    $remove\_admin\_query->bindValue(":email", $admin\_email);    return $remove\_admin\_query->execute();  }    }  ?> |
| **group.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    require\_once "specialty.class.php";    use IEP\Structures\Specialty;    /\*!  \class Group group.class.php "iep/structures/group.class.php"  \brief Класс, который несёт в себе информацию об группе  \author pmswga  \version 1.0    Класс представляет собой сущность, в которую помещаются данные из базы данных    \*/    class Group  {    private $id; ///< Идентификатор группы  private $number; ///< Номер  private $spec; ///< Специальность  private $year\_education; ///< Год обучения  private $is\_budget; ///< Тип группы (1 - бюджетная, 0 - коммерческая)    /\*!  \param[in] $number - название группы  \param[in] $spec - специальность  \note Представляет собой объект класса Specialty  \param[in] $year\_education - год выпуска  \note Указывается в формате [год]/[год]  \param[in] $is\_budget - тип группы  \note 1 - бюджетная, 0 - коммерческая  \*/    function \_\_construct(string $number, $spec, string $year\_education, int $is\_budget = 1)  {  $this->id = 0;  $this->number = $number;  $this->spec = $spec;  $this->year\_education = $year\_education;  $this->is\_budget = $is\_budget;  }    /\*!  \param[in] $method - вызываемый метод  \param[in] $args - аргументы вызываемого метода  \return Вызов существующего метода    \warning    \par  Конкретно в этом классе переопределён метод \_\_call().  Это было сделано для того, чтобы убрать "паравозик" при вызове методов из объекта    \par  Например, чтобы раньше получить код специальности, нужно было писать:  $group->getSpec()->getCode()    \par  После переопределения метода \_call() можно писать:  $group->getCode()    \par  В обоих случаях мы получаем код специальности      \*/    public function \_\_call($method, $args)  {  switch ($method)  {  case "getCode":  {  return $this->getSpec()->getCode();  } break;  }  }    /\*!  \brief Задаёт идентификатор группе  \param[in] $id - идентификатор группы  \note Идентификатор берётся из базы данных  \*/    public function setGroupID(int $id)  {  $this->id = $id;  }    /\*!  \brief Возвращает идентификатор группы  \return Идентификатор группы  \note Далее идентификатор используется для манипуляций с данными в базе данных  \*/    public function getGroupID() : int  {  return $this->id;  }    /\*!  \brief Возвращает номер группы  \return Название группы  \*/    public function getNumberGroup() : string  {  return $this->number;  }    /\*!  \brief Возвращает специальность  \return Объект типа Specialty  \*/    public function getSpec()  {  return $this->spec;  }    /\*!  \brief Возвращает год обучения  \return Год обучения  \*/    public function getYearEducation() : string  {  return $this->year\_education;  }    /\*!  \brief Возвращает тип группы  \return Тип группы  \*/    public function getStatus() : int  {  return $this->is\_budget;  }    }    ?> |
| **news.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;  /\*!    \class News news.class.php "iep/structures/news.class.php"  \brief Класс, который хранит в себе информацию об новости  \author pmswga  \version 1.0    Класс представляет собой сущность, в которую помещаются данные из базы данных    \*/    class News  {  private $id; ///< Идентификатор  private $caption; ///< Заголовок  private $content; ///< Содержание  private $author; ///< Автор  private $date\_publication; ///< Дата публикации    /\*!  \param[in] $caption - Заголовок  \param[in] $content - Содержание новости    \param[in] $author - Автор  \note Формат отображения "Фамилия И.О."    \param[in] $date\_publication - Дата публикации  \note Формат даты "d.m.Y H:i:s"  \*/    function \_\_construct(string $caption, string $content, string $author, string $date\_publication)  {  $this->caption = $caption;  $this->content = $content;  $this->author = $author;  $this->date\_publication = $date\_publication;  $this->id = 0;  }    /\*!  \brief Задаёт идентификатор новости  \param[in] $id - Идентификатор новости  \note Идентификатор из базы данных  \*/    public function setNewsID(int $id)  {  $this->id = $id;  }    /\*!  \brief Возвращает идентификатор новости  \return Идентификатор новости  \*/    public function getNewsID() : int  {  return $this->id;  }    /\*!  \brief Возвращает заголовок  \return Заголовок новости  \*/    public function getCaption() : string  {  return $this->caption;  }    /\*!  \brief Возвращает содержимое новости  \return Возвращает HTML содержимое новости  \note Для декодирования и отображения HTML на сайте используется функция html\_entity\_decode()  \*/    public function getContent()  {  return html\_entity\_decode($this->content, ENT\_HTML5);  }    /\*!  \brief Возвращает автора новости  \return Автора новости  \note Формат отображения "Фамилия И.О."  \*/    public function getAuthor() : string  {  return $this->author;  }    /\*!  \brief Возвращает дату публикации  \return Дату публикации  \note Формат даты "d.m.Y H:i:s"  \*/    public function getDatePublication() : string  {  return $this->date\_publication;  }    /\*!  \brief Задаёт заголовок  \param[in] $caption - Заголовок новости  \*/    public function setCaption($caption)  {  $this->caption = $caption;  }    /\*!  \brief Задаёт содержимое новости  \param[in] $content - Содержание новости  \*/    public function setContent($content)  {  $this->content = $content;  }    /\*!  \brief Задаёт автора новости  \param[in] $author - Автор новости  \note Формат отображения "Фамилия И.О."  \*/    public function setAuthor($author)  {  $this->author = $author;  }    /\*!  \brief Задаёт дату публикации  \param[in] $date\_publication - Дата публикации  \note Формат даты "d.m.Y H:i:s"  \*/    public function setDatePublication($date\_publication)  {  $this->date\_publication = $date\_publication;  }    }    ?> |
| **parent.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;  require\_once "user.class.php";    /\*!    \class Parent\_ parent.class.php "iep/structures/parent.class.php"  \extends User  \brief Класс для хранения данных об родителе  \author pmswga  \version 1.0    \details  Данный класс содержит в себе информацию об родителе    \*/    class Parent\_ extends User  {  private $age; ///< Возраст  private $education; ///< Образование  private $work\_place; ///< Место работы  private $post; ///< Должность  private $home\_phone; ///< Домашний телефон  private $cell\_phone; ///< Сотовый телефон  private $childs; ///< Дети    /\*!  \param[in] - $user - Объект типа класса User  \note Необходим для хранения основной информации    \param[in] - $age - Возраст  \param[in] - $education - Образование  \param[in] - $work\_place - Место работы  \param[in] - $post - Должность  \param[in] - $home\_phone - Домашний телефон  \param[in] - $cell\_phone - Сотовый телефон  \*/    function \_\_construct(  User $user,  int $age,  string $education,  string $work\_place,  string $post,  string $home\_phone,  string $cell\_phone  ) {  parent::\_\_construct($user->sn, $user->fn, $user->pt, $user->email, $user->password, $user->typeUser);  $this->age = $age;  $this->education = $education;  $this->work\_place = $work\_place;  $this->post = $post;  $this->home\_phone = $home\_phone;  $this->cell\_phone = $cell\_phone;  $this->childs = array();  }    /\*!  \brief Возвращает возраст  \return Возраст  \*/    public function getAge() : int  {  return $this->age;  }    /\*!  \brief Возвращает образование родителя  \return Образование  \note К примеру, высшее или среднее  \*/    public function getEducation() : string  {  return $this->education;  }    /\*!  \brief Возвращает место работы  \return Место работы  \*/    public function getWorkPlace() : string  {  return $this->work\_place;  }    /\*!  \brief Возвращает должность  \return Должность  \*/    public function getPost() : string  {  return $this->post;  }    /\*!  \brief Возвращает домашний телефон  \return Домашний телефон  \*/    public function getHomePhone() : string  {  return $this->home\_phone;  }    /\*!  \brief Возвращает сотовый телефон  \return Сотовый телефон  \*/    public function getCellPhone() : string  {  return $this->cell\_phone;  }    /\*!  \brief Возвращает детей  \return Массив с объектами класса Student  \*/    public function getChilds() : array  {  return !empty($this->childs) ? $this->childs : array();  }    /\*!  \brief Задаёт детей  \param[in] $childs - дети  \note Массив с объектами класса Student  \*/    public function setChilds(array $childs)  {  $this->childs = $childs;  }    }  ?> |
| **specialty.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    /\*!    \class Specialty specialty.class.php "iep/structures/specialty.class.php"  \brief Класс, который описывает Специальность  \author pmswga  \version 1.0      \*/    class Specialty  {  private $id; ///< Идентификатор  private $code; ///< Код специальности  private $description; ///< Описание  private $file; ///< Файл в формате pdf    /\*!  \param[in] $code - Код специальности  \note Формат кода специальности "XX.XX.XX", где X - число от 0 до 9    \param[in] $description - Описание специальности  \param[in] $file - Путь до файла специальности  \note Файл в формате PDF    \*/    function \_\_construct(string $code, string $description, string $file = "")  {  $this->code = $code;  $this->description = $description;  $this->file = $file;  $this->id = 0;  }    /\*!  \brief Задаёт идентификатор  \param[in] $id - Идентификатор специальности  \*/    public function setSpecialtyID(int $id)  {  $this->id = $id;  }    /\*!  \brief Возращает идентификатор специальности  \return Идентификатор специальности  \*/    public function getSpecialtyID() : int  {  return $this->id;  }    /\*!  \brief Возращает код специальности  \return Код специальности  \note Формат кода специальности "XX.XX.XX", где X - число от 0 до 9  \*/    public function getCode() : string  {  return $this->code;  }    /\*!  \brief Возращает название специальности  \return Название специальности  \*/    public function getDescription() : string  {  return $this->description;  }    /\*!  \brief Возращает файл специальности  \return Путь до файла специальности  \*/    public function getFilepath() : string  {  return str\_replace("/", "\\", $this->file);  }    /\*!  \brief Возращает имя файла специальности  \return Имя файла специальности  \*/    public function getFilename() : string  {  return basename($this->file);  }    /\*!  \brief Задаёт файл специальности  \return Задаёт путь до файла специальности  \*/    public function setFile(string $filename)  {  $this->file = $filename;  }    }    ?> |
| **student.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    require\_once "user.class.php";  require\_once "group.class.php";  require\_once $\_SERVER['DOCUMENT\_ROOT']."/iep/consts/typeusers.consts.php";  use IEP\Structures\Group;    /\*!    \class Student student.class.php "iep/structures/student.class.php"  \extends User  \brief Класс описывающий сущность Студент  \author pmswga  \version 1.0    \*/    class Student extends User  {  private $home\_address; ///< Домашний адрес  private $cell\_phone; ///< Сотовый телефон  private $group; ///< Группа    /\*!  \param[in] $user - Объект класса User  \param[in] $home\_address - Домашний адрес  \param[in] $cell\_phone - Сотовый телефон  \param[in] $group - Группа  \note Объект класса Group  \*/    function \_\_construct(User $user, string $home\_address, string $cell\_phone, $group)  {  parent::\_\_construct($user->sn, $user->fn, $user->pt, $user->email, $user->password, $user->typeUser);  $this->home\_address = $home\_address;  $this->cell\_phone = $cell\_phone;  $this->group = $group;  }    /\*!  \brief Возвращает домашний адрес  \return Домашний адрес  \*/    public function getHomeAddress() : string  {  return $this->home\_address;  }    /\*!  \brief Возвращает сотовый телефон  \return Сотовый телефон  \*/    public function getCellPhone() : string  {  return $this->cell\_phone;  }    /\*!  \brief Возвращает группу  \return Группу  \note Объект класса Group  \*/    public function getGroup()  {  return $this->group;  }    }    ?> |
| **studentanswer.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    require\_once "student.class.php";    use IEP\Structures\Student;    /\*!    \class StudentAnswer studentanswer.class.php "iep/structures/studentanswer.class.php"  \brief Класс описывает сущность - ответы студента на тест, которые отправятся на запись в бд  \author pmswga  \version 1.0    \*/    class StudentAnswer  {  private $test\_id; ///< id преподавательского теста  private $student; ///< Студент  private $answers; ///< Ответы  private $subject; ///< Предмет  private $caption; ///< Заголовок теста  private $date; ///< Дата сдачи  private $mark; ///< Оценка    /\*!  \param[in] $student - Студент, который проходил тест  \param[in] $subject - Предмет  \param[in] $caption - Заголовок теста  \param[in] $answers - Ответы  \param[in] $date - Дата сдачи  \param[in] $mark - Оценка  \*/    function \_\_construct($student, string $subject, string $caption, array $answers, string $date, int $mark)  {  $this->student = $student;  $this->subject = $subject;  $this->caption = $caption;  $this->answers = $answers;  $this->date = $date;  $this->mark = $mark;  $this->test\_id = $test\_id;  }    public function setTestID(int $test\_id)  {  $this->test\_id = $test\_id;  }  public function getTestID() : int  {  return $this->test\_id;  }  /\*!  \brief Возвращает студента  \return Студент  \note Объект класса Student  \*/    public function getStudent()  {  return $this->student;  }    /\*!  \brief Возвращает предмет теста  \return Предмет  \*/    public function getSubject() : string  {  return $this->subject;  }    /\*!  \brief Возвращает заголовок теста  \return Заголовок теста  \*/    public function getCaption() : string  {  return $this->caption;  }    /\*!  \brief Возвращает ответы студента  \return Ответы  \*/    public function getAnswers() : array  {  return $this->answers;  }    /\*!  \brief Возвращает дату сдачи теста  \return Дата сдачи теста  \*/    public function getPassDate() : string  {  return $this->date;  }    /\*!  \brief Возвращает оценку за тест  \return Оценка за тест  \*/    public function getMark() : int  {  return $this->mark;  }    }  ?> |
| **studentresult.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;  /\*!  \*/  class StudentResult  {  private $student;  private $answers;  private $mark;  private $date;  public function \_\_construct(array $student, array $answers, int $mark, string $date)  {  $this->student = $student;  $this->answers = $answers;  $this->mark = $mark;  $this->date = $date;  }  public function student() : array  {  return $this->student;  }  public function answers() : array  {  return $this->answers;  }  public function mark() : int  {  return $this->mark;  }  public function date() : string  {  return $this->date;  }  }  ?> |
| **studenttest.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    /\*!    \class StudentTest studenttest.class.php "iep/structures/studenttest.class.php"  \brief Класс описывающий сущность результатов теста, которые будут отображаться в личном кабинете  \author pmswga  \version 1.0    \*/    class StudentTest  {  private $id\_test; ///< Идентификатор теста  private $test\_author; ///< Автор теста  private $student; ///< Студент, который проходит тест  private $caption; ///< Заголовок теста  private $subject; ///< Предмет  private $date\_pass; ///< Дата сдачи  private $mark; ///< Оценка  private $answers; ///< Ответы    /\*!  \param[in] $student - Email студента  \param[in] $caption - Название теста  \param[in] $subject - Предмет  \param[in] $date\_pass - Дата сдачи  \param[in] $mark - Оценка  \*/    function \_\_construct(string $student, string $caption, string $subject, string $date\_pass, int $mark)  {  $this->student = $student;  $this->caption = $caption;  $this->subject = $subject;  $this->date\_pass = $date\_pass;  $this->mark = $mark;  $this->answers = array();  $this->id\_test = 0;  }    /\*!  \brief Задаёт идентификатор теста  \param[in] $id - Идентификатор пройденного теста  \*/    public function setTestID(int $id)  {  $this->id\_test = $id;  }    /\*!  \brief Возвращает идентификатор теста  \return Идентификатор пройденного теста  \*/    public function getTestID() : int  {  return $this->id\_test;  }  /\*!  \brief Задаёт автора теста  \param[in] $test\_author - Автор теста  \note Формат "Фамилия И.О."  \*/    public function setTestAuthor(string $test\_author)  {  $this->test\_author = $test\_author;  }    /\*!  \brief Возвращает автора теста  \return Автора теста  \note Формат "Фамилия И.О."  \*/    public function getTestAuthor() : string  {  return $this->test\_author;  }    /\*!  \brief Возвращает электронную почту студента, который прошёл тест  \return Email студента  \*/    public function getStudent() : string  {  return $this->student;  }    /\*!  \brief Возвращает название теста  \return Название теста  \*/    public function getCaption() : string  {  return $this->caption;  }    /\*!  \brief Возвращает предмет студента  \return Предмет  \*/    public function getSubject() : string  {  return $this->subject;  }    /\*!  \brief Возвращает дату сдачи теста  \return Дату сдачи теста  \*/    public function getDatePass() : string  {  return $this->date\_pass;  }    /\*!  \brief Возвращает оценку за тест  \return Оценку за тест  \*/    public function getMark() : int  {  return $this->mark;  }    /\*!  \brief Задаёт ответы студента  \param[in] $answers - массив с ответами  \note Формат массива [[question, answer], [question, answer], ...]  \*/    public function setAnswers(array $answers)  {  $this->answers = $answers;  }    /\*!  \brief Возвращает ответы студента  \return Массив с ответами  \note Формат массива [[question, answer], [question, answer], ...]  \*/    public function getAnswers() : array  {  return $this->answers;  }    }    ?> |
| **subject.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;  /\*!  \class Subject subject.class.php "iep/structures/subject.class.php"  \brief Класс, который несёт в себе информацию об предмете  \author pmswga  \version 1.0    Класс представляет собой сущность, в которую помещаются данные из базы данных    \*/    class Subject  {    private $id; ///< Идентификатор  private $description; ///< Название предмета    /\*!  \param[in] $description - Название предмета  \*/    function \_\_construct(string $description)  {  $this->id = 0;  $this->description = $description;  }    /\*!  \brief Задаёт идентификатор предмета  \param[in] $id - Идентификатор предмета  \*/    public function setSubjectID(int $id)  {  $this->id = $id;  }    /\*!  \brief Возвращает идентификатор предмета  \return Идентификатор предмета  \*/    public function getSubjectID() : int  {  return $this->id;  }    /\*!  \brief Возвращает название предмета  \return Название предмета  \*/    public function getDescription() : string  {  return $this->description;  }    }    ?> |
| **teacher.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    require\_once "user.class.php";  /\*!    \class Teacher teacher.class.php "iep/structures/teacher.class.php"  \extends User  \brief Класс описывает сущность пользователя Преподаватель  \author pmswga  \version 1.0    \*/    class Teacher extends User  {  private $info; ///< Информация о преподавтеле  private $news; ///< Новости, опубликованные преподавтелем  private $subjects; ///< Предметы, которые ведёт преподавтель  private $tests; ///< Тесты, которые создал преподавтель    /\*!  \param[in] $user - Объект класса User  \param[in] $info - Информация об преподавтеле  \note Может быть описана в формате HTML  \*/    function \_\_construct(User $user, string $info)  {  parent::\_\_construct($user->sn, $user->fn, $user->pt, $user->email, $user->password, $user->typeUser);  $this->info = $info;  $this->subjects = array();  $this->tests = array();  }    /\*!  \brief Задаёт новости, которые были опубликованны преподавтелем  \param[in] $news - Опубликованные новости  \note Массив с объектами класса News  \*/    public function setNews(array $news)  {  $this->news = $news;  }    /\*!  \brief Задаёт предметы, которые ведёт преподавтель  \param[in] $subjects - Предметы, которые ведёт преподавтель  \note Массив с объектами класса Subject  \*/    public function setSubjects(array $subjects)  {  $this->subjects = $subjects;  }    /\*!  \brief Задаёт тесты, которые создал преподавтель  \param[in] $tests - Тесты, которые создал преподавтель  \note Массив с объектами класса Test  \*/    public function setTests(array $tests)  {  $this->tests = $tests;  }    /\*!  \brief Возвращает информацию о преподавтеле  \return Информацию об преподавтеле  \note Информация может быть описана в HTML  \*/    public function getInfo() : string  {  return $this->info;  }    /\*!  \brief Возвращает новости  \return Опубликованные новости  \note Массив с объектами класса News  \*/    public function getNews()  {  return $this->news;  }    /\*!  \brief Возвращает предметы  \return Предметы, которые ведёт преподавтель  \note Массив с объектами класса Subject  \*/    public function getSubjects() : array  {  return $this->subjects;  }    /\*!  \brief Возвращает список предметов, разделённых запятой  \return Список предметов разделённых запятой  \warning Тестовый метод  \*/    public function getStrSubjects() : string  {  return implode(", ", $this->subjects);  }    /\*!  \brief Возвращает тесты  \return Созданные тесты  \note Массив с объектами класса Test  \*/    public function getTests() : array  {  return $this->tests;  }  }  ?> |
| **test.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    /\*!    \class Test test.class.php "iep/structures/test.class.php"  \brief Класс описывает сущность - тест  \author pmswga  \version 1.0    \*/    class Test  {    private $id; ///< Идентификатор теста  private $caption; ///< Заголовок теста    /\*!  \var $subject  \note Объект класса Subject  \*/    private $subject; ///< Предмет, привязанный к тесту  private $author; ///< Электронная почта преподавателя    /\*!  \var $questions  \brief Вопросы  \note Массив с объектами класса TestQuestion  \*/    private $questions; ///< Вопросы    /\*!  \var $for\_groups  \brief Группы, которые могут проходить тест  \note Массив с объектами класса Group  \*/    private $for\_groups;    /\*!  \param[in] $caption - Заголовок теста  \param[in] $subject - Предмет  \param[in] $author - Электронная почта преподавателя  \param[in] $for\_groups - Группы, которые могут проходить тест  \note Массив с объектами класса Group  \*/    function \_\_construct(string $caption, $subject, $author, array $for\_groups = array())  {  $this->id = 0;  $this->caption = $caption;  $this->subject = $subject;  $this->author = $author;  $this->questions = array();  $this->for\_groups = $for\_groups;  }    /\*!  \brief Задаёт идентификатор теста  \param[in] $id - Идентификатор теста  \*/    public function setTestID(int $id)  {  $this->id = $id;  }    /\*!  \brief Возвращает идентификатор теста  \return Идентификатор теста  \*/    public function getTestID() : int  {  return $this->id;  }    /\*!  \brief Задаёт предмет  \return Предмет  \note Объект класса Subject  \*/    public function setSubject($subject)  {  $this->subject = $subject;  }    /\*!  \brief Возвращает предмет  \return Предмет  \note Объект класса Subject  \*/    public function getSubject()  {  return $this->subject;  }    /\*!  \brief Возвращает заголовок теста  \return Заголовок теста  \*/    public function getCaption() : string  {  return $this->caption;  }    /\*!  \brief Возвращает почту преподавателя  \return Электронная почта преподавателя  \*/    public function getAuthor()  {  return $this->author;  }    /\*!  \brief Задаёт группы, которые могут проходить тест  \param[in] $groups - Группы  \note Массив с объектами класса Subject  \*/    public function setGroups(array $groups)  {  $this->for\_groups = $groups;  }    /\*!  \brief Возвращает группы, которые могут проходить тест  \return Группы  \note Массив с объектами класса Subject  \*/    public function getGroups() : array  {  return $this->for\_groups;  }    /\*!  \brief Задаёт вопросы  \param[in] $questions - Вопросы  \note Массив с объектами класса TestQuestion  \*/    public function setQuestions(array $questions)  {  $this->questions = $questions;  }    /\*!  \brief Возвращает вопросы теста  \return Вопросы  \note Массив с объектами класса TestQuestion  \*/    public function getQuestions() : array  {  return $this->questions;  }    }    ?> |
| **testquestion.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    /\*!    \class TestQuestion testquestion.class.php "iep/structures/testquestion.php"  \brief Класс описывает вопрос к тесту  \author pmswga  \version 1.0    \*/    class TestQuestion  {  private $id; ///< Идентификатор вопроса  private $question; ///< Вопрос    /\*!  \var $answers  \brief Ответы  \note Представляет собой ассоциативный массив в формате  \code  $answers = array(  [0] => array(  "id\_answer" => ,  "answer" =>  ),  ...  );  \endcode  \*/    private $answers;  private $r\_answer; ///< Правильный ответ    /\*!  \param[in] $question - Вопрос  \param[in] $r\_answer - Правильный ответ  \param[in] $answers - Ответы  \note См. формат свойства $this->answers  \*/    function \_\_construct(string $question, string $r\_answer, array $answers = array())  {  $this->id = 0;  $this->question = $question;  $this->r\_answer = $r\_answer;  $this->answers = $answers;  }    /\*!  \brief Задаёт идентификатор вопроса  \param[in] $id - Идентификатор вопроса  \*/    public function setQuestionID(int $id)  {  $this->id = $id;  }    /\*!  \brief Возвращает идентификатор вопроса  \return Идентификатор вопроса  \*/    public function getQuestionID() : int  {  return $this->id;  }    /\*!  \brief Возвращает вопрос  \return Вопрос  \*/    public function getQuestion() : string  {  return $this->question;  }    /\*!  \brief Задаёт ответы на вопрос  \return Ответы  \note Ассоциативный массив  \*/    public function setAnswers(array $answers)  {  $this->answers = $answers;  }    /\*!  \brief Возвращает все ответы на вопрос  \return Ответы на вопрос  \note Ассоциативный массив  \*/    public function getAnswers() : array  {  return $this->answers;  }    /\*!  \brief Возвращает правильный ответ на вопрос  \return Правильный ответ  \*/    public function getRAnswer() : string  {  return $this->r\_answer;  }    }    ?> |
| **trafficentry.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    /\*!    \class TrafficEntry trafficentry.class.php "iep/structures/trafficentry.class.php"  \brief Запись об посещамости  \author pmswga  \version 1.0    \*/    class TrafficEntry  {  private $id; ///< Идентификатор записи об посещамости  private $email; ///< Электронная почта студента  private $date\_visit; ///< Дата посещения  private $count\_passed\_pair; ///< Кол-во посещённых пар  private $count\_all\_pair; ///< Общее кол-во пар    /\*!  \param[in] $student\_email - Электронная почта студента  \param[in] $date\_visit - Дата посещения  \param[in] $count\_passed\_pair - Кол-во посещённых пар  \param[in] $count\_all\_pair - Общее кол-во пар  \*/    function \_\_construct(string $student\_email, string $date\_visit, int $count\_passed\_pair, int $count\_all\_pair)  {  $this->id = 0;  $this->email = $student\_email;  $this->date\_visit = $date\_visit;  $this->count\_passed\_pair = $count\_passed\_pair;  $this->count\_all\_pair = $count\_all\_pair;  }    /\*!  \brief Задаёт идентификатор записи об посещамости  \param[in] $id - Идентификатор записи об посещамости  \*/    public function setTrafficID(int $id)  {  $this->id = $id;  }    /\*!  \brief Возвращает идентификатор посещамости  \return Идентификатор посещамости  \*/    public function getTrafficID() : int  {  return $this->id;  }    /\*!  \brief Возвращает электронную почту студента  \return Электронная почта студента  \*/    public function getStudentEmail() : string  {  return $this->email;  }    /\*!  \brief Возвращает дату посещения колледжа  \return Дата посещения  \todo Возвращать сразу отформатированную строку через функцию date() в формате "d.m.Y H:i:s"  \*/    public function getDateVisit() : string  {  return $this->date\_visit;  }    /\*!  \brief Возвращает кол-во посещённых пар  \return Кол-во посещённых пар  \warning Возвращает кол-во часов, а не пар. Чтобы получить кол-во пар, нужно поделить часы на 2  \*/    public function getCountPassedPair() : int  {  return $this->count\_passed\_pair;  }    /\*!  \brief Возвращает всего кол-во пар  \return Общее кол-во пар  \warning Возвращает кол-во часов, а не пар. Чтобы получить кол-во пар, нужно поделить часы на 2  \*/    public function getCountAllPair() : int  {  return $this->count\_all\_pair;  }    }    ?> |
| **user.class.php** |
| <?php  declare(strict\_types = 1);  namespace IEP\Structures;    /\*!    \class User user.class.php "iep/structures/user.class.php"  \brief Общий класс для всех пользователей  \author pmswga  \version 1.0    \details  Данный класс содержит в себе общую информацию об пользователе    \*/    class User  {  protected $sn; ///< Фамилия  protected $fn; ///< Имя  protected $pt; ///< Отчество  protected $email; ///< Почта  protected $password; ///< Пароль    /\*!  \var $typeUser  \brief Тип пользователя  \note См. константы в typeusers.class.php  \*/    protected $typeUser;    /\*!  \param[in] $sn - Фамилия  \param[in] $fn - Имя  \param[in] $pt - Отчество  \param[in] $email - Электронная почта  \param[in] $password - Пароль    \param[in] $typeUser - Тип пользователя  \note  там их много крч....  \*/    function \_\_construct(string $sn, string $fn, string $pt, string $email, string $password, int $typeUser = 0)  {  $this->sn = $sn;  $this->fn = $fn;  $this->pt = $pt;  $this->email = $email;  $this->password = $password;  $this->typeUser = $typeUser;  }    /\*!  \brief Возвращает фамилию  \return Фамилию пользователя  \*/    public function getSn() : string  {  return $this->sn;  }    /\*!  \brief Возвращает имя  \return Имя пользователя  \*/    public function getFn() : string  {  return $this->fn;  }    /\*!  \brief Возвращает отчество  \return Отчество пользователя  \*/    public function getPt() : string  {  return $this->pt;  }    /\*!  \brief Возвращает электронную почту  \return Электронная почта пользователя  \*/    public function getEmail() : string  {  return $this->email;  }    /\*!  \brief Возвращает пароль  \return Пароль пользователя  \note Пароль захеширован с помощью MD5  \*/    public function getPassword() : string  {  return $this->password;  }    /\*!  \brief Возвращает тип пользователя  \return Тип пользователя  \note См. константы в typeusers.consts.php  \*/    public function getUserType() : int  {  return $this->typeUser;  }    }    ?> |
| **index.php** |
| <?php  require\_once "start.php";  require\_once "iep/pages/index.page.class.php";  use IEP\Structures\User;  use IEP\Pages\IndexPage;  $IndexPage = new IndexPage("Информационно-образовательный портал", "index.tpl");    if (isset($\_SESSION['user']) &&  $\_SESSION['user'] instanceof User  ) {  $IndexPage->setData("user", $\_SESSION['user']);  }    $CT->assign($IndexPage->data());  $CT->Show($IndexPage->template());    ?> |
| **news.php** |
| <?php  require\_once "start.php";    use IEP\Structures\User;    $CT->assign("news", $NM->getAllNews());    if (isset($\_SESSION['user']) &&  $\_SESSION['user'] instanceof User  ) {  $CT->assign("user", $\_SESSION['user']);  }    $CT->Show("news.tpl");    ?> |
| **regparent.php** |
| <?php  require\_once "start.php";  require\_once "engine/cform.php";    use IEP\Structures\User;  use IEP\Structures\Parent\_;    $childrens = $UM->getAllStudents();    $studentsByGroup = array();  for ($i = 0; $i < count($childrens); $i++) {  $studentsByGroup[$childrens[$i]->getGroup()->getNumberGroup()][] = $childrens[$i];  }    $CT->assign("studentsByGroup", $studentsByGroup);  $CT->Show("reg\_parent.tpl");    if (!empty($\_POST['regParentButton'])) {    if (!empty($\_POST['isAgree']) && !empty($\_POST['isMyChildren'])) {  $reg\_parent\_data = CForm::GetData(array(  "sn",  "fn",  "pt",  "email",  "password",  "home\_phone",  "cell\_phone",  "education",  "age",  "work\_place",  "post"  ));  $reg\_parent\_data['password'] = md5($reg\_parent\_data['password']);  $reg\_parent\_data['childs'] = $\_POST['childs'];    $parent = new Parent\_(  new User(  $reg\_parent\_data['sn'],  $reg\_parent\_data['fn'],  $reg\_parent\_data['pt'],  $reg\_parent\_data['email'],  $reg\_parent\_data['password'],  4  ),  (int)$reg\_parent\_data['age'],  $reg\_parent\_data['education'],  $reg\_parent\_data['work\_place'],  $reg\_parent\_data['post'],  $reg\_parent\_data['home\_phone'],  $reg\_parent\_data['cell\_phone']  );    if (!empty($reg\_parent\_data['childs'])) {  if ((count($reg\_parent\_data['childs']) <= 10) && count($reg\_parent\_data['childs']) > 0) {    $parent->setChilds($reg\_parent\_data['childs']);    if($UM->add($parent)) {  CTools::Message("Регистрация прошла успешно");  }  else {  CTools::Message("При регистрации произошла ошибка");  }    CTools::Redirect("regparent.php");  } else {  CTools::Message("Детей не может быть больше 10ти");  }  } else {  CTools::Message("Вы не можете зарегистрироваться, не выбрав детей");  }    } else {  CTools::Message("Вы должны принять соглашения");  }    }    ?> |
| **schedule.php** |
| <?php  require\_once "start.php";  require\_once "iep/pages/schedule.page.class.php";  use IEP\Structures\User;  use IEP\Pages\SchedulePage;    $SchedulePage = new SchedulePage("Расписание", "schedule.tpl");  $SchedulePage->setData("week", date("W"));  $groups = $GM->getGroupsOfCurrentYear();  if (!empty($groups)) {  if (!empty(current($groups))) {  setcookie("current\_group", current($groups)->getGroupID());  if (!empty($\_POST['selectGroupButton'])) {  $SchedulePage->callback($\_POST['selectGroupButton']);  }    if (!empty($\_COOKIE['current\_group'])) {    $SchedulePage->setData("schedules", $SHM->getScheduleGroup($\_COOKIE['current\_group']));  $SchedulePage->setData("changed\_schedules", $SHM->getChangeScheduleGroup($\_COOKIE['current\_group']));  $SchedulePage->setData("groups", $groups);    }  }  }    if (isset($\_SESSION['user']) &&  $\_SESSION['user'] instanceof User  ) {  $SchedulePage->setData("user", $\_SESSION['user']);  }    $CT->assign($SchedulePage->data());  $CT->Show($SchedulePage->template());  ?> |
| **start.php** |
| <?php  require\_once "engine/ctemplater.php";  require\_once "engine/ctools.php";  require\_once "engine/cform.php";  require\_once "engine/settings.php";    require\_once "iep/structures/user.class.php";  require\_once "iep/managers/um.class.php";  require\_once "iep/managers/sbm.class.php";  require\_once "iep/managers/nm.class.php";  require\_once "iep/managers/gm.class.php";  require\_once "iep/managers/tm.class.php";  require\_once "iep/managers/trm.class.php";  require\_once "iep/managers/shm.class.php";  require\_once "iep/consts/typeusers.consts.php";    use IEP\Managers\UserManager;  use IEP\Managers\SubjectManager;  use IEP\Managers\NewsManager;  use IEP\Managers\GroupManager;  use IEP\Managers\TestManager;  use IEP\Managers\TrafficManager;  use IEP\Managers\ScheduleManager;    $ROOT\_PATH = $\_SERVER['DOCUMENT\_ROOT'];    $CT = new CTemplater(  $ROOT\_PATH."/templates/tpl",  $ROOT\_PATH."/templates/tpl\_c",  $ROOT\_PATH."/templates/configs",  $ROOT\_PATH."/templates/cache"  );    $DB = new PDO("mysql:dbname=".DATA\_BASE\_NAME.";host=127.0.0.1", USER\_NAME, USER\_PASSWORD);  $DB->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  $DB->exec("SET NAMES utf8");    $UM = new UserManager($DB);    $SM = new SubjectManager($DB);    $NM = new NewsManager($DB);    $GM = new GroupManager($DB);    $TM = new TestManager($DB);    $TRM = new TrafficManager($DB);    $SHM = new ScheduleManager($DB);    $CT->assign("groups", $GM->getGroupsOfCurrentYear());    session\_start();  ?> |
| **teacher.php** |
| <?php  require\_once "start.php";  require\_once "iep/pages/teachers.page.class.php";    use IEP\Structures\User;  use IEP\Pages\TeachersPage;    $TeachersPage = new TeachersPage("Препоадаватели", "teacher.tpl");  $TeachersPage->setData("teachers", $UM->getAllTeachers());    if (isset($\_SESSION['user']) &&  $\_SESSION['user'] instanceof User  ) {  $TeachersPage->setData("user", $\_SESSION['user']);  }    $CT->assign($TeachersPage->data());  $CT->Show($TeachersPage->template());  ?> |
| **exit.php** |
| <?php  require\_once "../start.php";    unset($\_SESSION['user']);  CTools::Redirect("../index.php");  ?> |

|  |
| --- |
| **getInfoTest.php** |
| <?php    require\_once "../start.php";    if (!empty($\_POST['test\_id'])) {    $test = $TM->getTest((int)$\_POST['test\_id']);    include "../templates/tpl/tests/info.tpl";  } else {  echo "Произошла ошибка";  }    ?> |
| **getEmail.php** |
| <?php  require\_once "../start.php";    if(!empty($\_POST['email']) && (is\_string($\_POST['email']))) {  $result = $UM->query(  "SELECT `email` FROM `users` WHERE `email`=:email",  [":email" => htmlspecialvarchars($\_POST['email'])]  );    if(!empty($result)) echo json\_encode(true);  else echo json\_encode(false);  }  else {  CTools::Redirect($\_SERVER['HTTP\_REFERER']);  }  ?> |
| **login.php** |
| <?php  require\_once $\_SERVER['DOCUMENT\_ROOT']."/start.php";    if (!empty($\_POST['signInButton'])) {  $email = htmlspecialvarchars($\_POST['email']);  $password = md5(htmlspecialvarchars($\_POST['passwd']));    $user = $UM->authentification($email, $password);  if(!empty($user)) {  $\_SESSION['user'] = $user;  }  else {  CTools::Message("Такого пользователя не существует");  }    CTools::Redirect($\_SERVER['HTTP\_REFERER']);  }  else {  CTools::Redirect($\_SERVER['HTTP\_REFERER']);  }    ?> |

# приложение в

**Протокол системы контроля версий**

Таблица 30 – Протокол системы контроля версий

|  |  |
| --- | --- |
| **Summary** | **Description** |
| **Commits on Sep 24, 2017** | |
| [fix](https://github.com/pmswga/EDUKIT/commit/9def8db39a59a4624867ba7ccfc837821c80f224) |  |
| [new ignore files](https://github.com/pmswga/EDUKIT/commit/4c20377bb274ed39115a3a693729decd9e1ff584) |  |
| [end design](https://github.com/pmswga/EDUKIT/commit/35c4ed7ebf68b298d0162fa4b451bebdc312ee98) |  |
| new design of parent account |  |
| [new design in teacher account](https://github.com/pmswga/EDUKIT/commit/bf55bdf142822a352f567e0573a041b131161b12) |  |
| [new design](https://github.com/pmswga/EDUKIT/commit/ecd632ed0783934a922868e7ce86969bb8f7c403) |  |
| **Commits on Sep 20, 2017** | |
| [new docs](https://github.com/pmswga/EDUKIT/commit/d4eb141e29f2f4f9acdb5ef78a527707f099c111) |  |
| **Commits on Sep 18, 2017** | |
| [fast fix](https://github.com/pmswga/EDUKIT/commit/5a0e49cceb0379bdd55997a1548a34ebac36ebe6) |  |
| [капец](https://github.com/pmswga/EDUKIT/commit/ef3136e0d1477094e143901a5c2a589b71f33373) |  |
| [asfafs](https://github.com/pmswga/EDUKIT/commit/62326a771caa52359d3d0979b5bd04f877e64ca9) |  |
| **Commits on Sep 17, 2017** | |
| small fix |  |
| fix |  |
| [write documents](https://github.com/pmswga/EDUKIT/commit/e2bff22526e13604c0c918749b7de5d95a1be452) |  |
| **Commits on Sep 16, 2017** | |
| [fix](https://github.com/pmswga/EDUKIT/commit/068ddbb08725bbb746607f0a35295185339efdcd) |  |
| [fix!!!!](https://github.com/pmswga/EDUKIT/commit/916e3e2dd8477ebc5f4d03294e0af111b6f96c2e) |  |
| [asfasf](https://github.com/pmswga/EDUKIT/commit/bc34722b27aafdb5c522cab2c49a16630785c71a) |  |
| [fix](https://github.com/pmswga/EDUKIT/commit/068ddbb08725bbb746607f0a35295185339efdcd) |  |
| [fix](https://github.com/pmswga/EDUKIT/commit/068ddbb08725bbb746607f0a35295185339efdcd) |  |
| [fix](https://github.com/pmswga/EDUKIT/commit/068ddbb08725bbb746607f0a35295185339efdcd) |  |
| [fix remove news](https://github.com/pmswga/EDUKIT/commit/0d392c9fc3d3faf7a8a02702b6a4cb52dcb80407) |  |
| [fix](https://github.com/pmswga/EDUKIT/commit/068ddbb08725bbb746607f0a35295185339efdcd) |  |
| [fix](https://github.com/pmswga/EDUKIT/commit/068ddbb08725bbb746607f0a35295185339efdcd) |  |
| [some fix](https://github.com/pmswga/EDUKIT/commit/a96f2310e1457657a3027ea5c90c56d188cc404d) |  |
| [ffix](https://github.com/pmswga/EDUKIT/commit/ab1bee42d31c995bdd6e7e8473123a4b4db566c4) |  |
| install |  |
| [new](https://github.com/pmswga/EDUKIT/commit/184aeb808e41cdaa3791483b164746825b7bf019) |  |
| **Commits on Sep 3, 2017** | |
| [write ПМИ](https://github.com/pmswga/EDUKIT/commit/9300b37f43e2455fa608a533f776bbb343bf41f2) |  |
| **Commits on Sep 2, 2017** | |
| [fix 21](https://github.com/pmswga/EDUKIT/commit/79d92b33267fc807b7b1bfe4d062c18fbe3bf306) |  |
| [fix 20](https://github.com/pmswga/EDUKIT/commit/c7a8a0beb93b5e6a1e388febe18e03e41eae6034) |  |
| **Summary** | **Description** |
| fix 19 |  |
| [fix 18](https://github.com/pmswga/EDUKIT/commit/d8dbdb6e089debe0d0a459d1eab6f63449d33c11) |  |
| **Commits on Aug 29, 2017** | |
| [fix 17](https://github.com/pmswga/EDUKIT/commit/4653e6632ace4843b1c398d970c3f81cd8d3fdb0) |  |
| [fix 16](https://github.com/pmswga/EDUKIT/commit/918fe1810d2936fc75cdd1af0e9af5330ec8b6aa) | Реализована поддержка чётности/нечётности недели |
| **Commits on Aug 25, 2017** | |
| [fix 15](https://github.com/pmswga/EDUKIT/commit/6e0dbfa62edbeb51eab6c082554c409cc84e5282) |  |
| [fix 14](https://github.com/pmswga/EDUKIT/commit/c5266e7d4b99db86cb41ac3a5349eed725de4e44) |  |
| [fix 13](https://github.com/pmswga/EDUKIT/commit/c20b204b119cad700e73c2d17424bf7342339b58) |  |
| [fix 12](https://github.com/pmswga/EDUKIT/commit/d87a4a6d41c1464fbca7c4ef939057a617f4e458) |  |
| **Commits on Aug 24, 2017** | |
| [fix 11](https://github.com/pmswga/EDUKIT/commit/49c1566793346225d2fd6476020aa4946cb24023) |  |
| [fix 10](https://github.com/pmswga/EDUKIT/commit/60411e32e52158388bc46405b456ae04250fab64) |  |
| **Commits on Aug 23, 2017** | |
| [fix 9](https://github.com/pmswga/EDUKIT/commit/d48c47fe7d967c6fdfacf74dd99d25a338d0767e) |  |
| [fix 8](https://github.com/pmswga/EDUKIT/commit/bcb7372755a7a91b5c3cd6463eba76e65776731e) |  |
| [fix 7](https://github.com/pmswga/EDUKIT/commit/ddd24132eb060fb36b0c8c2d66ecf35bf08ae5ff) |  |
| **Commits on Aug 22, 2017** | |
| [fix 6](https://github.com/pmswga/EDUKIT/commit/5c0609978ac0fb792ef19320ff829db317858d74) |  |
| **Commits on Aug 21, 2017** | |
| [fix 5](https://github.com/pmswga/EDUKIT/commit/29f27a1e912c1ed289ed917cf609fb7088363f8a) |  |
| [fix 4](https://github.com/pmswga/EDUKIT/commit/5876f90d1687897f1073d5b06cecbaa26df621eb) |  |
| **Commits on Aug 20, 2017** | |
| [fix 3](https://github.com/pmswga/EDUKIT/commit/93fbbdc7563766bcd73fe638fff53d4cfd931c2f) | add description for classes:  - User  - Group |
| [fix 2](https://github.com/pmswga/EDUKIT/commit/1816de0ea2155c2dbe77b6bfbb92da71be705db9) |  |
| [fix 1](https://github.com/pmswga/EDUKIT/commit/5b9adbd1e08742972bcd7ebda24cc15371396b24) |  |
| **Commits on Aug 19, 2017** | |
| [write documents!!!](https://github.com/pmswga/EDUKIT/commit/0981ac9e756469ede37aa9fbebccf283f9dd3aaf) |  |
| **Commits on Aug 18, 2017** | |
| [small fixed](https://github.com/pmswga/EDUKIT/commit/65cf78751b367475813b395c7331365a084fd455) |  |
| **Commits on Aug 17, 2017** | |
| [fix](https://github.com/pmswga/EDUKIT/commit/a768aebf8ad2556f7c65699c5dbd8d4dc1e74d5b) |  |
| **Commits on Aug 15, 2017** | |
| [Выполняем требования заказчика](https://github.com/pmswga/EDUKIT/commit/58efa4557984a3053b3c38a50abc7baa1133abdd) |  |
| **Commits on Aug 14, 2017** | |
| [fix some bugs](https://github.com/pmswga/EDUKIT/commit/e22fe4f58b0f9cf23c79a370fc8909259fe69a99) | - view admins news  - view traffic of selected student  - small fixed |
| **Summary** | **Description** |
| **Commits on Aug 12, 2017** | |
| [New](https://github.com/pmswga/EDUKIT/commit/dd9d5fdaad31c8fcc48d44575356f69d1993f9b3) |  |
| **Commits on Aug 8, 2017** | |
| [add traffic view for admin](https://github.com/pmswga/EDUKIT/commit/5be147cab4830b820a4c105f346eb10f185d0929) |  |
| [small fixed](https://github.com/pmswga/EDUKIT/commit/7b412651236bd89e7d169b856548bc9b85647759) | 1) Correct information messages  2) Create traffic page of students |
| **Commits on Aug 7, 2017** | |
| new features...... |  |
| [свчпаасвмт](https://github.com/pmswga/EDUKIT/commit/445411523afaa90fca867eb62e28ad3b098a1281) |  |
| [говно а не кп](https://github.com/pmswga/EDUKIT/commit/2cbd6e04c94f0da7132806ba5ac060bc92843471) |  |
| **Commits on Aug 5, 2017** | |
| [new](https://github.com/pmswga/EDUKIT/commit/14a1691b4267f1008eff4ea6f69fd2c7a23e6c49) |  |
| **Commits on Aug 3, 2017** | |
| Fgasfa |  |
| [new](https://github.com/pmswga/EDUKIT/commit/b1a6f178018346636e4576bca3fbffe39a9e2f2b) |  |
| **Commits on Jul 31, 2017** | |
| [support news controller for admin](https://github.com/pmswga/EDUKIT/commit/74a2d34cacf1c452ec6dd41cc0d9aed04def996e) |  |
| [small fixed](https://github.com/pmswga/EDUKIT/commit/1cd5aa57108b5be292ec40cb0f4c120bb1a74660) |  |
| **Commits on Jul 30, 2017** | |
| [small fixed](https://github.com/pmswga/EDUKIT/commit/7146db77ec69ac8a33c16f3e1360a1b95cd10fa5) |  |
| [Parents control panel is ready!](https://github.com/pmswga/EDUKIT/commit/4aa4cafcd53078f9330c4b76853ffd12353ca5bb) |  |
| **Commits on Jul 27, 2017** | |
| [TEST SYSTEM IS FINISHED!!!](https://github.com/pmswga/EDUKIT/commit/684fe7c92c8af98710c770e05d38f9b42ff8d4a4) |  |
| **Commits on Jul 25, 2017** | |
| [all cool](https://github.com/pmswga/EDUKIT/commit/e0297ee7711fcce418ad81e204676b1b3184e1a3) |  |
| **Commits on Jul 24, 2017** | |
| [new features](https://github.com/pmswga/EDUKIT/commit/f2f7d76c019a9938e07f776550890e60d175bb6e) |  |
| **Commits on Jul 23, 2017** | |
| [FIXXXXXXXXXXX](https://github.com/pmswga/EDUKIT/commit/7885acf6bf0594043804288952ac594b098c7d76) |  |
| Fix small bugs |  |
| **Commits on Jun 20, 2017** | |
| [new features](https://github.com/pmswga/EDUKIT/commit/5487f5d46838d93598fbdf6b487109ce235b0a85) |  |
| **Commits on Jun 19, 2017** | |
| [fix](https://github.com/pmswga/EDUKIT/commit/6d64d8742300e4e10b15640e828998853dc4e25b) |  |
| **Commits on Jun 15, 2017** | |
| [asfsa](https://github.com/pmswga/EDUKIT/commit/7a3f1697a37b9e7fdb5b52905596abf43c7c131f) |  |
| [add log functions](https://github.com/pmswga/EDUKIT/commit/e02eff48c0b5550f998ecf07a049d0344b1449ce) |  |
| **Commits on Jun 14, 2017** | |
| new features | add logger windows for overview logs in a system |
| **Commits on Jun 13, 2017** | |
| [new features](https://github.com/pmswga/EDUKIT/commit/a2caaf3844af93e185c4a4a3d30518266933973c) |  |
| **Commits on May 22, 2017** | |
| **Summary** | **Description** |
| [Small fixed](https://github.com/pmswga/EDUKIT/commit/05779dc14ad67ab6b3417664650304ae78f25efb) |  |
| [pzdz](https://github.com/pmswga/EDUKIT/commit/9302729929d8060e0ae456bffc2c65015b5e8547) |  |
| **Commits on May 21, 2017** | |
| [Документация и прочие](https://github.com/pmswga/EDUKIT/commit/393a97b47378c1e19160c4bdb042cb450eeb613e) |  |
| [Пишем документацию](https://github.com/pmswga/EDUKIT/commit/9a811c88e1aa3a20b8f084d66f8659e45123d3c8) |  |
| [Out of range](https://github.com/pmswga/EDUKIT/commit/a8132f9edbc900ccc8d37ca3e38bc98d4ee25549) |  |
| [бя бя ббя](https://github.com/pmswga/EDUKIT/commit/ed3bc2f59a07a5dea5507294193398f39f9c4726) |  |
| **Commits on May 20, 2017** | |
| [Реализация прохождения тестов](https://github.com/pmswga/EDUKIT/commit/0973a9e1145e1a3750f71c4084f385a8ff011078) |  |
| [Implement of tests!!!](https://github.com/pmswga/EDUKIT/commit/5f089e1a83d3982377a4abe4c322c39f2f74535a) |  |
| **Commits on May 19, 2017** | |
| [Work on test](https://github.com/pmswga/EDUKIT/commit/de6511d702f98153f8223bff4ebea9750e0b1178) |  |
| [write triggers](https://github.com/pmswga/EDUKIT/commit/ab7b8e70a8f0a11b47b1d4eba8b0384f00441d4d) |  |
| **Commits on May 18, 2017** | |
| [add triggers](https://github.com/pmswga/EDUKIT/commit/aa1ab56abe8a0c3a3f1c4309409684fe39757bbe) |  |
| **Commits on May 16, 2017** | |
| [Work on teacher panel](https://github.com/pmswga/EDUKIT/commit/d884e82b049a386d0f46fb54725abc950b2c2edb) |  |
| [Add triggers for tables](https://github.com/pmswga/EDUKIT/commit/73870de3a388ad72da8b4b9d0313a31e66d5c0cf) |  |
| **Commits on May 14, 2017** | |
| [Work on the tests](https://github.com/pmswga/EDUKIT/commit/85d7ba1734e74f367d584da98244961cfd9273b0) |  |
| **Commits on May 12, 2017** | |
| [Implement of student traffic control](https://github.com/pmswga/EDUKIT/commit/cc499ae05553eb73adeba4153c13654576ffdbc6) |  |
| [Implement of elder control panel](https://github.com/pmswga/EDUKIT/commit/3a7fe099889fbaed2823ec9e9b44322631c4dd13) | Add new features:  - control student traffic |
| **Commits on May 11, 2017** | |
| [Implementation of elder control panel](https://github.com/pmswga/EDUKIT/commit/01db025314284bfce4eda2d5ec1319fe42745976) |  |
| **Commits on May 9, 2017** | |
| [Implementation parent control panel](https://github.com/pmswga/EDUKIT/commit/cce33d8e89e58ebebbfa1e4a9d70a0eb09765347) |  |
| [Implementation of importants managers](https://github.com/pmswga/EDUKIT/commit/e6bb7630fc3e1f64115f6ee3777a0b2693ce86a0) |  |
| **Commits on May 8, 2017** | |
| [implementation TestManager](https://github.com/pmswga/EDUKIT/commit/9721f8eb87f8ac9e992d0d7cedd46ebe7aa19937) | !!!!!! |
| [Full implemented of NewsManager](https://github.com/pmswga/EDUKIT/commit/a4de251952c22a267dd749dc0b8cab2329201054) | Next methods are implemented:  - add  - remove  - getAllNews()  - changeCaptionNews()  - changeContentNews() |
| [SubjectManager is full implemented](https://github.com/pmswga/EDUKIT/commit/a7abbe496be7a2cea18b7e2b568565c6dee542c3) | The following methods were implemented:  - add  - remove  - getAllSubjects()  - changeDescriptionSubject() |
| **Summary** | **Description** |
| [Implement of UserManager](https://github.com/pmswga/EDUKIT/commit/c9b202eb14c3ac9eff984133a3e2e65d86256dc8) |  |
| **Commits on May 7, 2017** | |
| [GroupManager is full implemented](https://github.com/pmswga/EDUKIT/commit/6c6fd1bc0f77d751c1b37a0473bee118cf06908c) | Next methods is implemented:  - add  - remove  - getAllGroups  - changeDescriptionGroup  - changeSpecGroup  - upCourse |
| [SpecialtyManager is full implemented](https://github.com/pmswga/EDUKIT/commit/2e5dc538d76f2c1f81d3ae3afd665e4d8c3d85b0) | The following methods were implemented:  - add  - remove  - changeCodeSpecialty  - changeDescpSpecialty  - changeFileSpecialty |
| [Немного, всё по новому сделаем](https://github.com/pmswga/EDUKIT/commit/9d779981a8215e2d81524681d53d8520c35cd29d) |  |
| **Commits on May 2, 2017** | |
| [К чёрту логи](https://github.com/pmswga/EDUKIT/commit/cb571ba5cf9d26c5a284f5e864993eb95b11d93a) |  |
| **Commits on May 1, 2017** | |
| Уточнение БД |  |
| [Ещё много работы](https://github.com/pmswga/EDUKIT/commit/c1e9bac5d1b62944be12d0a607ab0ae39db7e1eb) |  |
| **Commits on Apr 27, 2017** | |
| [Backup](https://github.com/pmswga/EDUKIT/commit/d1c17d36efb78a8b3228be85532d11e5fcda2412) |  |
| **Commits on Apr 25, 2017** | |
| [Писать комиты на русском - это стыд](https://github.com/pmswga/EDUKIT/commit/066d2949e7e128fda85fa2a030cb155ccd48b6cf) |  |
| **Commits on Apr 24, 2017** | |
| [Исправлен баг](https://github.com/pmswga/EDUKIT/commit/b8e85efb1cec49df7f246e6d748ab83c23a33b04) | - Отображение тестов |
| [Работа над системой прохождения теста](https://github.com/pmswga/EDUKIT/commit/356aa3bd795f325fea7e80373531784df0f6a2f3) |  |
| [Работа над кабинетом студента](https://github.com/pmswga/EDUKIT/commit/d0fde97c99e0656baa913cec195c1bd74b09e507) |  |
| [Работа с тестами](https://github.com/pmswga/EDUKIT/commit/f872d3944759e545f5d4a5e70f441185112d72ce) |  |
| **Commits on Apr 21, 2017** | |
| [Работа над панель преподавателя](https://github.com/pmswga/EDUKIT/commit/e2a37d5b11a7b8b8665e96c89386450c8692468c) |  |
| **Commits on Apr 19, 2017** | |
| [Фиксы](https://github.com/pmswga/EDUKIT/commit/36393473a8f39f9bf28029b49bc5f58a5ad2037b) |  |
| **Commits on Apr 17, 2017** | |
| [Реализация TestManager](https://github.com/pmswga/EDUKIT/commit/b27a54850244ca4bc60bfd3bafbbcd9d07cd87ad) |  |
| [Найстройки подключения](https://github.com/pmswga/EDUKIT/commit/36c43ade907a42cfe6a0545a1e038d8898d64759) |  |
| **Commits on Apr 16, 2017** | |
| [Работа над панелью преподавателя](https://github.com/pmswga/EDUKIT/commit/6650c2c381a1536c9ddb37d1319e85f20e720875) |  |
| **Commits on Apr 11, 2017** | |
| [Управление администраторами](https://github.com/pmswga/EDUKIT/commit/17bc252ef7f2a134b1254db006c2b1322801c79d) |  |
| [Работа над панелью администратора](https://github.com/pmswga/EDUKIT/commit/79c45544fa689e529e454085b99b8c599aa9cfdf) |  |
| **Summary** | **Description** |
| **Commits on Apr 10, 2017** | |
| [Создание отчётов](https://github.com/pmswga/EDUKIT/commit/ba7267b6a1c4b8cf38e83e43c9c2c01e0e1ccc23) |  |
| [Добавление системы создания отчётов](https://github.com/pmswga/EDUKIT/commit/5162580bbe5bcdc08b36484cd6b414785ae17242) |  |
| **Commits on Apr 9, 2017** | |
| [Система оповещений](https://github.com/pmswga/EDUKIT/commit/835fdd16aa48681e477a59c57e8f20eb0ace1890) | Реализована отправка |
| Работа над системой оповещения |  |
| **Commits on Apr 6, 2017** | |
| [Работа над панелью администратора](https://github.com/pmswga/EDUKIT/commit/8e53c8e77438523c4126d57d7125c0848f6ccd7b) |  |
| [Обновление презентации](https://github.com/pmswga/EDUKIT/commit/94dff744bd33a7563e381a785c5ae018eb15c2a3) |  |
| **Commits on Apr 4, 2017** | |
| [Обновление представления v\_Specialtyes](https://github.com/pmswga/EDUKIT/commit/cc33f1310d63432c376c764df941e11388664eef) | Данное исправление позволило исправить несколько багов |
| [Исправлены баги](https://github.com/pmswga/EDUKIT/commit/4d47cb518ee22a71a44ef44fc8d182629b2f9470) |  |
| **Commits on Apr 2, 2017** | |
| [Работа над панелью преподавателя](https://github.com/pmswga/EDUKIT/commit/44c668de5db5baf92e55eadf76fa341870e60a56) |  |
| **Commits on Mar 24, 2017** | |
| [Работа над панелью преподавателя](https://github.com/pmswga/EDUKIT/commit/6f8916a23b53d26aa93cf3ae76e638f648dd6246) |  |
| [Панель преподавателя](https://github.com/pmswga/EDUKIT/commit/b277c38eb7d0c657395a9a5e2df2561eb3e1b7da) |  |
| [Работа над новостной лентой](https://github.com/pmswga/EDUKIT/commit/6a2598649b7226ed2c5dde19fe48aa4f4ceff5fb) |  |
| [Личный кабинет преподавателя](https://github.com/pmswga/EDUKIT/commit/cc97f86b67e24293827a764305ab13beebfc1eb4) | - Реализовано добавление тестов |
| [Лажа](https://github.com/pmswga/EDUKIT/commit/d716709eea6bd2e177e66124064f81cd100f2aa5) |  |
| [Работа над личным кабинетом преподавателя](https://github.com/pmswga/EDUKIT/commit/b687018591bc3586aa3692961d3ea2f372776c58) |  |
| **Commits on Mar 20, 2017** | |
| [Create README.md](https://github.com/pmswga/EDUKIT/commit/254389d928a2927a1730188ce9fe3352c4bef247) |  |
| [Личный кабинет](https://github.com/pmswga/EDUKIT/commit/ec59e8b77d8577924cbe0a5def93c644ff62dfda) |  |
| [Панель преподавателя](https://github.com/pmswga/EDUKIT/commit/6f6cd2a6a171ae8668c9c494b8324c460e2ffbf8) |  |
| [Личные кабинеты](https://github.com/pmswga/EDUKIT/commit/759b21ab48e60908b8311586c0985c4d3f47a451) |  |
| [Новостная лента](https://github.com/pmswga/EDUKIT/commit/8405a5ab60cef43a705572d36e66d5c1aa01f3ac) |  |
| [Личный кабинет преподавателя](https://github.com/pmswga/EDUKIT/commit/70838ad29ccf91f0ae6c05b09652c1f62d2ba52b) |  |
| **Commits on Mar 19, 2017** | |
| [Статистика](https://github.com/pmswga/EDUKIT/commit/3fba4b3bda910ecb40bb81ffd403cfacd3fd68b1) |  |
| [Аутентификация пользователей](https://github.com/pmswga/EDUKIT/commit/0eab366f847924fe9d74cff539b52afa42d7c412) |  |
| [Работа над панелью администратора](https://github.com/pmswga/EDUKIT/commit/de437edbf37a6ec351b516f0468dc73bf26896b2) |  |
| **Commits on Mar 18, 2017** | |
| [Работа с пользователями](https://github.com/pmswga/EDUKIT/commit/c6cda24d061f451b2b14d1a70843cbb2d2615c43) |  |
| [Работа с пользователями](https://github.com/pmswga/EDUKIT/commit/763e77bfbfa06940efe36bc58325ce57ea3fe002) |  |
| **Commits on Mar 17, 2017** | |
| [Исправление процедур](https://github.com/pmswga/EDUKIT/commit/aaac178f84ec6194cad9d2d9cf7f99675b0de09b) |  |
| [Работа с пользователями](https://github.com/pmswga/EDUKIT/commit/aedaf0cc2928e3517d6c9c766ec76c193ab4f514) |  |
| [Работа с пользователями](https://github.com/pmswga/EDUKIT/commit/e8011e7d8a170fd1e022702450733db140067d31) |  |
| **Commits on Mar 16, 2017** | |
| [Работа с предметами](https://github.com/pmswga/EDUKIT/commit/311c03b8fffded61ff033080475abe75b247764d) |  |
| **Summary** | **Description** |
| [Работа с предметами](https://github.com/pmswga/EDUKIT/commit/23d06f71cda04d652a915e2ee86f0adc443146a1) |  |
| [Работа с группами и предметами](https://github.com/pmswga/EDUKIT/commit/d933e7cd285f691ff93080ca5e70dddb492aec7d) |  |
| [Работа со специальностями](https://github.com/pmswga/EDUKIT/commit/02cf1c0c8ecde88d60cdbb1c45d4a83cbb1c0c00) |  |
| [Tables, procedures и все все все](https://github.com/pmswga/EDUKIT/commit/139a735411e433cb5e52537492af83256e3f12ed) |  |
| **Commits on Mar 14, 2017** | |
| Процедуры |  |
| Процедуры |  |
| **Commits on Mar 13, 2017** | |
| Уточнение и изменение |  |
| Procedures |  |
| Таблицы, процедуры и все всё всё |  |
| Работа над процедурами |  |
| **Commits on Mar 8, 2017** | |
| Корректировка БД |  |
| Таблицы, представления, процедуры |  |
| Новые представления |  |
| Создание пользователей |  |
| **Commits on Mar 7, 2017** | |
| Незначительные исправления |  |
| **Commits on Feb 26, 2017** | |
| Добавлена логическая модель |  |
| SQL | Реализованы основные хранимые процедуры |
| Ограничения, таблицы и процедуры |  |
| Процедуры и таблицы |  |
| Уточнение |  |
| Ограничения |  |
| **Commits on Feb 24, 2017** | |
| Уточнение и дополнение |  |
| **Commits on Feb 20, 2017** | |
| ПМИ | Уточнение и дополнение |
| ПМИ |  |
| **Commits on Feb 19, 2017** | |
| Все файлы сюда! |  |
| Работа над SQL | Сделано разбиение на файлы:  - tables (создаёт таблицы)  - constraints (создаёт ограничения)  - views (создаёт представления)  - procedures (создаёт процедуры для работы с БД)  - users (создаёт пользователей для доступа к БД) |
| **Commits on Feb 15, 2017** | |
| **Summary** | **Description** |
| Мелкие исправления |  |
| Исправления | - Добавление групп  - Отображения списка предметов у преподавателя |
| **Commits on Feb 11, 2017** | |
| Исправление багов |  |
| Глобальные изменения | - Fix ошибок  - Реализована отправка на почту родителям |
| Commits on Feb 8, 2017 |  |
| Полностью переписан скрипт проверки данных при регистрации | - Убрана "Дата рождения" |
| Незначительные изменения | - Добавлена вкладка "Оповещения"  - Изменена форма регистрации студента (так же изменнён скрипт  регистрации) |
| **Commits on Feb 3, 2017** | |
| Мелкие исправления |  |
| **Commits on Feb 2, 2017** | |
| Работа над UserManager | Реализована функция удаления пользователя  Реализована функция getTeachers() |
| Работа над UserManager | Реализован getParents() |
| Улучшение структуры класса User | Добавлена проверка типов |
| Работа над UserManager | - Реализовано добавление администратора  - Реализованы методы getUsers() и getStudents() |
| Работа над UserManager | Реализовано добавления родителя |
| Работа над UserManager | Реализовано добавления преподавателя |
| Работа на архитектурой | - Исправления в класса Structures  - Добавления пользователей реализовано через транзакции  - Мелкие исправления |
| **Commits on Jan 31, 2017** | |
| Работа над менеджером | Добавлен метод removeQuestion и мелки исправления |
| Работа над структурой |  |
| **Commits on Jan 29, 2017** | |
| Переезд с Bitbucket |  |
| Added .gitattributes & .gitignore files |  |

Таблица – Протокол ошибок

|  |  |  |
| --- | --- | --- |
| **№** | **Описание** | **Статус** |
| 1 | Предоставление информации об преподавателях | complete |
| 2 | Написать руководство | complete |
| 3 | Вставить диаграммы классов | complete |
| 4 | Нарисовать диаграммы классов | complete |
| 5 | Редактирование своего профиля: | complete |
| 6 | Добавление вопросов к тесту | complete |
| 7 | Просмотр информации об своих детях: | complete |
| 8 | Просмотр информации своей посещаемости | complete |
| 9 | Прохождение доступного теста | complete |
| 10 | Просмотр информации об своём расписании | complete |
| 11 | Просмотр результатов пройденных тестов | complete |
| 12 | Дописать протокол использования системы контроля версий | complete |
| 13 | Отметка посещаемости | complete |
| 14 | Создание тестов | complete |
| 15 | Удаление теста | complete |
| 16 | Удаление вопроса | complete |
| 17 | Добавление новостей | complete |
| 18 | Просмотр новостной ленты | complete |
| 19 | Регистрация/аутентификация | complete |
| 20 | Просмотр чужого профиля студента | complete |
| 21 | Добавление нового пользователя | complete |
| 22 | Назначение старосты в группе | complete |
| 23 | Просмотр студентов в группах | complete |

# ПРИЛОЖЕНИЕ Г

**РУКОВОДСТВО ПРОГРАММИСТА**

Руководство содержит следующие основные разделы:

* Титульная страница (Рисунок 19);
* Описания (Рисунок 20);
* Пространства имён (Рисунок 24);
* Структуры данных (Рисунок 25);
* Файлы (Рисунок 26).

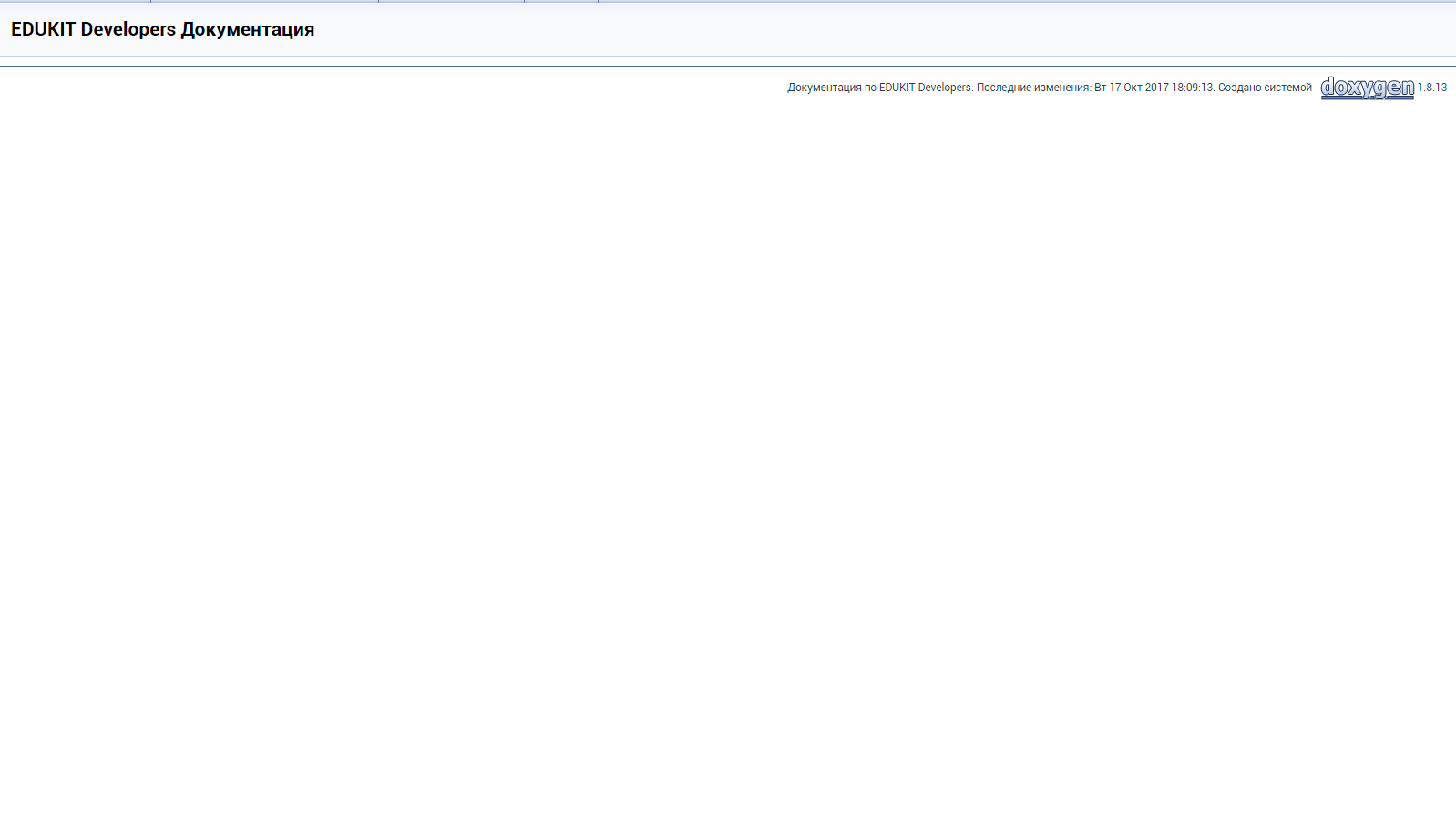


Рисунок 19 – Титульная страница

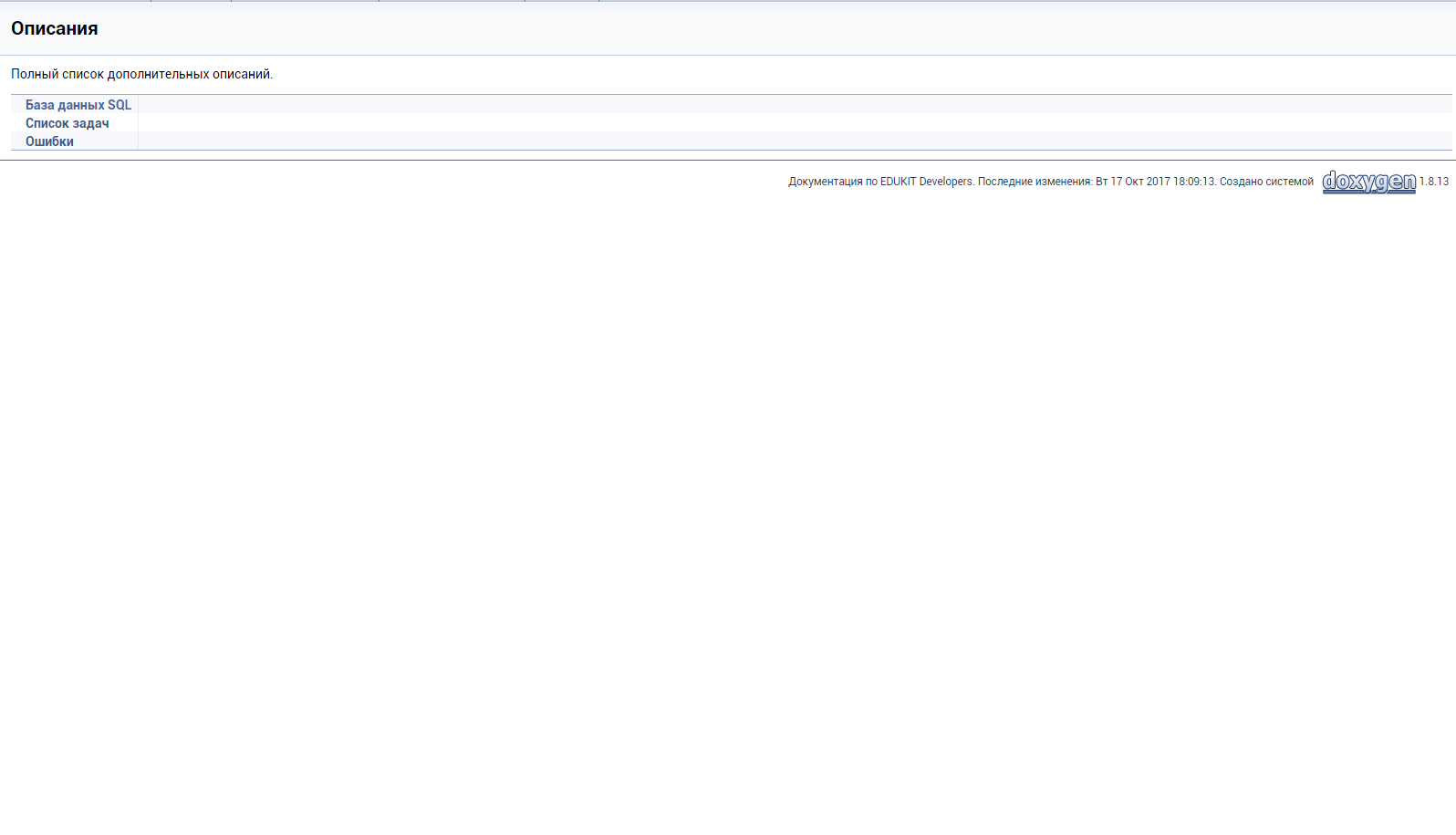


Рисунок 20 – Описания

В данном разделе приводятся ссылки на следующие страницы:

* База данных SQL (Рисунок 21);
* Список задач (Рисунок 22);
* Ошибки (Рисунок 23).

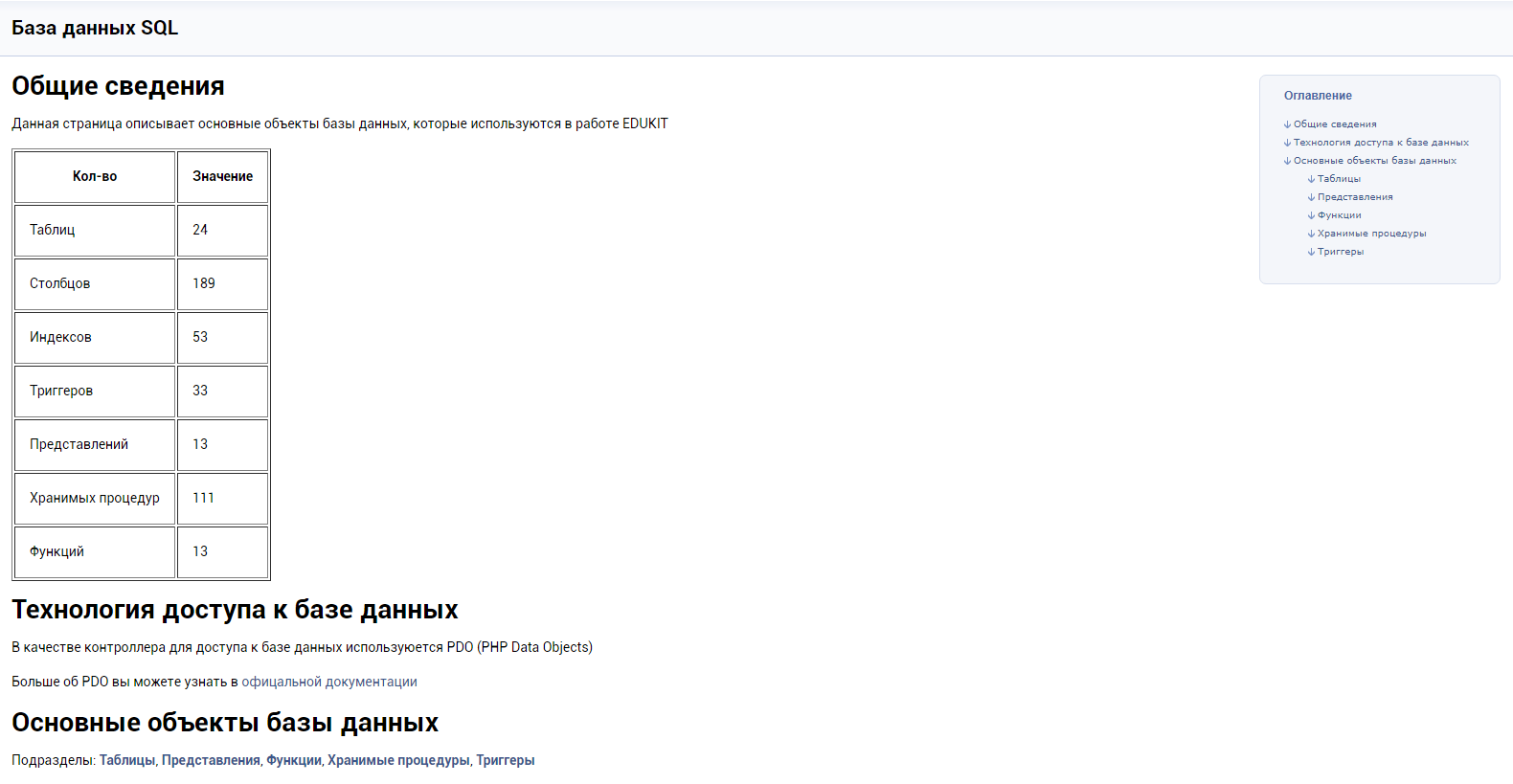


Рисунок 21 – Страница «База данных SQL»

На данной странице описаны сведения об объектах базы данных.

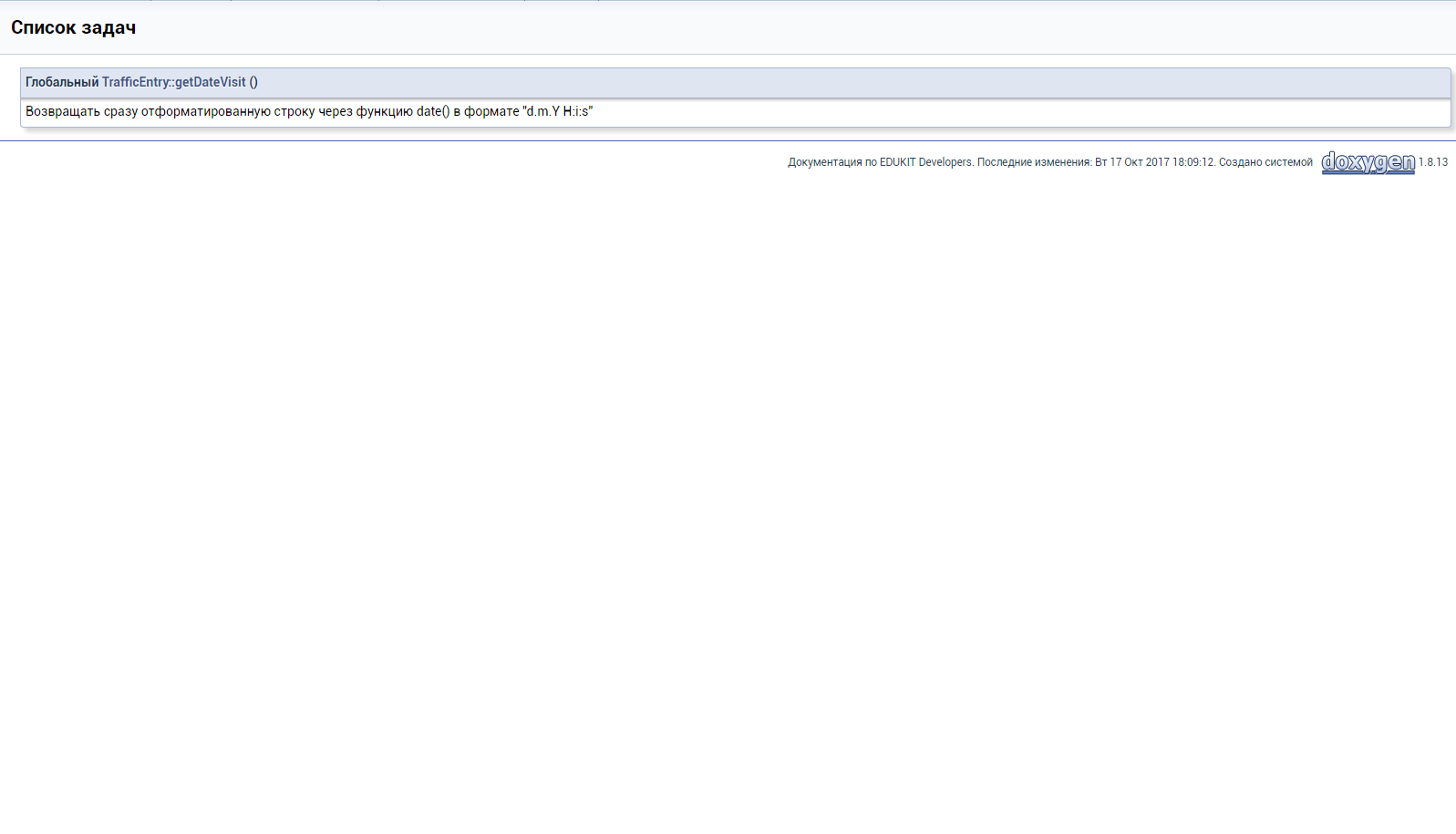


Рисунок 22 – Страница «Список задач»

На данной странице приводится список задач, которые относятся к исходному коду.

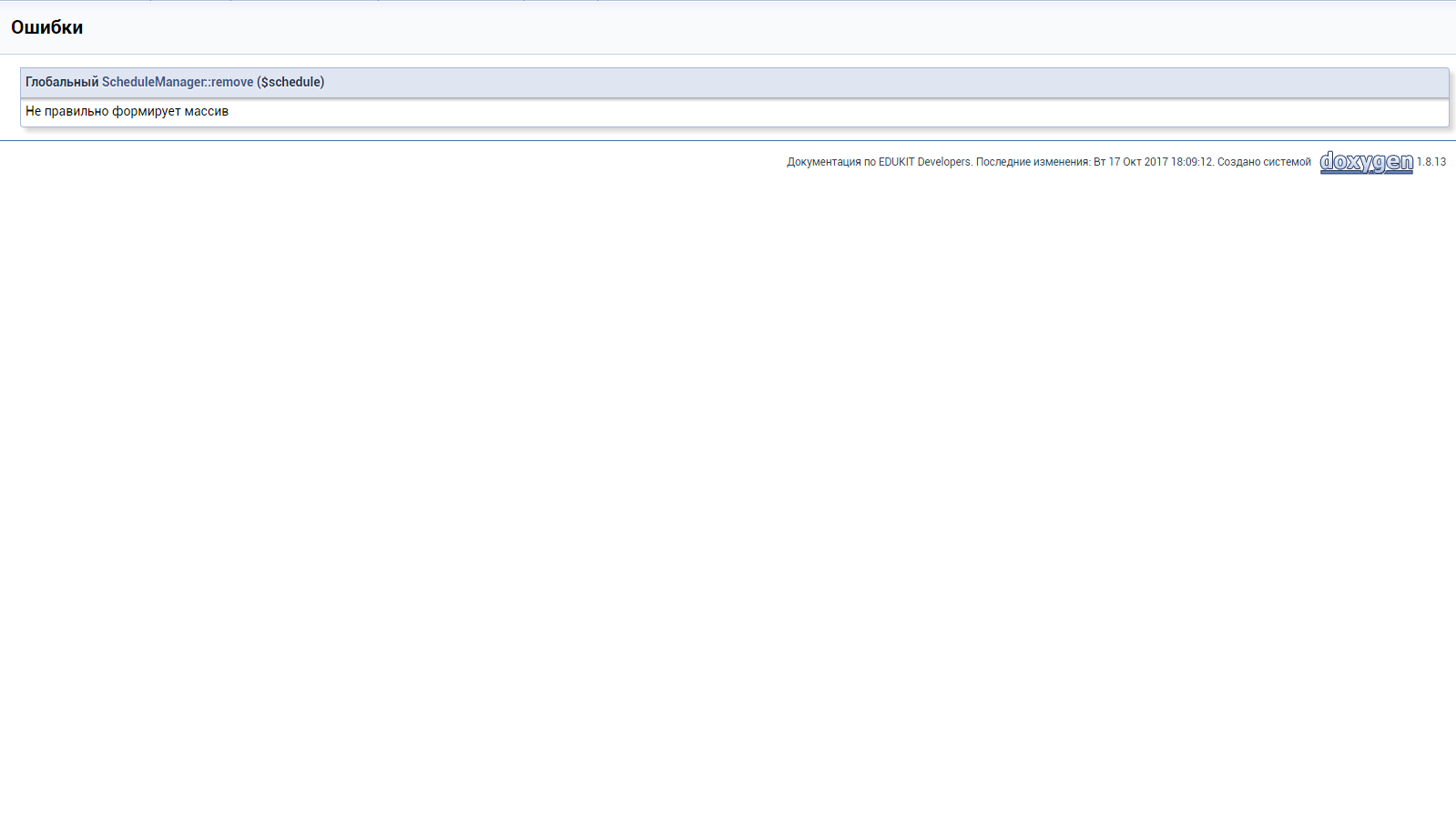


Рисунок 23 – Страница «Ошибки»

На данной странице приводится список ошибок, которые найдены в исходном коде.

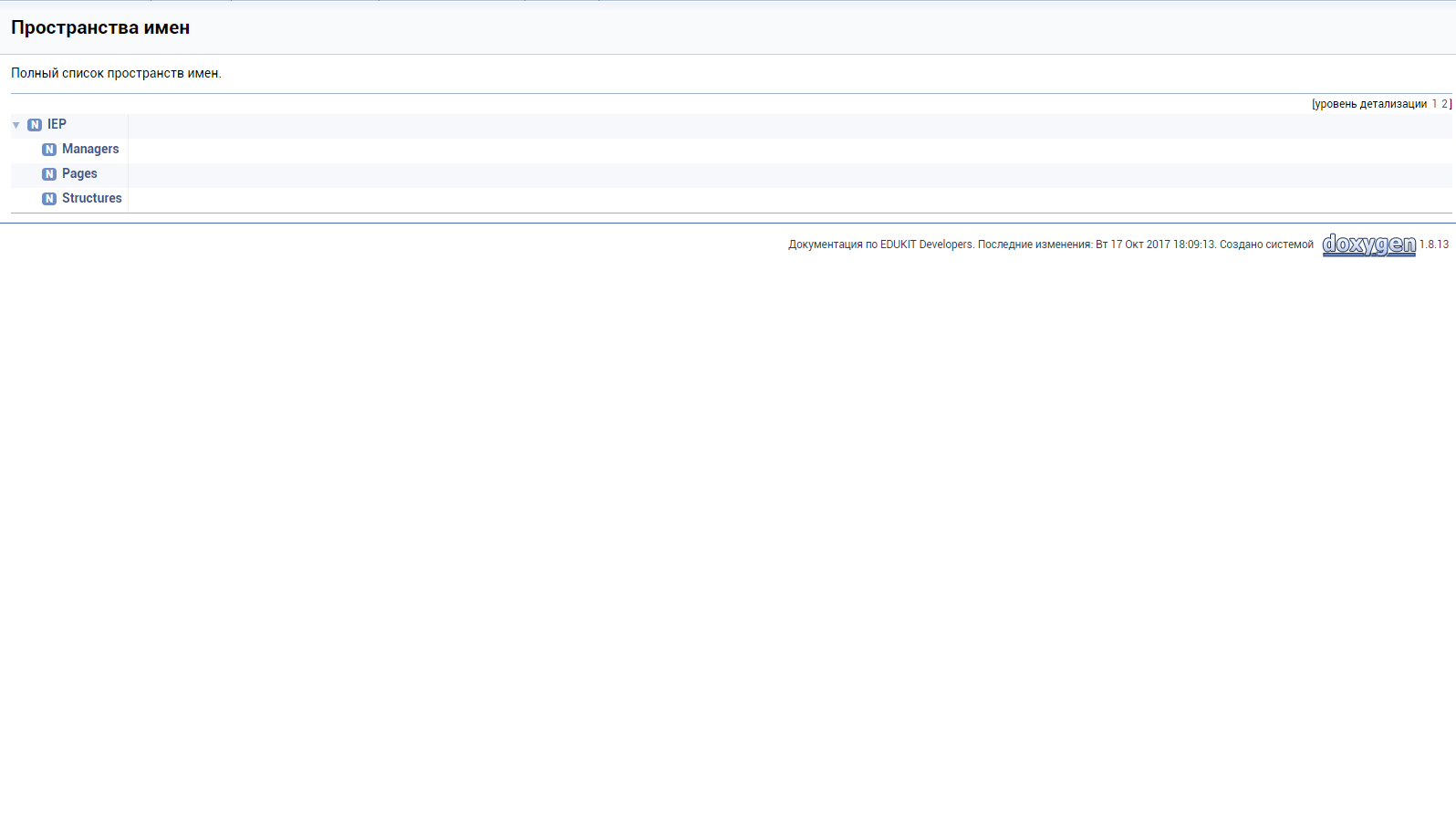


Рисунок 24 – Пространства имён

В данном разделе отображается иерархия пространств имён, которые содержат в себе классы.

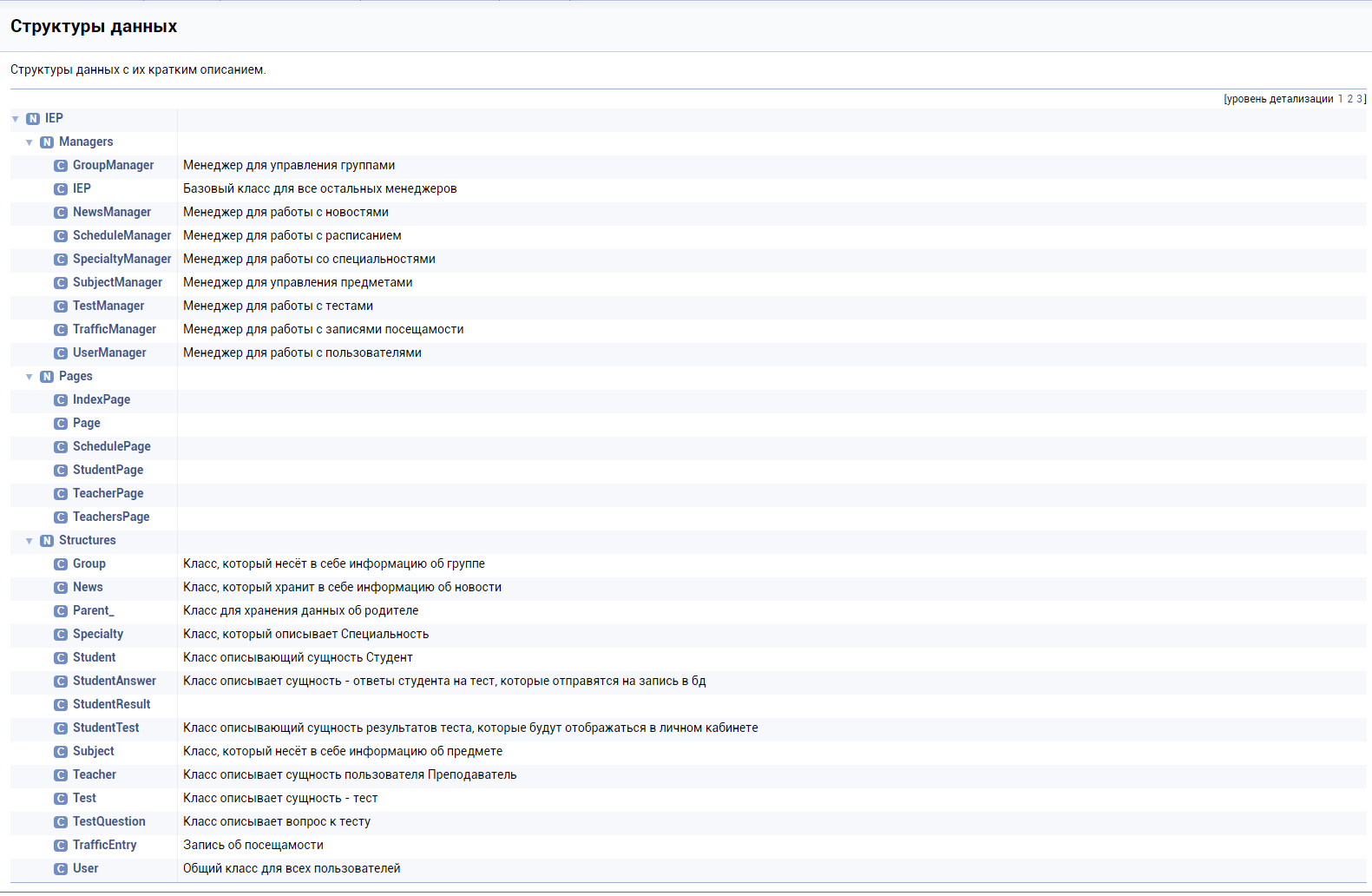


Рисунок 25 – Структуры данных

В данном разделе приводиться иерархия структур данных (классы) и их краткое описание.

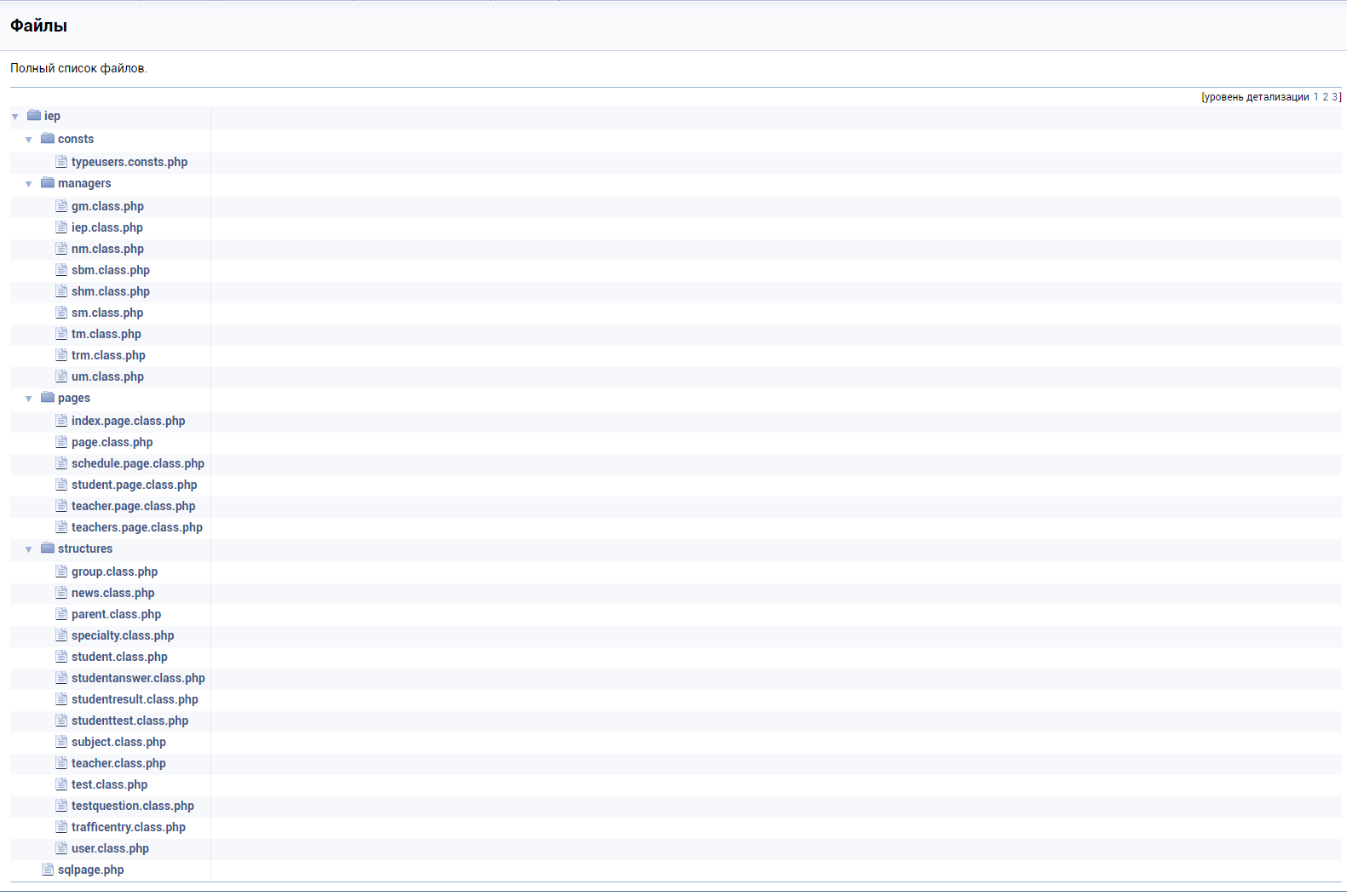


Рисунок 26 – Файлы

В данном разделе приводиться иерархия файлов.

# приложение д

**ДОКЛАД**

1. Доклад

### Введение

Информационно-образовательный портал разрабатывается для автоматизации деятельности учебного отделения.

### Основная часть

Основные задачи, решаемые информационно-образовательным порталом:

* Централизованное хранение данных о студентах, группах, специальностях, образовательных предметах, преподавательском составе;
* Хранение данных о посещаемости студентов;
* Хранение данных о результатах тестирования для дальнейшей их обработки;
* Составление расписания для информирования студентов;
* Оповещение родителей по электронной почте.

Основные достоинства:

1. Бесплатная;
2. Централизованное хранилище данных;
3. Автоматизирует основные процессы в работе учебного отделения.

Основные недостатки:

1. Нет интеграции с другими системами;
2. Любительская разработка.

### Заключение

Результаты моей работы заключается в изучении разработки информационных систем, предназначенных для сбора, хранения и обработки информации, которые несут в себе цель – автоматизация процессов.

Разработанная система предназначена только для внутреннего использования колледжем и не претендует на дальнейшее развитие как универсальной информационной системы, которая была бы способна конкурировать с аналогами, такими как 1С, МРКО и другие.

1. Презентация



Рисунок 27 – 1 слайд

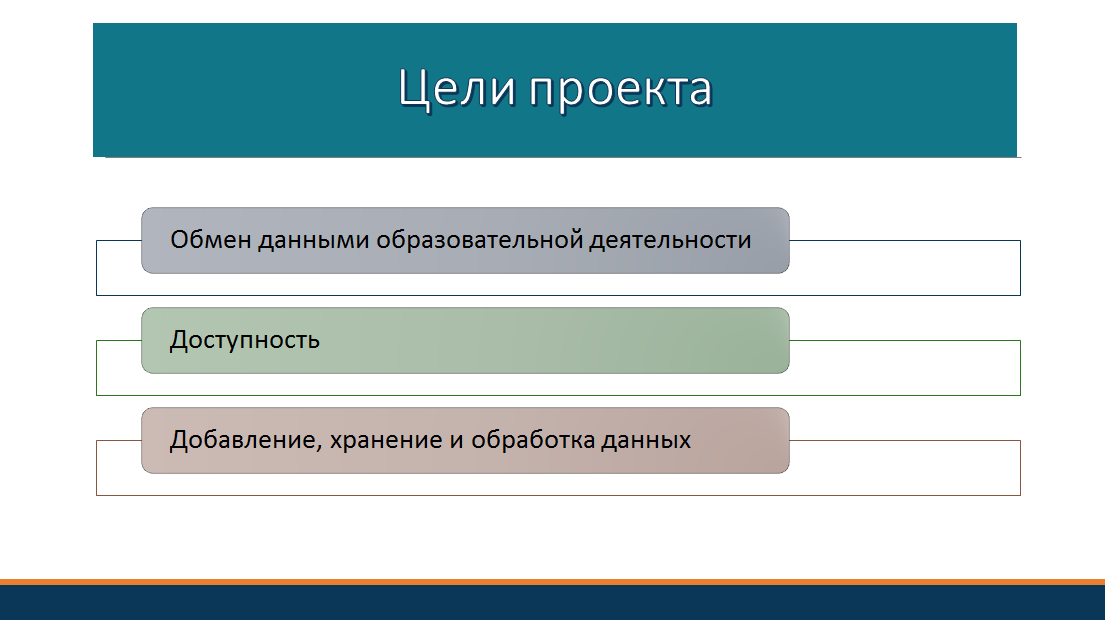


Рисунок 28 – 2 слайд

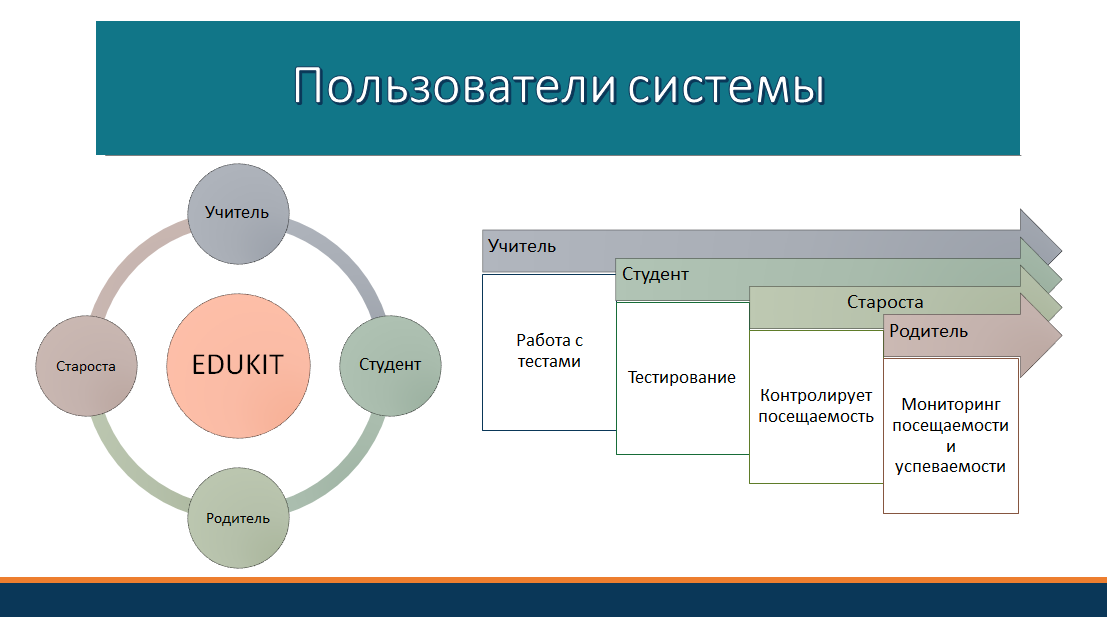


Рисунок 29 – 3 слайд



Рисунок 30 – 4 слайд



Рисунок 31 – 5 слайд

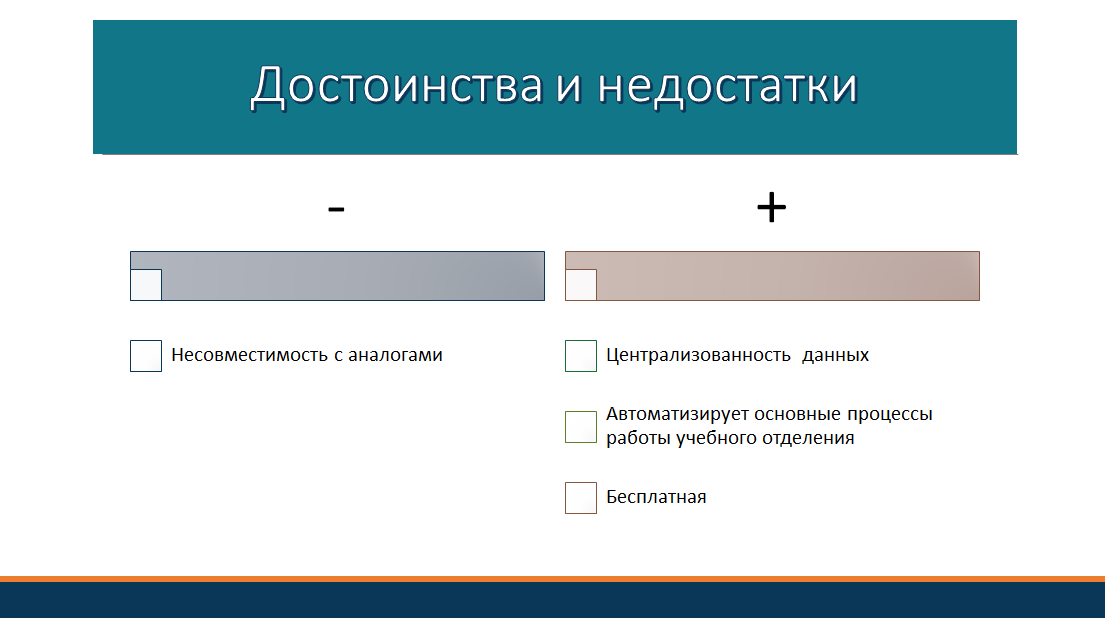


Рисунок 32 – 6 слайд



Рисунок 33 – 7 слайд

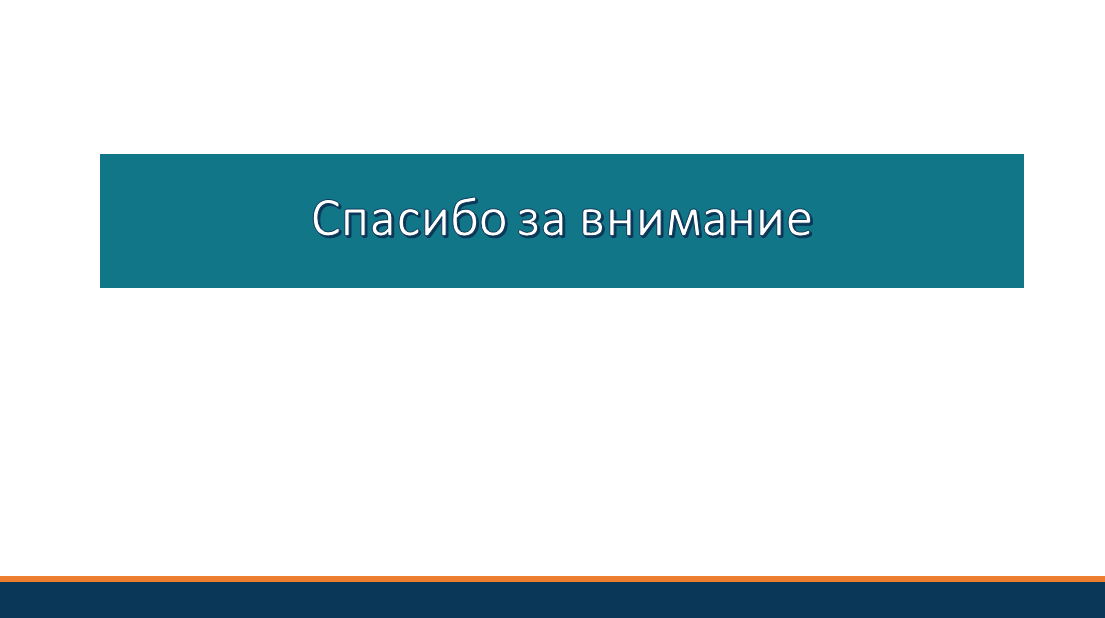


Рисунок 34 – 8 слайд