

Государственное бюджетное профессиональное образовательное учреждение
«Политехнический колледж городского хозяйства»

к.т.н. Л.В. Ильюшенков

Методические указания по курсовому проектированию для студентов всех
форм обучения специальности 09.02.07 — Программирование в компьютерных
системах

МДК 09.01 - Проектирование и разработка веб-приложений

Санкт-Петербург
2025

Автор: Леонид Владимирович Ильюшенков – кандидат технических наук, преподаватель высшей категории

Л.В. Ильюшенков

Курсовое проектирование по МДК.09.01 - Проектирование и разработка веб-приложений. Методические указания предназначены для практического обучения по модулю ПМ.09 — Проектирование, разработка и оптимизация веб-приложений, входящим в специальность 09.02.07 – Информационные технологии и программирование всех форм обучения. СПБ: Политехнический колледж городского хозяйства, 2025 – 32 с.

Методические указания предназначены для студентов среднего профессионального образования. В качестве тем курсовых проектов предлагается разработать *REST API* на основе фреймворка *Yii2*. Рассмотрены вопросы настройки сервера, работа с базой данных *mysql*, создание маршрутизации, настройка моделей и контроллеров. Даны сведения об обеспечении безопасности приложения. Приведены указания по оформлению программной документации в соответствии с действующими стандартами.

СОДЕРЖАНИЕ

Введение	4
Задание на курсовое проектирование.....	5
Перечень примерных тем проектов.....	7
Занятие 1. Разработка технического задания	7
Занятие 2. Проектирование базы данных.....	10
Занятие 3. Развёртывание Yii2 на сервере.....	10
Занятие 4. Реализация API	13
Занятие 5. Авторизация и аутентификация пользователей, загрузка файлов	20
Занятие 6. Тестирование REST API	24
Занятие 7. Разработка пояснительной записи по ГОСТ 34.201 — 2020.....	25
Занятие 8. Разработка руководства администратора	29
Список источников	30

Введение

Методические указания предназначены для студентов всех форм обучения специальности 09.02.07 — Информационные системы и программирование, квалификация — разработчик веб и мультимедийных приложений. Курсовое проектирование производится в рамках изучения профессионального модуля ПМ.09 — Проектирование, разработка и оптимизация веб-приложений.

Целью курсового проектирования является освоение следующих компетенций:

- ПК 9.1. Разрабатывать техническое задание на веб-приложение в соответствии с требованиями заказчика.
- ПК 9.2. Разрабатывать веб-приложение в соответствии с техническим заданием.
- ПК 9.4. Осуществлять техническое сопровождение и восстановление веб-приложений в соответствии с техническим заданием.
- ПК 9.5. Производить тестирование разработанного веб приложения.
- ПК 9.6. Размещать веб приложения в сети в соответствии с техническим заданием.

В качестве курсового проекта предлагается разработать *REST API* на фреймворке *Yii2* для решения задач бизнеса. Чтобы успешно выполнить проект вам потребуются следующие знания:

- язык программирования *PHP*;
- реляционная система управления базами данных *MariaDB*;
- протокол *HTTP*;
- веб-интерфейс для администрирования СУБД *Phpmyadmin*;
- панель управления сервером *Hestia*;
- текстовый формат обмена данными *JSON*;
- программа тестирования серверных приложений *Postman*;
- Среда разработки *Phpstorm* или *VSCode*.

Технология *REST API* нужна для предоставления и организации доступа к веб-службам. Технология позволяет пользователям гибко подключаться к облачным сервисам, управлять ими и взаимодействовать в распределенной среде. Приложения использующие эту технологию легко масштабируются, имеют унифицированный интерфейс и простоту поддержки.

Задание на курсовое проектирование

Государственное бюджетное профессиональное образовательное учреждение
«Политехнический колледж городского хозяйства»

СОГЛАСОВАНО

Председатель
П(Ц)К 09.02.07
_____ Л.В. Левит
«__» ____ 20____г.

Специальность: 09.02.07 — Информационные технологии и программирование

Задание на курсовой проект студента

(фамилия, имя, отчество)

Тема: _____

Группа: _____ Курс: ____ Форма обучения: _____

Руководитель: _____ к.т.н. Л.В. Ильюшенков
подпись

Время выполнения проекта/работы с «__» ____ по «__» ____ 20____г.

Студент _____ ФИО
подпись

ПЕРЕЧЕНЬ ВОПРОСОВ, ПОДЛЕЖАЩИХ РАЗРАБОТКЕ

Содержание задания по профилирующим разделам курсового проекта	Срок выполнения	ПМ*	МДК*	ПК*	ОК*
1) Техническое задание, ГОСТ 34.602— 2020	ПМ 09	МДК 09.01	9.1; 9.2; 9.4; 9.5; 9.6	1-10	
2) Проектирование базы данных					
3) Развёртывание Yii2 на сервере					
4) Реализация REST API					
5) Тестирование REST API					
6) Разработка пояснительной записи, ГОСТ 34.201 — 2020					
7) Разработка руководства администратора ГОСТ Р ИСО/МЭК 25051—2017					
Источники, используемые в разработке					

Примечание: расшифровку ПМ, МДК, ПК, ОК см. Федеральный государственный образовательный стандарт среднего профессионального образования по специальности 09.02.07 Информационные системы и программирование (утв. приказом Министерства образования и науки РФ от 9 декабря 2016 г. № 1547)

Требования к оформлению

- Документация должна быть выполнена машинным способом, на одной стороне листа, шрифт Times New Roman 14, полуторный интервал. Допускается уменьшать размер шрифта при оформлении рисунков, таблиц, текста программы.
- Программная документация и материалы к курсовому проекту (работе) оформляются в соответствии с требованиями ГОСТ 19.106 — 78 .
- Программный продукт, программные документы предоставляются на защите курсового проекта.

Дата выдачи курсового задания «__» _____ 20____г

Руководитель курсового проекта: _____ Л.В. Ильюшенков
подпись

Рассмотрено и одобрено предметной комиссией: протокол № __ от «__» _____ 20____г.

Председатель П(Ц)К 09.02.07 _____ Л.В. Левит

Перечень примерных тем проектов

- 1) Разработка серверного приложения REST API «Агентство по продаже билетов на транспорт»
- 2) Разработка серверного приложения REST API «Автоматизированное рабочее место сотрудника отдела кадров»
- 3) Разработка серверного приложения REST API «Курьерская доставка еды»
- 4) Разработка серверного приложения REST API «Платформа для проведения спортивных соревнований»
- 5) Разработка серверного приложения REST API «Автоматизированное рабочее место сотрудника склада»
- 6) Разработка серверного приложения REST API «Автоматизированное рабочее место страхового агента»
- 7) Разработка серверного приложения REST API «Метрологическая служба по поверке средств измерений»
- 8) Разработка серверного приложения REST API «Инвентаризация»
- 9) Разработка серверного приложения REST API «Расчет сдельной оплаты труда»
- 10) Разработка серверного приложения REST API «Сестринский патронаж за новорожденными»
- 11) Разработка серверного приложения REST API «Оказания материальной помощи студентам»
- 12) Разработка серверного приложения REST API «Бронирование гостиничных номеров»
- 13) Разработка серверного приложения REST API «Автоматизированное рабочее место диспетчера жилищной управляющей компании»
- 14) Разработка серверного приложения REST API «Управление документацией на предприятии»
- 15) Разработка серверного приложения REST API - Свободная тема

Занятие 1. Разработка технического задания

Содержание разделов технического задания должно соответствовать **ГОСТ 34.602 — 2020 Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.** Обязательными разделами являются:

- 1) общие сведения;
- 2) цели и назначение создания автоматизированной системы;
- 3) характеристика объектов автоматизации;
- 4) требования к автоматизированной системе;
- 5) состав и содержание работ по созданию автоматизированной системы;
- 6) порядок контроля и приемки автоматизированной системы;

- 7) требования к составу и содержанию работ по подготовке объекта автоматизации к вводу автоматизированной системы в действие;
- 8) требования к документированию;
- 9) источники разработки.

1) Общие сведения. Раздел должен содержать следующие сведения:

- полное наименование программы и ее условное обозначение;
- наименование организации-разработчика;
- перечень документов, на основании которых создается программа, кем и когда утверждены эти документы;
- плановые сроки начала и окончания работ по созданию программы.

Сведения этого раздела необходимо взять из задания на курсовое проектирование. Условное обозначение программы:

ПКГХ 09.02.07 ИР-__-__ .номер студенческого.П2

2) Цели и назначение создания автоматизированной системы. Примерами целей проекта является автоматизация процесса бронирования билетов, покупки товаров и пр. В назначении указывают категории пользователей и их функционал. В соответствии с требованиями заказчика необходимо предусмотреть три категории пользователей: незарегистрированные пользователи, зарегистрированные пользователи и администраторы и указать их функционал.

3) Характеристика объектов автоматизации. Опишите основные автоматизируемые функции. Например: процесс покупки товаров включает в себя: просмотр и выбор товаров, добавление товаров в корзину, редактирование корзины, оформление заказа. Пока заказ не подтвержден продавцом, покупатель может редактировать заказ или удалить. Обязательно должны быть описаны процесса регистрации аутентификации и регистрации.

4) Требования к автоматизированной системе. Требования к системе должны содержать требования к передаваемым параметрам, запросу и ответу. Например:

Запрос для регистрации нового пользователя в системе (см. табл 1). При отправке запроса необходимо передать объект со следующими свойствами:

first_name – имя пользователя латинские или кириллические буквы обязательное поле, строка;

last_name – фамилия, латинские или кириллические буквы обязательное поле обязательное поле, строка;

phone – телефон, обязательное и уникальное поле, строка, не меньше 10 цифр и +;

document_number – номер документа, обязательное, строка из 10 цифр;

password – обязательное поле, строка, не менее чем из 6 цифр и латинских букв.

Таблица 1 — Регистрация пользователя

Запрос	Ответ
Регистрация пользователя	
URL: {host}/api/register Method: POST Headers - Content-Type: application/json Body: { "first_name": "Иван", "last_name": "Петров", "phone": "89001234568", "document_number": "1224567890", "password": "paSSword" }	-----Удачно ----- Status: 204 Body: null -----Ошибка валидации ----- Status: 422 Content-Type: application/json Body: { "error": { "code": 422, "message": "Validation error", "errors": { <key>: <массив ошибок> } } }

Примечание: Если запрос требует авторизации включите в заголовки: *Authorization: Bearer {token}*

Требования заказчика:

- Посты должны содержать временную метку (Timestamp) создания;
- Посты должны быть разбиты на категории;
- Профиль или другой ресурс должен содержать фотографию или файл (валидируются расширение и размер);
- При удалении ресурса, должны удаляться связанные с ним файлы.

5) Состав и содержание работ по созданию автоматизированной системы. Перечень работ содержится в п. 2-5 задания на курсовое проектирование.

6) Порядок контроля и приемки автоматизированной системы.

Контроль качества и приемка программного средства будет производиться при помощи программы *Postman*. Требуется разработать запросы к API. Проверке подлежат удачные ответы, ошибки валидации, авторизации и пр. На каждую автоматизированную функцию не менее трех запросов.

7) Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу автоматизированной системы в действие. Требуется указать по какому адресу должна быть размещена автоматизированная система, по какому адресу должны быть доступны файлы системы в *Github*. Какие работы необходимо произвести на сервере, чтобы система была доступна и находилась в рабочем состоянии.

8) Требования к документированию. Требования к документированию указаны в п. 6 и далее задания на курсовое проектирование.

Занятие 2. Проектирование базы данных

Войдите с предоставленными данными в свой профиль [https://xn—80ahdri7a.site/](https://xn--80ahdri7a.site/). Выберите DB/добавить базу данных и заполните форму — рис. 1.

Рис 1 - Форма создания базы данных

После сохранения формы к Вам на почту придут данные для входа на страницу Phpmyadmin. С порядком работы Вы можете ознакомиться в источнике [4].

Занятие 3. Развёртывание Yii2 на сервере

REST (от англ. Representational State Transfer — «передача репрезентативного состояния» или «передача самоописываемого состояния») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Другими словами, REST — это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать. В основе технологии лежит протокол HTTP (HTTPS) и J-son формат данных.

Приложение разрабатывается с применением фреймворка Yii2. Yii – это универсальный фреймворк и может быть задействован во всех типах веб-приложений. Благодаря его компонентной структуре фреймворк особенно подходит для разработки таких крупных проектов, как порталы, форумы, CMS, магазины или REST-приложения.

Для установки фреймворка скачайте со странице фреймворка <https://www.yiiframework.com/download> архив **Yii 2 with basic application template.tgz**. Конвертируйте архив в zip формат. Загрузите zip архив в папку public_html своего домена и распакуйте. Переместите все файлы из папки basic в папку public_html. Переименуйте папку web в api.

По-молчанию сервис apache2 входной точкой приложения является файл index.php в папке public_html. У нас файл index.php находится в папке api необходимо сделать переадресацию всех запросов в эту папку. Для этого в папке public_html создайте файл .htaccess со следующим содержанием:

```
RewriteEngine On
RewriteRule ^(.*)$ api/$1 [L]
```

Отредактируйте настройки приложения в файле config/web.php.

```
<?php

$params = require __DIR__ . '/params.php';
$db = require __DIR__ . '/db.php';

$config = [
    'id' => 'basic',
    'name'=>'shop',
    'language'=>'ru-RU',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'aliases' => [
        '@bower' => '@vendor/bower-asset',
        '@npm' => '@vendor/npm-asset',
    ],
    'components' => [
        'request' => [
            // !!! insert a secret key in the following (if it is empty) - this is required by cookie validation
            'cookieValidationKey' => 'Leonid',
            'parsers' => [
                'application/json' => 'yii\web\JsonParser',
            ]
        ],
        'cache' => [
            'class' => 'yii\caching\FileCache',
        ],
        'user' => [
            'identityClass' => 'app\models\User',
            'enableAutoLogin' => true,
        ],
        'errorHandler' => [
            'errorAction' => 'site/error',
        ],
    ],
];
```

```

'mailer' => [
    'class' => 'yii\swiftmailer\Mailer',
    // send all mails to a file by default. You have to set
    // 'useFileTransport' to false and configure transport
    // for the mailer to send real emails.
    'useFileTransport' => true,
],
'log' => [
    'traceLevel' => YII_DEBUG ? 3 : 0,
    'targets' => [
        [
            'class' => 'yii\log\FileTarget',
            'levels' => ['error', 'warning'],
        ],
    ],
],
'db' => $db,
]

'urlManager' => [
'enablePrettyUrl' => true,
'enableStrictParsing' => true,
'showScriptName' => false,
'rules' => [
['class' => 'yii\rest\UrlRule', 'controller' => 'user'],
['class' => 'yii\rest\UrlRule', 'controller' => 'trip'], // и так далее все табл.
],
]
];

if (YII_ENV_DEV) {
    // configuration adjustments for 'dev' environment
    $config['bootstrap'][] = 'debug';
    $config['modules']['debug'] = [
        'class' => 'yii\debug\Module',
        // uncomment the following to add your IP if you are not connecting from localhost.
        'allowedIPs' => ['127.0.0.1', '::1', '*'],
    ];
    $config['bootstrap'][] = 'gii';
    $config['modules']['gii'] = [
        'class' => 'yii\gii\Module',
        // uncomment the following to add your IP if you are not connecting from localhost.
        'allowedIPs' => ['127.0.0.1', '::1', '*'],
    ];
}

return $config;

```

Ведите данные для подключения к базе данных в файле config/db.php

```

<?php

return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=wsr_shop',
    'username' => 'wsr_shop',

```

```
'password' => 'Admin123*',
'charset' => 'utf8',

// Schema cache options (for production environment)
// "enableSchemaCache" => true,
// "schemaCacheDuration" => 60,
// "schemaCache" => 'cache',
];
```

Занятие 4. Реализация API

Сгенерируйте модели для всех таблиц с помощью Codegenerator yii, который находится по адресу {domain}/api/gii. Выберите Model Generator и заполните данные формы.

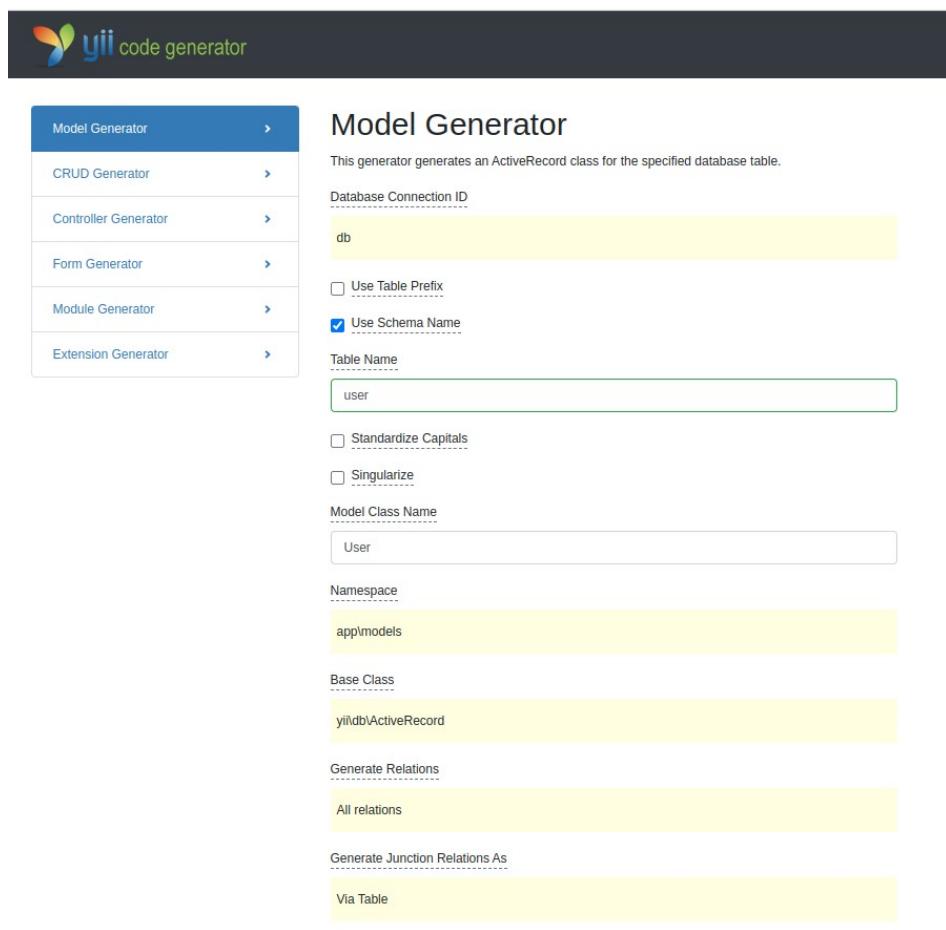


Рисунок 2 — Создание модели

Нажмите кнопку «Preview» и затем «Generate». Создайте каждой модели контроллер в папке public_html/controllers. Для модели User файл UserController.php, со следующим кодом

```
<?php
namespace app\controllers;
use yii\rest\ActiveController;
class UserController extends ActiveController
```

```
{
public $modelClass = 'app\models\User';
}
```

Мы создали RESTful API для доступа к данным пользователя. API нашего сервиса сейчас включает в себя:

- GET /users : получение постраничного списка всех пользователей;
- HEAD /users : получение метаданных листинга пользователей;
- POST /users : создание нового пользователя;
- GET /users/123 : получение информации по конкретному пользователю с id равным 123;
- HEAD /users/123 : получение метаданных по конкретному пользователю с id равным 123;
- PATCH /users/123 и PUT /users/123 : изменение информации по пользователю с id равным 123;
- DELETE /users/123 : удаление пользователя с id равным 123;
- OPTIONS /users : получение поддерживаемых методов, по которым можно обратится к /users ;
- OPTIONS /users/123 : получение поддерживаемых методов, по которым можно обратится к /users/123 .

Настройка маршрутизации

Готовое API Yii2 не всегда удовлетворяет требованиям проекта, часто требуется изменять формат отдаваемых данных, прописывать новые действия с моделями. Небольшие изменения можно произвести настройкой `yii\rest\ActiveController` [1 с.465 — 471]. `yii\rest\Controller` позволяет полностью переписать действия по умолчанию. Для начала нужно сконфигурировать пути и выполняемое действие в контроллере в файле `config/web.php`

```
'urlManager' => [
    'enablePrettyUrl' => true,
    'enableStrictParsing' => true,
    'showScriptName' => false,
    'rules' => [
        'POST register' => 'user/create',
        'POST login' => 'user/login',
        'GET station' => 'station/index',
        'GET station/<id>' => 'station/view',
        'GET trip' => 'trip/search'
    ]
]
```

Это производится в свойстве `rules`. Например строка `'POST register' => 'user/create'` означает

- POST — метод запроса;

- register — часть URL после имени хоста, например `http://busstation.сделай.site/register`;
- user — ссылка на контроллер — `controllers/UserController.php`;
- create — метод контроллера user — `actionCreate()`;

Настройка моделей и валидация

Правила валидации указываются в методе `rules()`:

```
public function rules()
{
    return [
        [['first_name', 'last_name', 'birth_day', 'phone', 'document_number'], 'required'],
        [['birth_day'], 'date'],
        [['first_name', 'last_name'], 'string', 'max' => 150],
        [['phone'], 'string', 'max' => 50],
        [['document_number'], 'string', 'length' => 10, 'unique'],
        [['password', 'token'], 'string', 'max' => 250],
    ];
}
```

Каждое правило содержит два элемента: список валидируемых полей и имя валидатора. Список встроенных валидаторов указан [1, с. 509-524]. Метод `$model->validate()` проверяет валидность данных моделей, возвращает `true`, если модель валидна. `$model->errors` — возвращает массив ошибок [1 с. 83]. Вернуть ошибки можно при валидации формы `ActiveForm::validate($model)`.

Настройка отображения полей модели осуществляется в методе `fields()`.

```
public function fields()
{
    return [
        // название поля совпадает с именем атрибута
        'id',
        // название поля "email", атрибут "email_address"
        'email' => 'email_address',
        // название поля "name", значение определяется callback-ом PHP
        'name' => function () {
            return $this->first_name . ' ' . $this->last_name;
        },
    ];
}
```

Удаление полей

```
public function fields()
{
    $fields = parent::fields();
    // удаляем небезопасные поля
    unset($fields['auth_key'], $fields['password_hash'],
        $fields['api_token']);
    return $fields;
}
```

```
}
```

Получение данных запросов

Чтобы получить параметры запроса, вы должны вызвать методы `get()` и `post()` компонента `request`

```
$request = Yii::$app->request;
$get = $request->get();
// эквивалентно: $get = $_GET;

$id = $request->get('id');
// эквивалентно: $id = isset($_GET['id']) ? $_GET['id'] : null;

$id = $request->get('id', 1);
// эквивалентно: $id = isset($_GET['id']) ? $_GET['id'] : 1;

$post = $request->post();
// эквивалентно: $post = $_POST;

$name = $request->post('name');
// эквивалентно: $name = isset($_POST['name']) ? $_POST['name'] : null;

$name = $request->post('name', '');
// эквивалентно: $name = isset($_POST['name']) ? $_POST['name'] : '';
```

При реализации *RESTful API*, требуется получить параметры, которые были отправлены через *PUT*, *PATCH* или другие методы запроса. Вы можете получить эти параметры, вызвав метод

```
Yii::$app->request->getBodyParams();
```

Загрузка данных при создании модели осуществляется методом `load()`

```
$model->load($data, $form);
```

где `$data` — данные свойств модели, `$form` — имя формы, в нашем случае `$form=""`, так как данные приходят в виде `json`.

Формирование ответа

Передача статус кода результата выполнения запроса

```
Yii::$app->response->statusCode = 200;
```

Формат ответа

```
$response->format = yii\web\Response::FORMAT_JSON;
```

Другие форматы ответа см. [1, с. 183]. Тело ответа

```
Yii::$app->response->content = $user;
```

Создание новых экземпляров моделей

Обычно создание новых моделей состоит из следующего алгоритма: получение запроса, загрузка данных в экземпляр модели, валидация данных, проверка прав доступа, сохранение в базе данных, отправка сообщения пользователю: 200 — Удачно, 201 — Создано, 204 — Нет контента, 422 — Ошибка валидации и пр.

```
<?php
namespace app\controllers;

use app\models\User;
use Yii;

class UserController extends FunctionController
{
    public $modelClass = 'app\models\User';

    public function actionCreate()
    {
        $data=Yii::$app->request->post();
        $user=new User();
        $user->load($data, '');
        if (!$user->validate()) return $user->errors;
        $user->password = Yii::$app->getSecurity()->generatePasswordHash($user->password);
        $user->save();
        $response=$this->response;
        header('Access-Control-Allow-Origin: *');
        $response->statusCode=204;
        return $response;
    }
}
```

Метод `save($validate)`, производит создание новой модели или обновления существующей. `$validate=true/false` — указывает нужно ли производить валидацию при обновлении модели. По умолчанию `true`.

В проектируемом приложении необходимо настроить политику безопасности CORS (*Cross-origin resource sharing*). В соответствии с политикой web-приложения, использующие API, могут запрашивать HTTP-ресурсы только с того домена, с которого были загружены, пока не будут использованы CORS-заголовки [2]. При отсутствии заголовка возникает ошибка в консоли браузера. Запросы бывают простые и сложные. Простой запрос, не требующий настройки удовлетворяет требованиям:

- Простой метод: *GET*, *POST* или *HEAD*
- Простые заголовки – разрешены только: *Accept*, *Accept-Language*, *Content-Language*,

- *Content-Type* со значением *application/x-www-form-urlencoded*, *multipart/form-data* или *text/plain*.

Все остальные запросы сложные. Перед тем, как послать сложный запрос, браузер генерирует и отправляет предварительный запрос, с параметрами:

- Метод: OPTIONS.
- Путь – точно такой же, как в основном запросе: /service.json.
- Особые заголовки: Origin – источник, Access-Control-Request-Method – запрашиваемый метод, Access-Control-Request-Headers – разделённый запятыми список «непростых» заголовков запроса.

Сервер должен ответить со статусом 200 и следующими заголовками: *header('Access-Control-Allow-Origin: *');* *header('Access-Control-Allow-Headers: *');* *header ('Access-Control-Allow-Methods: *');*. Значение * , означает что приложение не имеет ограничений. Для настройки политики CORS можно изменить входной файл index.php:

```
<?php

// comment out the following two lines when deployed to production
defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');

header( 'Access-Control-Allow-Origin: *');
header( 'Access-Control-Allow-Headers: *');
header ('Access-Control-Allow-Methods: *');

if ($_SERVER['REQUEST_METHOD']=='OPTIONS') {
    http_response_code(200);
    header('content-type: application/json');

    exit();
}
require __DIR__ . '/vendor/autoload.php';
require __DIR__ . '/vendor/yiisoft/yii2/Yii.php';

$config = require __DIR__ . '/config/web.php';

(new yii\web\Application($config))>run();
```

Для оптимизации программного кода, валидацию, отправку ответов и пр. можно вынести в отдельный класс.

```
<?php

namespace app\controllers;

use yii\rest\Controller;
use yii\widgets\ActiveForm;
```

```

class FunctionController extends Controller{

public function send($code, $data){
    $response=$this->response;
    $response->format = yii\web\Response::FORMAT_JSON;
    $response->data=$data;
    $response->statusCode=$code;
    return $response;
}

public function validation($model){
    $error=['error'=> ['code'=>422, 'message'=>'Validation error', 'errors'=>ActiveForm::validate($model)]]; 
    return $this->send(422, $error);
}
}

```

Этот класс наследует класс *yii\rest\Controller*, здесь приведено два метода: *send(\$code, \$data)* — возвращает ответ, *validation(\$model)* — производит валидацию модели. При наследовании этого класса в контроллерах будут доступны эти методы.

```

<?php
namespace app\controllers;

use app\models\User;
use Yii;

class UserController extends FunctionController
{

    public $modelClass = 'app\models\User_registration';

    public function actionCreate()
    {
        $data=Yii::$app->request->post();
        $user=new User_registration();
        $user->load($data, '');
        if (!$user->validate()) return $this->validation($user);
        $user->password=Yii::$app->getSecurity()->generatePasswordHash($user->password);
        $user->save();
        return $this->send(204, null);
    }
}

```

Выборка данных из базы

Создать новый объект запроса вызовом метода *yii\db\ActiveRecord ::find()*; например:

```

// возвращает покупателя с идентификатором 123
// SELECT * FROM `customer` WHERE `id` = 123
$customer = Customer::find()->where(['id' => 123])->one();

//возвращает продукты категории «Овощи»
// SELECT * FROM `product` WHERE `category` = 'овощи';

```

```
$product = Product::find()->where(['category'=>'овощи'])->all();
```

Метод *findOne()*: возвращает один объект модели, заполненный первой строкой результата запроса, метод *findAll()*: возвращает массив объектов

```
// возвращает покупателя с идентификатором 123
// SELECT * FROM `customer` WHERE `id` = 123
$customer = Customer::findOne(123);

// возвращает покупателей с идентификаторами 100, 101, 123 и 124
// SELECT * FROM `customer` WHERE `id` IN (100, 101, 123, 124)
$customers = Customer::findAll([100, 101, 123, 124]);
```

Условие выбора задается в методе *where()*, примеры использования [1 с. 271]. Дополнительные условия можно задать методами *andWhere()* или *orWhere()*

```
$status = 10;
$search = 'yii';
$query->where(['status' => $status]);
if (!empty($search)) {
    $query->andWhere(['like', 'title', $search]);
}
```

Метод *filterWhere()* формирует условие, когда пользователь не задал какие нибудь значение, например осуществить поиск по имени пользователя или email.

Для сортировки используют метод *orderBy()*

```
$query->orderBy([
    'id' => SORT_ASC,
    'name' => SORT_DESC,
]);
```

где *SORT_ASC* — сортировка по возрастанию, *SORT_DESC* — сортировка по убыванию. Метод *limit(20)* — указывает сколько записей выбрать, *offset(1050)*, начиная с какой делать выборку. Все инструменты работы с выборками указаны [1, с. 269-280].

Занятие 5. Авторизация и аутентификация пользователей, загрузка файлов

Аутентификация пользователей

При регистрации пользователя в базу данных записывают хешированный пароль

```
Yii::$app->getSecurity()->generatePasswordHash($user->password);
```

Для проверки соответствия пароля введенного пользователем - `$login_data->password` хешу пароля в базе данных `$user->password` используют

```
Yii::$app->getSecurity()->validatePassword($login_data->password, $user->password);
```

Для генерации случайного токена используют

```
Yii::$app->getSecurity()->generateRandomString()
```

Аутентификация пользователя в REST API состоит из проверки логина (в примере телефона), затем пароля, генерации токена, записи токена в базу данных, формирования ответа пользователю

```
public function actionLogin(){

    $data=\Yii::$app->request->post();
    $login_data=new LoginForm();
    $login_data->load($data, "");

    if (!$login_data->validate()) return $this->validation($login_data);

    $user=User::find()->where(['phone'=>$login_data->phone])->one();

    if (!is_null($user)) {
        if (\Yii::$app->getSecurity()->validatePassword($login_data->password, $user->password)) {
            $token = \Yii::$app->getSecurity()->generateRandomString();
            $user->token = $token;
            $user->save(false); //false — произвести запись без валидации
            $data = ['data' => ['token' => $token]];
            return $this->send(200, $data);
        }
    }

    return $this->send(401, ['error'=>['code'=>401, 'message'=>'Unauthorized', 'errors'=>['phone'=>'Неверный номер телефона или пароль']]]);
}
```

Данные введенные пользователем нуждаются в валидации. Для этого создана модель `LoginForm.php`

```
<?php

namespace app\models;

use Yii;
use yii\base\Model;

class LoginForm extends Model
{
    public $phone;
    public $password;
```

```

/**
 * @return array the validation rules.
 */
public function rules()
{
    return [
        // username and password are both required
        ['[phone', 'password'], 'required'],
        ...
    ];
}

```

Аналогичным образом следует поступить, если поля формы и модели не совпадают. Например в форме могут быть поля «Повторите пароль», «Подтвердите согласие на обработку персональных данных», которые в базу данных не записываются.

Авторизация пользователей

Авторизацию используют для контроля прав пользователя к выполнению какого-либо действия. В Yii2 для этого используют *IdentityInterface*. В настройках *config/web.php* нужно настроить компонент *user*

```

'components' => [
    'user' => [
        'identityClass' => 'app\models\User',
        'enableSession' => false
    ],
]

```

Необходимо прописать методы *IdentityInterface* в модели *User*

```

<?php
namespace app\models;
use yii\db\ActiveRecord;
use yii\web\IdentityInterface;

class User extends ActiveRecord implements IdentityInterface
{
    public static function tableName()
    {
        return 'user';
    }

    public static function findIdentity($id)
    {

    }

    public static function findIdentityByAccessToken($token, $type = null)
    {
        return static::findOne(['token' => $token]);
    }
}

```

```

public function getId()
{
    return $this->id;
}

public function getAuthKey()
{

}

public function validateAuthKey($authKey)
{
}

...

```

Подробнее об этих методах указано [1, с. 408]. Для REST приложения, вы можете реализовать только *findIdentityByAccessToken()* и *getId()*, остальные методы оставить пустыми. Приведенный выше код при отсутствии авторизации сформирует сообщение

```

{
    "name": "Unauthorized",
    "message": "Your request was made with invalid credentials.",
    "code": 0,
    "status": 401,
    "type": "yii\\web\\UnauthenticatedHttpException"
}

```

Для того, чтобы изменить сообщение нужно настроить компонент *response* в настройке *config/web.php*

```

'response' => [
    'class' => 'yii\web\Response',
    'on beforeSend' => function ($event) {
        $response = $event->sender;
        if ($response->data === null && $response->statusCode==401) {
            $response->data = ['error'=>['code'=>401, 'message'=>'Unauthenticated',
'errors'=>['phone'=>'Неверный номер телефона или пароль']];
            header('Access-Control-Allow-Origin: *');
            header('Content-Type: application/json');
        }
    },
],

```

В модулях (контроллерах), где нужно произвести авторизацию нужно изменить метод *behaviors()*

```

<?php
namespace app\controllers;

use app\controllers\FunctionController;
use app\models\Station;
use yii\filters\auth\HttpBearerAuth;

```

```

class StationController extends FunctionController

{
    public $modelClass = 'app\models\Station';

    public function behaviors()
    {
        $behaviors = parent::behaviors();
        $behaviors['authenticator'] = [
            'class' => HttpBearerAuth::class,
            'only'=>['delete']
        ];
        return $behaviors;
    }
}

```

Свойство *only* содержит список методов, для которых нужно контролировать доступ. Значение *delete* указывает на метод *actionDelete()*.

Загрузка файлов

Алгоритм загрузки файла состоит из следующих операций: получение данных формы, создание экземпляра модели, заполнение модели данными из запроса, валидация данных, получение доступа к загружаемому файлу, хеширование имени файла, сохранение файла в папку назначения, добавление имени и пути к файлу в модель, сохранение модели, информирование пользователя о результатах.

При загрузке файлов тело запроса должно иметь формат *form-data*. Загрузку файла выполняют при помощи класса *yii\web\UploadedFile*, доступ к файлу осуществляется методом *getInstanceByName()*

```
$station->photo=UploadedFile::getInstanceByName('photo');
```

где *photo* — метка файла. Хеширование имени файла можно осуществить функцией *hash()*

```
$hash=hash('sha256', $station->photo->baseName) . '.' . $station->photo->extension;
```

где *sha256* — алгоритм хеширования [3], *\$station->photo->baseName* — имя загруженного файла, *\$station->photo->extension* — расширение файла. Сохранение файла осуществляется методом *saveAs()*

```
$station->photo->saveAs(Yii::$app->basePath. '/assets/upload/' . $hash);
```

где *Yii::\$app->basePath* — абсолютный путь до папки с приложением на сервере. Программный код метода создания автобусной станции

```
public function actionCreate(){
```

```

$post_data=\Yii::$app->request->post();
$station=new Station();
$station->load($post_data, "");
if (!$station->validate()) return $this->validation($station);
$station->photo=UploadedFile::getInstanceByName('photo');
$hash=hash('sha256', $station->photo->baseName) . '.' . $station->photo->extension;
$station->photo->saveAs(\Yii::$app->basePath. '/assets/upload/' .
$hash);
$station->photo=$hash;
$station->save(false);
return $this->send(204, null);
}

```

Для проверки файлов используют валидаторы *file* и *image* [1, с. 515-519]. Для адекватной работы валидаторов нужно создать метод *beforeValidate()*

```

public function beforeValidate(){
    $this->image=UploadedFile::getInstanceByName('image');
    return parent::beforeValidate();
}

```

Занятие 6. Тестирование REST API

Для тестирования API предлагается использовать программу Postman. Узнать о порядке работы с Postman Вы можете в источнике [5]. Необходимо создать Postman коллекцию для каждого вида запросов. Эту коллекцию необходимо экспортить — рис. 3.

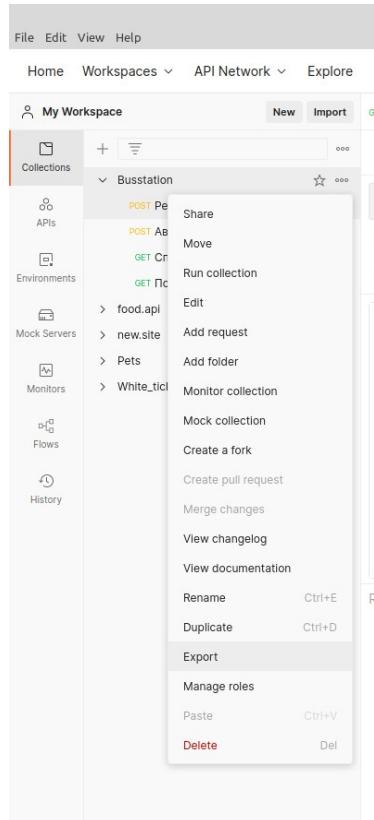


Рисунок 3 — Экспорт коллекции запросов Postman

Сохраните коллекцию в формате Json. Этот файл, вместе с другими файлами проекта необходимо разместить в Github.

Занятие 7. Разработка пояснительной записи по ГОСТ 34.201 — 2020

Пояснительная записка состоит из следующих разделов:

- 1) Описание проектируемых функций
- 2) Описание постановки задач
- 3) Описание информационного обеспечения системы
- 4) Описание организации информационной базы
- 5) Описание систем классификации и кодирования
- 6) Описание комплекса технических средств
- 7) Описание программного обеспечения
- 8) Описание алгоритма (проектной процедуры)
- 9) Отчет по результатам тестирования
- 10) Текст программы ГОСТ 19.401-78

1. Описание проектируемых функций и 2. Описание постановки задач должны соответствовать п. 2. Цели и назначение создания автоматизированной системы в техническом задании.

3. Описание информационного обеспечения системы. Необходимо указать внешние источники информации, с которыми система работает. Например, при работе системы, может быть необходимо подключение к сер-

вису google-мар для получения картографической информации. Если при работе программы необходима загрузка дополнительных программных модулей и файлов из сети Интернет (Bootstrap, jQery) необходимо это указать. Внешний носитель информации может быть необходим для получения данных. С помощью этих сведений может производиться диагностирование системы.

4. Описание организации информационной базы. В проектируемой информационной системе используется СУБД MariaDb. Интерфейс управления базой Phpmyadmin доступен по адресу: <http://xn—80ahdri7a.site/phpmyadmin>. Приведите ссылку на Github для скачивания дампа базы данных. Необходимо привести структуру базы данных, пример рис. 4

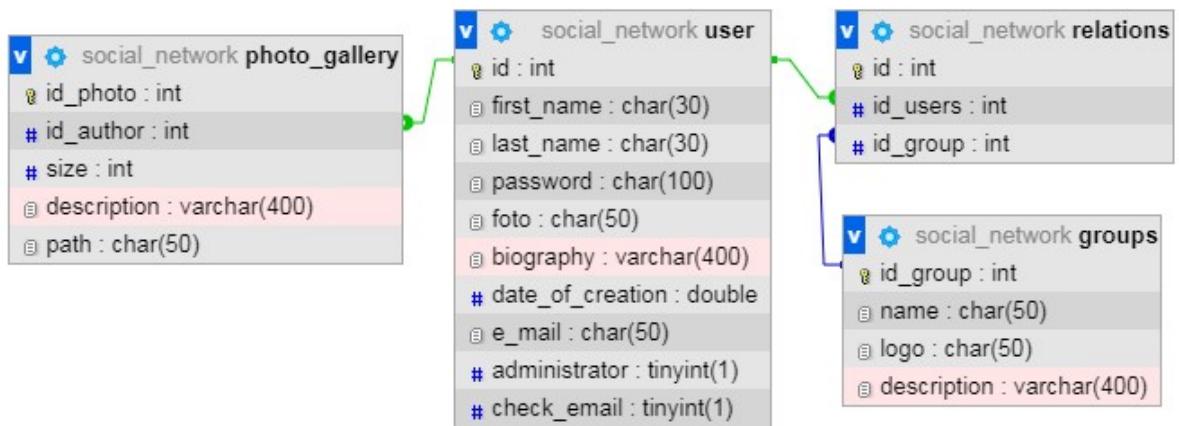


Рисунок 4 — Схема базы данных

К каждой таблице необходимо дать пояснение — табл. 2

Таблица 2 — Схема отношения photo_gallery

Наименование поля	Описание	Тип поля	Размер поля	Значение по умолчанию	Ограничения	Ключ или индекс
id_photo	Идентификатор	Целое	11	AUTO_INCREMENT	Обязательное поле	Первичный ключ
id_author	Идентификатор пользователя	Целое	11		Обязательное поле	Внешний ключ к user
size	Размер файла	Целое	11		Обязательное поле	
description	Описание	Строка	400		Необязательное поле	
path	Путь к файлу	Строка	50		Обязательное поле	

5. Описание систем классификации и кодирования. Системы классификации могут быть: Международные, общероссийские, отраслевые, системные (на предприятиях) и др. Список общероссийских классификаторов

приведен в перечне [6]. В нашем колледже используется Перечень специальностей среднего профессионального образования [7]:

– Укрупненная группа специальностей: 09.00.00 — Информатика и вычислительная техника:

- 09.02.01 Компьютерные системы и комплексы;
- 09.02.06 Сетевое и системное администрирование;
- 09.02.07 Информационные системы и программирование и др.

В колледже используется следующая система кодирования групп:

Группа ИР-19-4, где ИР — специальность и специализация **Информационные системы** и программирование, разработчик веб и мультимедийных приложений, 19 — год поступления, 2019 г., 4 — номер группы на отделении. На предприятиях используют системы кодирования продукции, сборочных единиц, документов, версий программного обеспечения и модулей и пр. Этот вопрос является предметом изучения на производственном обучении.

6. Описание комплекса технических средств. Для реализации API используются виртуальный сервер со следующими характеристиками:

Процессор	Xeon Scalable до 3.2 ГГц
Оперативная память	2 Гб
Объем памяти, SSD	40 Гб
Интернет канал	100 Мбит/с.
IP, 2 шт	82.146.57.151
Объем диска для резервных копий	20 Гб

7. Описание программного обеспечения. На сервере используется следующее программное обеспечение:

- Операционная система Ubuntu 20.04.3 LTS [8];
- Панель управления Hestia [4];
- Фреймворк Yii2 [1].

8. Описание алгоритма (проектной процедуры). В этом разделе необходимо составить диаграмму классов. Диаграмма классов описывает программную систему через ее компоненты (объект, классы), отношение и взаимодействия между ними [9], рис. 5.

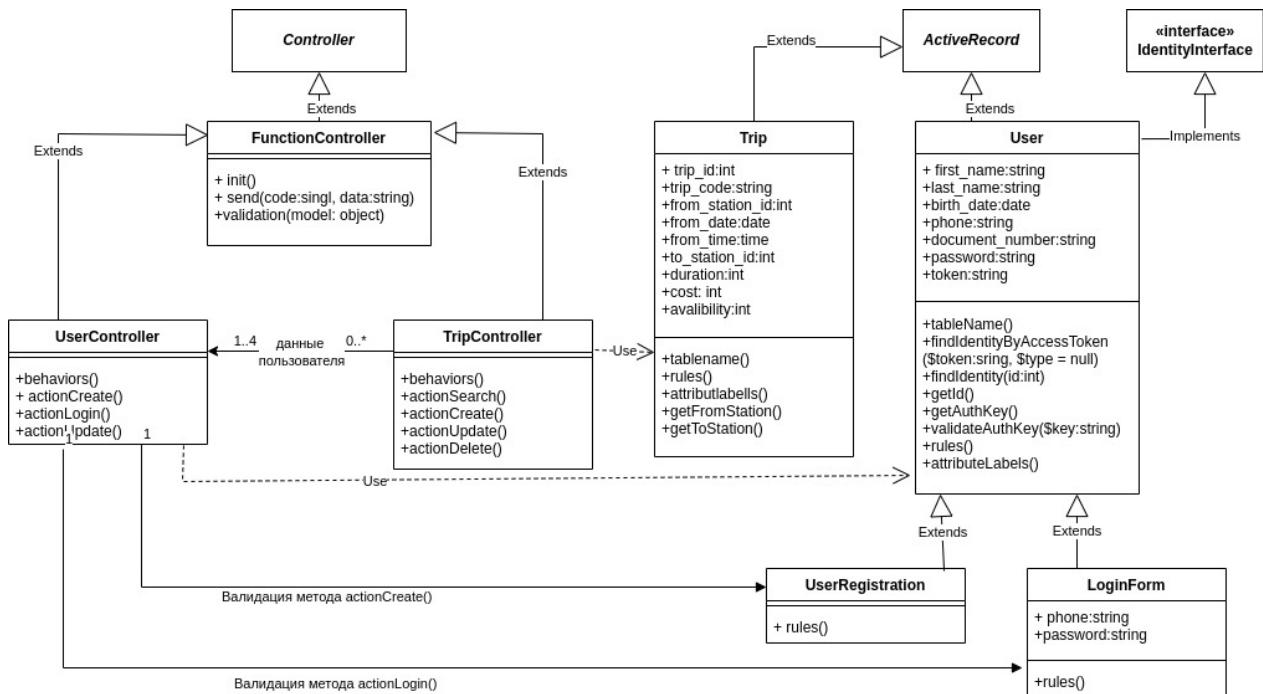


Рисунок 5 — Диаграмма классов

Все контроллеры наследуются от класса *Yii2 Controller* и *FunctionController*. *FunctionController* содержит общие методы для других классов: *UserController* — управление пользователями, *TripController* — управление бронированием:

- 1) **init()** - содержит настройки приложения;
- 2) **send(code, data)** — формирует ответ, *code* — статус код запроса, *data* — тело ответа;
- 3) **validation(model)** — производит валидацию модели (*model*) и формирует массив ошибок при неудачной валидации.

Методы контроллеров *UserController* и *TripController* понятны из их названия. Метод *behavior()* - обеспечивает контроль доступа пользователей к другим методам.

Модели *Trip* и *User* используются для работы с одноименными таблицами базы данных и наследуются от класса *Yii2 ActiveRecord*. Интерфейс *IdentityInterface* применяется к классу *User* и содержит набор методов для авторизации и аутентификации. Класс *Trip* получает данные о станциях со связанный таблицей *stations* с помощью методов *getFromStation()* и *getToStation()*. Классы *UserRegistration* и *LoginForm* наследуются от класса *User*. В них переопределяется метод *rules()* - содержащий правила валидации, так как при авторизации и регистрации правила валидации различны. Класса *UserController* и *TripController* используют модели *User* и *Trip* для работы с базой данных.

Экземпляры класса *Trip* используются для оформления бронирования данные экземпляров класса *User*. В одном бронировании может содержаться информация об 1...4 пользователях.

9. Отчет по результатам тестирования должен соответствовать ГОСТ Р ИСО/МЭК 25051 — 2017. Результаты тестирования сводятся в табл. 3. В начале процедуры тестирования необходимо привести описание программного обеспечения для тестирования.

Таблица 3 — Результаты тестирования

Элемент проверки	Запрос				Ответ		Вывод
	Метод	URL	Тело	Пользователь	Статус-код	Тело	
Авторизация	POST	http:// food.сделай.site/auth	login: admin password: admin	Гость	200	{"status": "true", "token": "8bc34fea221814b21ed7c0358373be"}	Соответствует
	POST	http:// food.сделай.site/auth	login: admin password: admin23	Гость	401	{"status": "false", "message": "Invalid authorization data"}	Соответствует
Просмотр блюд	GET	http:// food.сделай.site/posts	-	Гость	200	{"food_id": "43", "title": "Гороховый суп", "anons": "Гороховый суп (свежий)", "text": "Вода, горох, специи", "tags": "["Вода\", \"кубик\", \"маги\"]", "image": "post_image/1610116392_gorohovii_sup_obichnii-435081.jpg", "datetime": "14:33 08.01.2021" } и еще 4 блюда	Не соответствует, присутствует неоговоренное экранирование тегов

В конце таблицы необходимо сделать ссылку на файл тестовых запросов на Github.

Необходимо сделать вывод о соответствии результатов тестирования техническому заданию.

10) Текст программы ГОСТ 19.401-78. Текст программы должен содержать настройки приложения, модели и контроллеры. Текст программы разбивают на разделы и подразделы, при необходимости делают комментарии.

Занятие 8. Разработка руководства администратора

Руководство администратора содержит следующие разделы:

- 1) Введение
- 2) Назначение и условия применения
- 3) Подготовка к работе
- 4) Описание операций

- 5) Аварийные ситуации
- 6) Рекомендации по освоению

1) Введение. Этот раздел содержит: область применения; краткое описание возможностей; уровень подготовки пользователя.

2) Назначение и условия применения. В разделе указывают: виды деятельности, функции, для автоматизации которых предназначено данное средство; условия, при соблюдении (выполнении, наступлении) которых обеспечивается применение средства автоматизации в соответствии с назначением. В этом разделе допускаются ссылки на разделы технического задания. Программные средства, предназначенные для работы с API должны поддерживать протокол HTTP (включая методы....), иметь возможность принимать и отдавать данные в формате JSON.

3) Подготовка к работе. В этом разделе необходимо описать каким образом приложение расположенное на Github развернуть на сервере и проверить его работоспособность.

4) Описание операций, 5) Аварийные ситуации допускается приводить ссылки на техническое задание.

6) Рекомендации по освоению. Необходимо привести сведения о генерации запросов в Postman для различных языков программирования.

Список источников

- 1) Цян Суэ, Марков А. и др. Полное руководство по Yii 2.0 [Электронный ресурс] - <https://www.yiiframework.com/doc/download/yii-guide-2.0-ru.pdf>
- 2) Cross-Origin Resource Sharing (CORS) [Электронный ресурс] - <https://developer.mozilla.org/ru/docs/Web/HTTP/CORS>
- 3) PHP:hash — Manual [Электронный ресурс] - <https://www.php.net/manual/ru/function.hash.php>
- 4) An open-source Linux web server control panel Hestia [Электронный ресурс] - <https://www.hestiacp.com>
- 5) Postman Support [Электронный ресурс] - <https://www.postman.com/support>
- 6) Общероссийские и ведомственные классификаторы [Электронный ресурс] - <https://rosstat.gov.ru/classification>
- 7) Перечень профессий и специальностей среднего профессионального образования, необходимых для применения в области реализации приоритетных направлений модернизации и технологического развития экономики Российской Федерации [Электронный ресурс] - https://profstandart.rosmintrud.ru/obshchiy-informatsionnyy-blok/spravochno-informatsionnyy-blok/klassifikatory-spravochniki-i-perechni/detail.php?ELEMENT_ID=88492

8) Enterprise Open Source and Linux | Ubuntu [Электронный ресурс] -
<https://ubuntu.com>

9) Использование диаграммы классов UML при проектировании и документировании программного обеспечения [Электронный ресурс] -
<https://habr.com/ru/post/572234/>

Нормативная документация

1) ГОСТ 34.602-2020 Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы [Электронный ресурс] -
<https://protect.gost.ru/document1.aspx?control=31&id=241754>

2) ГОСТ 34.201-2020 Информационные технологии. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем [Электронный ресурс] -
<https://protect.gost.ru/document1.aspx?control=31&id=241756>

3) ГОСТ Р ИСО/МЭК 25051-2017 - Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Требования к качеству готового к использованию программного продукта (RUSP) и инструкции по тестированию [Электронный ресурс] -
<https://protect.gost.ru/document.aspx?control=7&id=217667>

4) ГОСТ 19.106-78 Единая система программной документации. Требования к программным документам, выполненным печатным способом [Электронный ресурс] - <https://protect.gost.ru/document.aspx?control=7&id=156370>

5) ГОСТ 19.401-78 Единая система программной документации. Текст программы. Требования к содержанию и оформлению [Электронный ресурс] -
<https://protect.gost.ru/document.aspx?control=7&id=155463>