

Подготовка к демонстрационному экзамену для чайников. Сборник подсказок.

Автор: Грушевский Юрий Викторович

Контакты: jurapro@yandex.ru

Оглавление

1. Введение	4
Для кого предназначено данный материал	4
Необходимые начальные знания	4
Примечание	4
2. Первоначальная установка и настройка фреймворка	5
Настройка рабочего места	5
Дано	5
Надо	5
Алгоритм	5
Установка фреймворка	6
Дано	6
Надо	6
Алгоритм	6
Настройка ЧПУ	8
Дано	8
Надо	8
Алгоритм	8
Создание локального репозитория	9
Дано	9
Надо	9
Алгоритм	9
3. Создание структуры и наполнение базы данных	10
Создание структуры БД	10
Дано	10
Надо	10
Алгоритм	10
Создание структуры БД с помощью миграций	11
Дано	11
Надо	11
Алгоритм	12
4. Реализация базовых алгоритмов	18
Аутентификация	18
Дано	18
Надо	18
Алгоритм	18
Базовая регистрация	21
Дано	21
Надо	21
Алгоритм	21

Валидация при регистрации	26
Дано	26
Надо	26
Алгоритм	26
5. Реализация специфических алгоритмов из задания	28
Организация личного кабинета пользователя	28
Дано	28
Надо	28
Алгоритм	28
Добавление новой заявки пользователем	33
Дано	33
Надо	33
Алгоритм	33
Организация панели управления сайтом администратора	35
Дано	35
Надо	36
Алгоритм	36
Добавление уведомлений	41
Дано	41
Надо	41
Алгоритм	41
Добавление анимаций	43
Дано	43
Надо	43
Алгоритм	43
Кастомизация дизайна	45
Дано	45
Надо	45
Алгоритм	45
Использования предоставленных медиаматериалов	50
Дано	50
Надо	50
Алгоритм	50
Сохранение дизайна согласно заданию	53
Дано	53
Надо	53
Алгоритм	53

1. Введение

Для кого предназначено данный материал

Данный материал не предназначен для использования в основном педагогическом процессе. Его можно воспринимать скорее как быстрый способ освежить знания и умения работы с фреймворком уіі или оптимизацию типовых задач, встречающихся на демонстрационном экзамене.

Необходимые начальные знания

Предполагается, что читатель знаком с основами программирования, основами HTML и CSS. Умеет читать и писать.

Примечание

Подразумевается, что на рабочем месте установлено все программное обеспечение, предусмотренное инфраструктурным листом для компетенции “Веб-технологии”

2. Первоначальная установка и настройка фреймворка

Настройка рабочего места

Дано

Не настроенное рабочее место для сдачи демонстрационного экзамена.

Надо

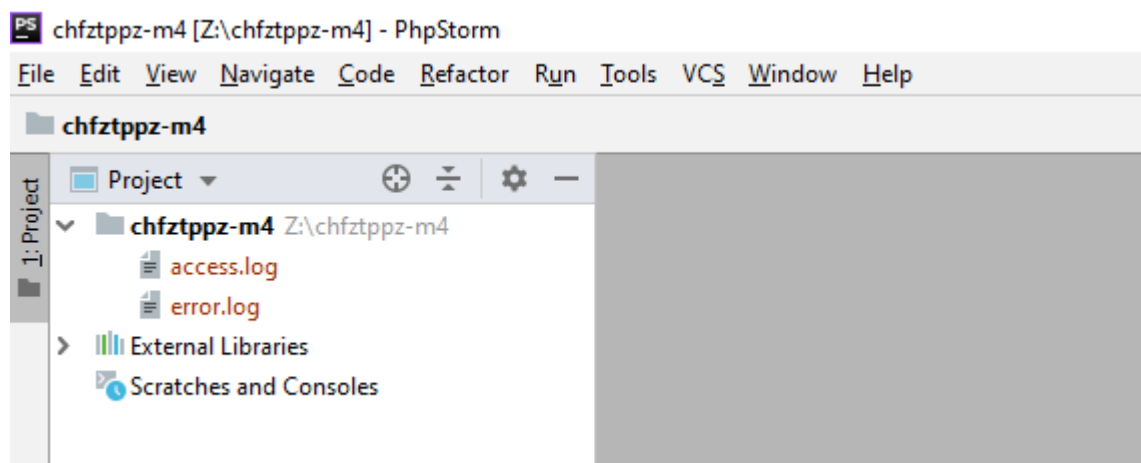
Подключиться к серверу и настроить инфраструктуру для дальнейшего выполнения задания

Алгоритм

1. В командной строке прописывать параметры подключения (конкретные параметры подключения у каждого свои):
 - a. linux: **sshfs login@192.168.14.26:/srv/users/login ~/mount** (директорию mount нужно предварительно создать в домашней директории)
 - b. windows: **net use Y: \\192.168.14.26\login password /user:login**

Примечание: 192.168.14.26 - IP адрес сервера, у каждой площадки собственный

2. Открыть в **Phpstorm** необходимый модуль как проект:



3. Настроить прокси-сервер:
 - a. linux: в зависимости от дистрибутива интерфейс настройки прокси-сервера может отличаться, параметры настройки совпадают с параметрами для windows
 - b. Windows: Пуск->Поиск->Прокси

Настройка прокси вручную

Использование прокси-сервера для подключений к Ethernet или сетям Wi-Fi. Эти параметры не применяются для VPN-подключений.

Использовать прокси-сервер

☒ Вкл.

Адрес

192.168.13.244

Порт

3128

Не использовать прокси-сервер для адресов, которые начинаются с указанных ниже записей. Для разделения записей используйте точку с запятой (;).

☒ Не использовать прокси-сервер для локальных (внутрисетевых) адресов

Сохранить

Примечание: для сохранения необходимо нажать на кнопку Сохранить

Установка фреймворка

Дано

Настроенное рабочее окружение

Надо

Установить и настроить фреймворк для выполнения задания

Алгоритм

1. Скопировать и разархивировать фреймворк в корневую директорию модуля

chfztpz (\192.168.13.244) (Z:) > chfztpz-m4

Имя	Дата изменения	Тип	Размер
.idea	29.02.2020 8:52	Папка с файлами	
assets	22.01.2020 5:41	Папка с файлами	
commands	22.01.2020 5:41	Папка с файлами	
config	22.01.2020 5:41	Папка с файлами	
controllers	22.01.2020 5:41	Папка с файлами	
mail	22.01.2020 5:41	Папка с файлами	
models	22.01.2020 5:41	Папка с файлами	
runtime	22.01.2020 5:41	Папка с файлами	
tests	22.01.2020 5:41	Папка с файлами	
vagrant	22.01.2020 5:41	Папка с файлами	
vendor	22.01.2020 5:42	Папка с файлами	

- Установить параметр **cookieValidationKey** в файле **config/web.php** (любая случайная строка):

```

11     '@bower' => '@vendor/bower-asset',
12     '@npm'   => '@vendor/npm-asset',
13 ],
14 'components' => [
15     'request' => [
16         // !!! insert a secret key in the following (if it is empty)
17         'cookieValidationKey' => 'randomstring',
18     ],

```

- Настроить подключение к вашей базе данных в файле **config/db.php** (учетные данные индивидуальны у каждого):

```

1 <?php
2
3 return [
4     'class' => 'yii\db\Connection',
5     'dsn'   => 'mysql:host=localhost;dbname=login_m1',
6     'username' => 'user',
7     'password' => 'pass',
8     'charset' => 'utf8',

```

- Активировать кодагенератор Gii в файле **config/web.php**:


```
web.php
64 // 'allowedIPs' => ['127.0.0.1', '::1'],
65 ];
66
67 $config['bootstrap'][] = 'gii';
68 $config['modules']['gii'] = [
69     'class' => 'yii\gii\Module',
70     // uncomment the following to add your IP
71     'allowedIPs' => ['*'],
72 ];
73 }
```

5. Настроить локализацию в файле **config/web.php**:

```
web.php
1 <?php
2
3 $params = require __DIR__ . '/params.php';
4 $db = require __DIR__ . '/db.php';
5
6 $config = [
7     'id' => 'basic',
8     'language' => 'ru-RU',
```

Настройка ЧПУ

Дано

Первоначально настроенный фреймворк

Надо

Настроить человекопонятные URL

Алгоритм

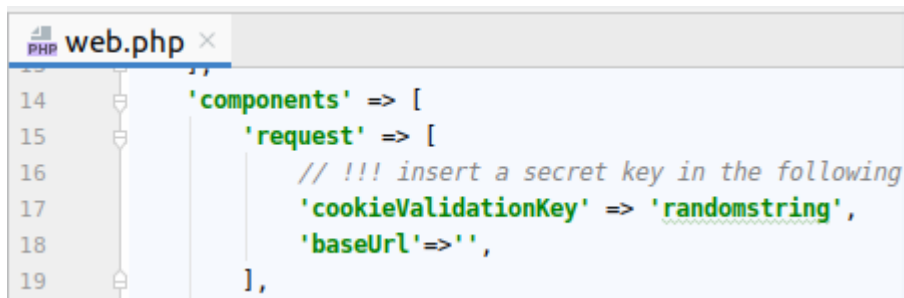
1. Скопировать файл **.htaccess** из директории **web** в корневую директорию модуля
2. Поменять настройки в скопированном файле:

```
.htaccess
1 RewriteEngine on
2 RewriteRule ^(.+)?$ /web/$1
```

или даже так

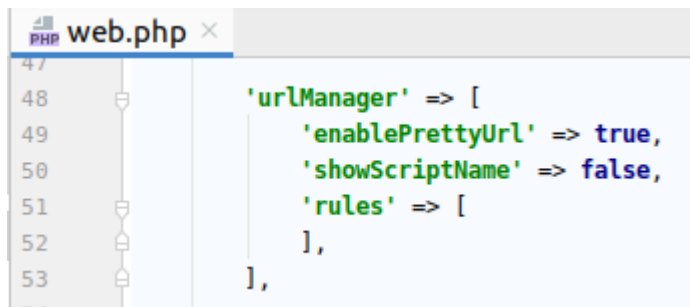
```
.htaccess
1 RewriteEngine on
2 RewriteRule (.+)?$ /web/$1
```

3. Добавить настройку **baseUrl** в файл **config/web.php**:



```
14 'components' => [  
15     'request' => [  
16         // !!! insert a secret key in the following  
17         'cookieValidationKey' => 'randomstring',  
18         'baseUrl'=> '',  
19     ],  
]
```

4. Раскомментировать **urlManager** в файл **config/web.php**:



```
47  
48 'urlManager' => [  
49     'enablePrettyUrl' => true,  
50     'showScriptName' => false,  
51     'rules' => [  
52     ],  
53 ],  
54
```

Создание локального репозитория

Дано

Настроенное рабочее место и фреймворк.

Надо

Создать локальный репозиторий на сервере и несколько коммитов

Алгоритм

1. Подключиться к серверу через **ssh** (учетные данные индивидуальны у каждого): **ssh login@192.168.14.26**
2. Перейти в текущий модуль: **cd module_m4**
3. Выполнить команду инициализации репозитория (если он не создавался автоматически): **git init**
4. Добавить все изменения в текущий индекс: **git add .**
5. Добавить новый коммит: **git commit -m "Реализация всех базовых функций"**

Примечание: делать коммиты (пункты 4-5) лучше по мере добавления новых функций в систему, для того чтобы осталась история изменений

3. Создание структуры и наполнение базы данных

Создание структуры БД

Дано

Настроенное рабочее место и фреймворк

Надо

Создать базу данных для выполнения задания

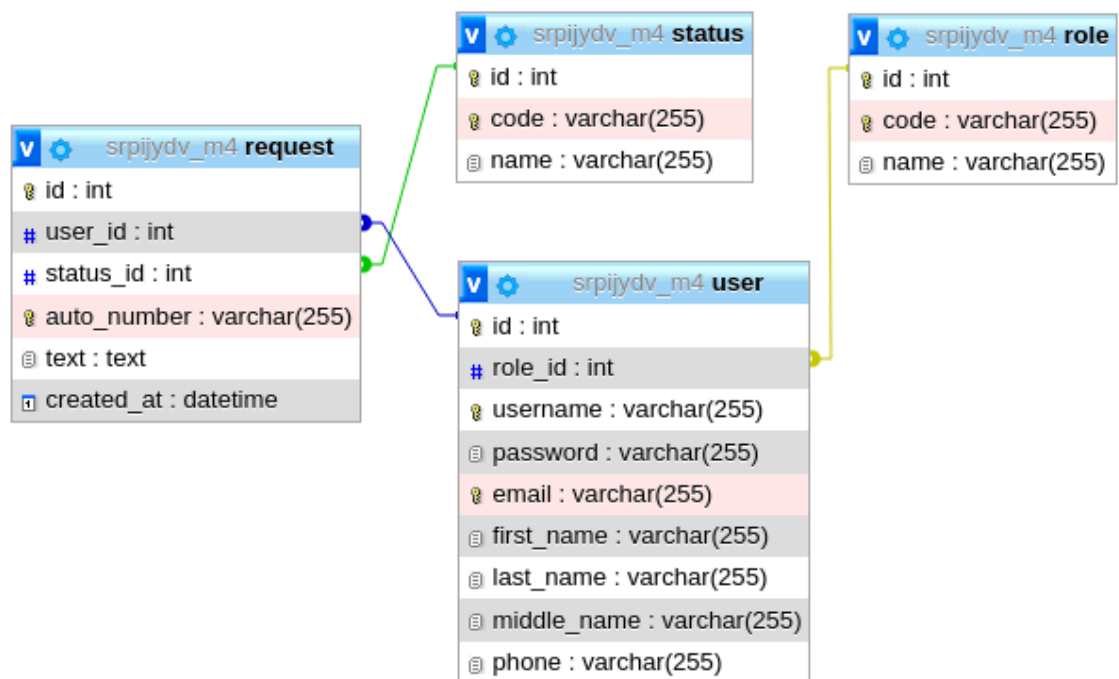
Алгоритм

1. Внимательно прочитать задание, определить основные необходимые сущности. Для задания про портал «НарушениямНет» по приему заявок по фиксации нарушений правил дорожного движения можно определить следующие основные сущности: **Пользователь**, **Роль пользователя**, **Заявка**, **Статус заявки**.
2. Еще раз внимательно прочитать задание, определить основные свойства выявленных сущностей и их связи друг с другом.

Сущность	Поле	Тип
user	id	счетчик
	username	строка, уникальный
	password	строка
	email	строка, уникальный
	first_name	строка
	last_name	строка
	middle_name	строка
	role_id	идентификатор роли
Сущность	Поле	Тип
role	id	счетчик
	code	строка, уникальный
	name	строка
Сущность	Поле	Тип
request	id	счетчик
	user_id	идентификатор пользователя

	auto_number	строка
	text	текст
	дата и время	датоатайм, таймстамп
	status_id	идентификатор статуса
Сущность	Поле	Тип
status	id	счетчик
	code	строка, уникальный
	name	строка

3. Используя phpmyadmin реализовать схему базы данных, описанную в предыдущих пунктах:



Создание структуры БД с помощью миграций

Дано

Настроенное рабочее место и фреймворк. Спроектированная схема базы данных

Надо

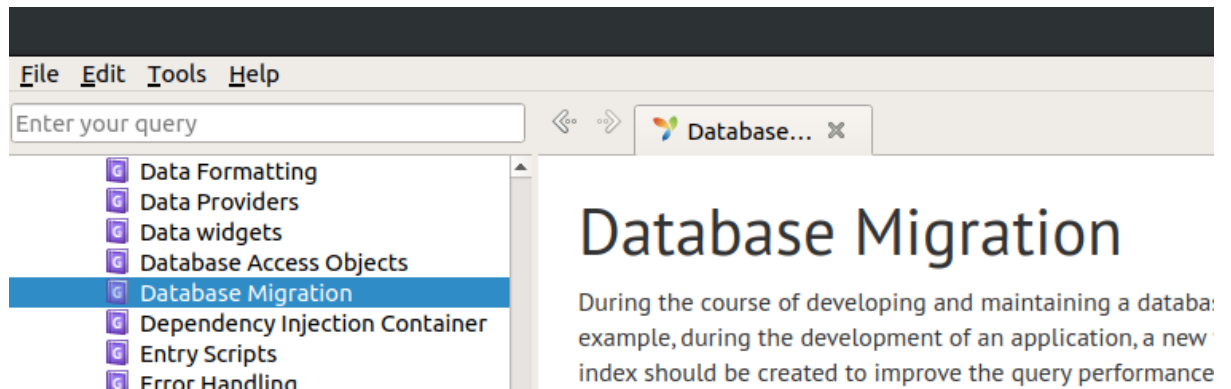
Создать базу данных для выполнения задания с помощью миграций

Алгоритм

1. Подключиться к серверу через **ssh** (учетные данные индивидуальны у каждого):

```
>ssh chfztppz@192.168.13.244
```
2. Перейти в текущий модуль:

```
cd chfztppz-m4
```
3. Открыть статью **Yii - Database Migration** в оффлайн справочнике **Zeal** (оттуда будут браться примеры кода):



4. Вставить в консоль команду создания шаблона миграции для таблицы Роли:

```
php yii migrate/create create_role_table
```
5. Отредактировать появившийся файл с миграцией в директории **migrations** в соответствии с разработанной схемой таблицы **role** (примеры нужного кода можно взять в статье **Database Migration**):

```
m240117_114325_create_role_table.php x
no usages
8 class m240117_114325_create_role_table extends Migration
9 {
10     /**
11      * {@inheritdoc}
12      */
13     no usages
14     public function safeUp()
15     {
16         $this->createTable( table: '{{%role}}', [
17             'id' => $this->primaryKey(),
18             'code' => $this->string()->notNull()->unique(),
19             'name' => $this->string()->notNull(),
20         ]);
21
22         $this->insert( table: '{{%role}}',
23             [
24                 'code' => 'user',
25                 'name' => 'Пользователь',
26             ]
27         );
28
29         $this->insert( table: '{{%role}}',
30             [
31                 'code' => 'admin',
32                 'name' => 'Administrator',
33             ]
34         );
35     }
36
37     /**
38      * {@inheritdoc}
39      */
40     no usages
41     public function safeDown()
42     {
43         $this->dropTable( table: '{{%role}}');
44     }
45 }
```

6. Используя код из статьи Database Migration

```
Database... x
Foreign keys
Since 2.0.8 the generator supports foreign keys using the foreignKey keyword.
te/create create_post_table --fields="author_id:integer:NotNull:foreignKey(user),category_id:integer:defaultValue(1):foreig
```

по аналогии создать файл миграции для таблицы Пользователи:

php yii migrate/create create_user_table

--fields="id_role:integer:NotNull:foreignKey(role)"

7. Отредактировать появившийся файл в соответствии с разработанной схемой таблицы user:

```
m240117_115159_create_user_table.php x
16 public function safeUp()
17 {
18     $this->createTable( table: '{{%user}}', [
19         'id' => $this->primaryKey(),
20         'role_id' => $this->integer()->defaultValue( default: 1),
21         'username' => $this->string()->notNull()->unique(),
22         'password' => $this->string()->notNull(),
23         'email' => $this->string()->notNull()->unique(),
24         'first_name' => $this->string()->notNull(),
25         'last_name' => $this->string()->notNull(),
26         'middle_name' => $this->string()->notNull(),
27         'phone' => $this->string()->notNull(),
28     ]);
29     // creates index for column `role_id`
30     $this->createIndex(
31         name: '{{%idx-user-role_id}}',
32         table: '{{%user}}',
33         columns: 'role_id'
34     );
35     // add foreign key for table '{{%role}}'
36     $this->addForeignKey(
37         name: '{{%fk-user-role_id}}',
38         table: '{{%user}}',
39         columns: 'role_id',
40         refTable: '{{%role}}',
41         refColumns: 'id',
42         delete: 'CASCADE'
43     );
44     $this->insert( table: '{{%user}}', [
45         'role_id' => 2,
46         'username' => 'copp',
47         'password' => md5( string: 'password'),
48         'email' => 'admin@admin.ru',
49         'first_name' => 'Иван',
50         'last_name' => 'Иванов',
51         'middle_name' => 'Иванович',
52         'phone' => '+7(913)-999-99-99',
53     ]);
54 }
```

8. По аналогии сделать миграцию для таблицы Статус заявки:

```
m240117_114736_create_status_table.php ×
no usages
13 public function safeUp()
14 {
15     $this->createTable( table: '{{%status}}', [
16         'id' => $this->primaryKey(),
17         'code' => $this->string()->notNull()->unique(),
18         'name' => $this->string()->notNull(),
19     ]);
20
21     $this->insert( table: '{{%status}}',
22     [
23         'code' => 'new',
24         'name' => 'Новое',
25     ]);
26
27     $this->insert( table: '{{%status}}',
28     [
29         'code' => 'approve',
30         'name' => 'Подтверждено',
31     ]
32 );
33
34     $this->insert( table: '{{%status}}',
35     [
36         'code' => 'rejected',
37         'name' => 'Отклонено',
38     ]
39 );
}
```

9. Сделать миграцию для таблицы Заявки:


```

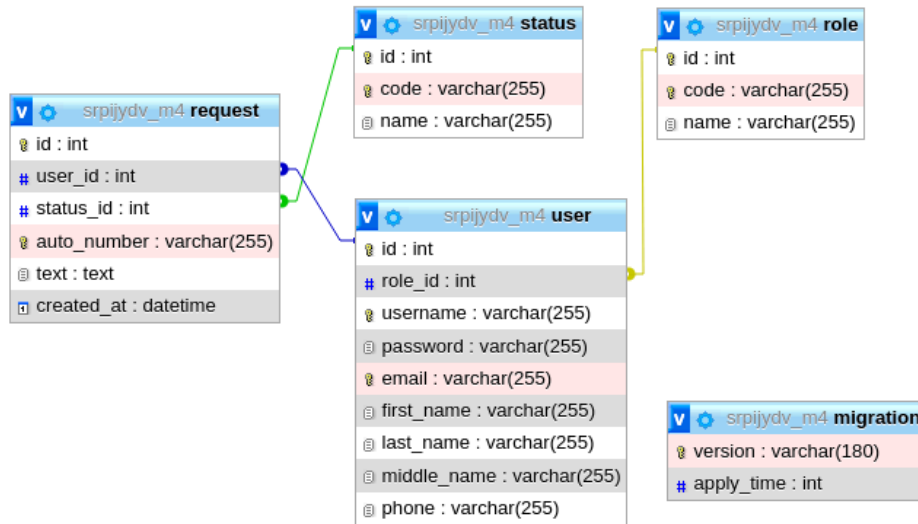
m240117_115552_create_request_table.php x
17 public function safeUp()
18 {
19     $this->createTable( table: '{{%request}}', [
20         'id' => $this->primaryKey(),
21         'user_id' => $this->integer(),
22         'status_id' => $this->integer()->defaultValue( default: 1),
23         'auto_number' => $this->string()->notNull()->unique(),
24         'text' => $this->text(),
25         'created_at' => $this->dateTime(),
26     ]);
27     // creates index for column `user_id`
28     $this->createIndex(
29         name: '{{%idx-request-user_id}}',
30         table: '{{%request}}',
31         columns: 'user_id'
32     );
33     // add foreign key for table `{{%user}}`
34     $this->addForeignKey(
35         name: '{{%fk-request-user_id}}',
36         table: '{{%request}}',
37         columns: 'user_id',
38         refTable: '{{%user}}',
39         refColumns: 'id',
40         delete: 'CASCADE'
41     );
42     // creates index for column `status_id`
43     $this->createIndex(
44         name: '{{%idx-request-status_id}}',
45         table: '{{%request}}',
46         columns: 'status_id'
47     );
48     // add foreign key for table `{{%status}}`
49     $this->addForeignKey(
50         name: '{{%fk-request-status_id}}',
51         table: '{{%request}}',
52         columns: 'status_id',
53         refTable: '{{%status}}',
54         refColumns: 'id',
55         delete: 'CASCADE'
56     );
57 }

```

10. Применить все миграции:

```
$ php yii migrate
```

11. Убедиться, что базы данных разработана в соответствии с заданием:



4. Реализация базовых алгоритмов

Аутентификация

Дано

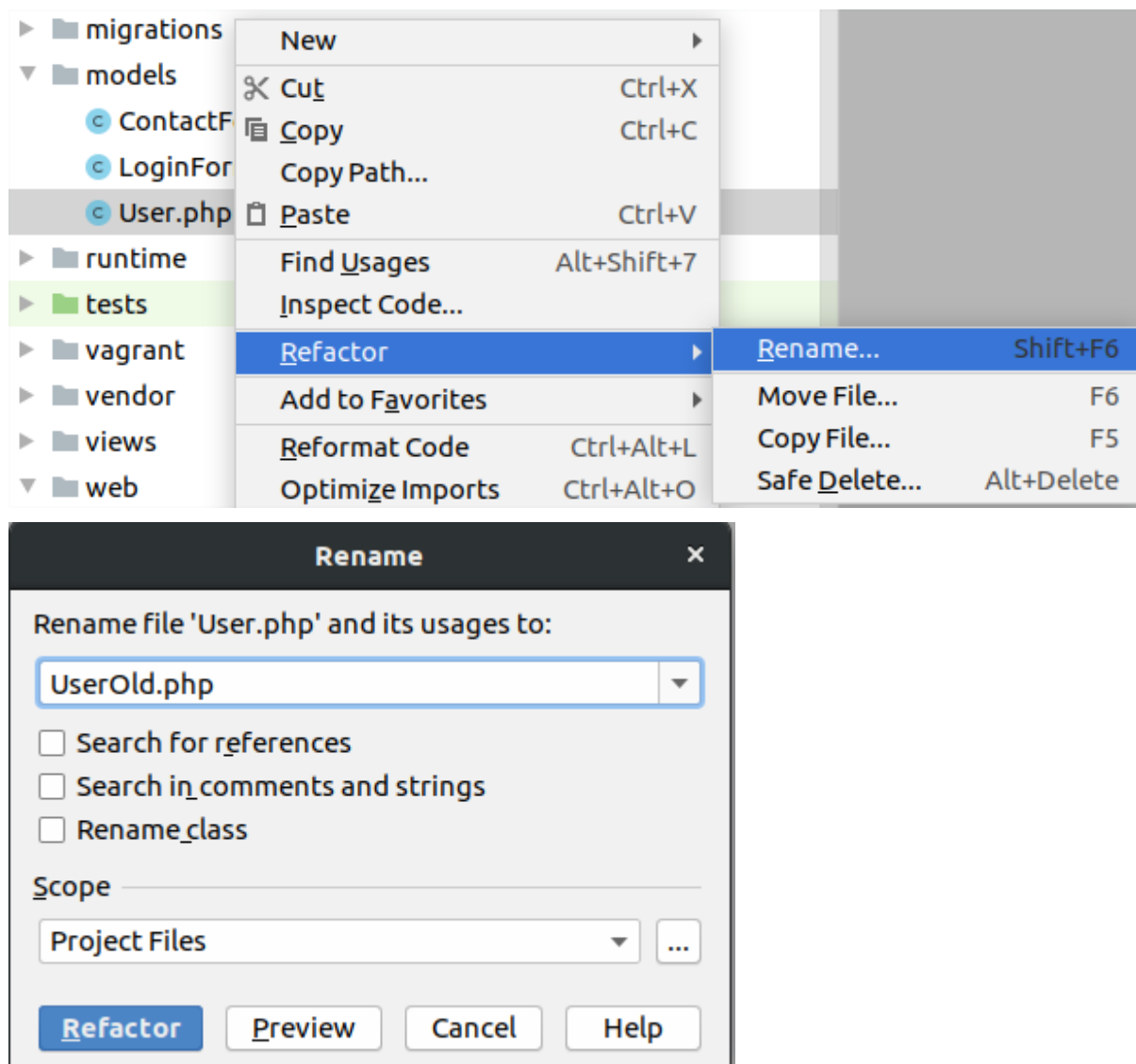
Настроенное рабочее место и фреймворк. Созданная база данных со связями.

Надо

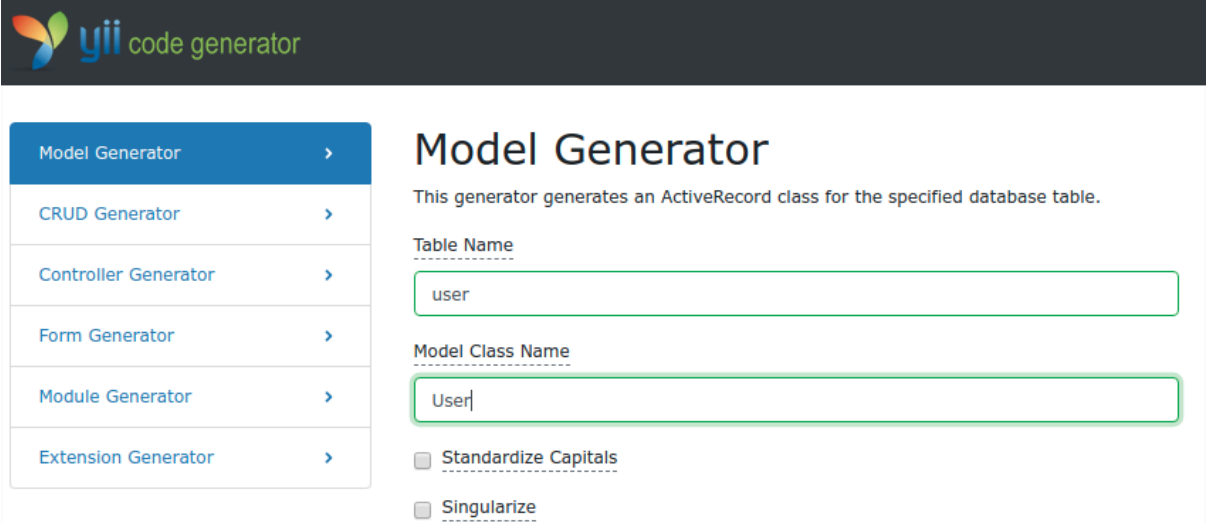
Реализовать алгоритм входа на сайт по логину и паролю

Алгоритм

1. Переименовать стандартный файл модели **User** в **UserOld**



2. Перейти в кодогенератор **Gii**(адрес вида **login-m1.wsr.ru/gii**) и сгенерировать модель **User**:



yii code generator

Model Generator

This generator generates an ActiveRecord class for the specified database table.

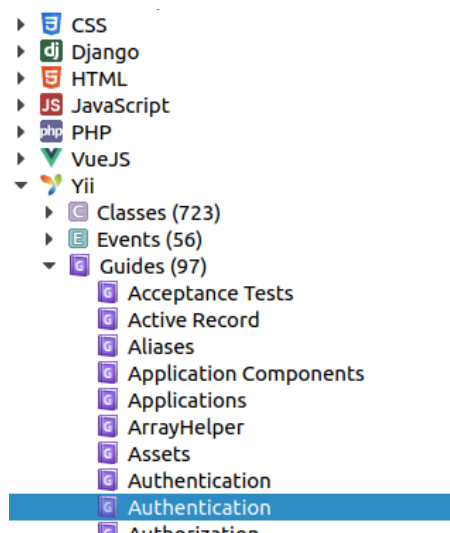
Table Name:

Model Class Name:

☐ Standardize Capitals

☐ Singularize

3. В оффлайн справке **Zeal** открыть статью **Authentication**, посвященную реализации **аутентификации**:



Authentication

Authentication is the process of verifying the identity of a user. It usually involves an access token) to judge if the user is the one whom he claims as. Authentication

Yii provides an authentication framework which wires up various components.

- Configure the [user](#) application component;
- Create a class that implements the [yii\web\IdentityInterface](#) interface.

Configuring [yii\web\User](#)

The [user](#) application component manages the user authentication status. In the following application configuration, the [identity class](#) for [user](#) is configured.

```
return [
```

4. Скопировать в файл **models/User.php** пример реализации интерфейса **IdentityInterface** из статьи **Authentication** (для ненужного функционала добавить заглашки):

```
User.php x
19
20 class User extends \yii\db\ActiveRecord implements \yii\web\IdentityInterface
21 {
22     public static function findIdentity($id)
23     {
24         return static::findOne($id);
25     }
26
27     public static function findIdentityByAccessToken($token, $type = null)
28     {
29         return null;
30     }
31
32     public function getId()
33     {
34         return $this->id;
35     }
36
37     public function getAuthKey()
38     {
39         return null;
40     }
41
42     public function validateAuthKey($authKey)
43     {
44         return false;
45     }
46 }
```

5. Из файла **models/UserOld.php** скопировать функции **findByUsername** и **validatePassword**. Функцию **findByUsername** отредактировать для поиска пользователя по логину (код также можно взять из статьи **Authentication**):

```
Authentic... x
$request = Yii::$app->user->request;
```

To login a user, you may use the following code:

```
// find a user identity with the specified username.
// note that you may want to check the password if needed
$identity = User::findOne(['username' => $username]);
```

```
User.php x UserOld.php x
51
52 public static function findByUsername($username)
53 {
54     return User::findOne(['username' => $username]);
55 }
56
57 public function validatePassword($password)
58 {
59     return $this->password === md5($password);
60 }
61
```

6. Русифицировать ссылку на страницу входа (файл **views/layouts/main.php**):

```
main.php x
45 Yii::$app->user->isGuest ? (
46     ['label' => 'Вход', 'url' => ['/site/login']]
47 ) : (
48     '<li>'
49     . Html::beginForm(['/site/logout'], method: 'post')
50     . Html::submitButton(
51         content: 'Выход (' . Yii::$app->user->identity->username . ')',
52         ['class' => 'btn btn-link logout']
53     )
54 )
```

Базовая регистрация

Дано

Настроенное рабочее место и фреймворк. Созданная база данных со связями.

Надо

Реализовать базовый алгоритм регистрации

Алгоритм

1. Перейти в кодогенератор **Gii**(адрес вида **login-m1.wsr.ru/gii**) и сгенерировать **CRUD** для модели **User**:

CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) data model.

Model Class

Search Model Class

Controller Class

Preview

Generate

Click on the above **Generate** button to generate the files selected below:

☒ Create☒ Unchanged☒ Overwrite

Code File	Action	
controllers/UserController.php	create	<input checked="" type="checkbox"/>
views/user/_form.php	create	<input checked="" type="checkbox"/>
views/user/create.php	create	<input checked="" type="checkbox"/>
views/user/index.php	create	<input type="checkbox"/>
views/user/update.php	create	<input type="checkbox"/>
views/user/view.php	create	<input type="checkbox"/>

2. Сгенерировать все остальные модели по таблицам:

Model Generator

This generator generates an ActiveRecord class for the specified database table.

Database Connection ID

db

☐ Use Table Prefix

☒ Use Schema Name

Table Name

*

This indicates whether the generated class names should have standardized capitals. For example, table names like **SOME_TABLE** or

Preview

Generate

Click on the above **Generate** button to generate the files selected below:

Type to filter

☒ Create

☒ Unchanged

☒ Overwrite

Code File	Action	<input type="checkbox"/>
models/Migration.php	create	<input type="checkbox"/>
models/Request.php	create	<input checked="" type="checkbox"/>
models/Role.php	create	<input checked="" type="checkbox"/>
models/Status.php	create	<input checked="" type="checkbox"/>
models/User.php diff	overwrite	<input type="checkbox"/>

3. В оффлайн справке **Zeal** открыть статью **Yii - Core Validators** (из нее будут браться некоторые примеры кода):

Enter your query

Core Validators

Authentication

Authorization

Behaviors

Bootstrapping

Caching

Class Autoloading

Collecting tabular input

Components

Configurations

Console applications

Controllers

Controllers

Core Validators

Creating Forms

Creating your own Applica...

Cryptography

Data Caching

Data Formatting

Data Providers

Core Validators

Yii provides a set of commonly used core validators, found primarily under the `yii\validators` namespace to specify the use of these core validators. For example, you can use the alias `required` to re

```
public function rules()
{
    return [
        [['email', 'password'], 'required'],
    ];
}
```

The `yii\validators\Validator::$builtInValidators` property declares all supported validator aliases. In the following, we will describe the main usage and properties of every core validator.

4. Из статьи скопировать правило **default** для поля **role** модели **models/User.php**:


```

User.php x
37 public function rules()
38 {
39     return [
40         [['role_id'], 'integer'],
41         [['role_id'], 'default', 'value' => 1],
42         [['username', 'password', 'email', 'first_name', 'last_name', 'middle_name', 'phone'], 'required'],
43         [['username', 'password', 'email', 'first_name', 'last_name', 'middle_name', 'phone'], 'string', 'max' => 255],
44         [['username'], 'unique'],
45         [['email'], 'unique'],
46         [['role_id'], 'exist', 'skipOnError' => true, 'targetClass' => Role::class, 'targetAttribute' => ['role_id' => 'id']],
47     ];
48 }

```

5. Перед сохранением нужно, чтобы пароль хешировался. Для этого в файле **models/User.php** нужно переопределить метод **beforeSave**:

```

User.php x
62
63 public function beforeSave($insert)
64 {
65     $this->password = md5($this->password);
66     return parent::beforeSave($insert); // TODO: Change the autogenerated stub
67 }

```

6. Русифицировать подписи к полям модели User:

```

User.php x
53 public function attributeLabels()
54 {
55     return [
56         'id' => 'ID',
57         'username' => 'Логин',
58         'password' => 'Пароль',
59         'email' => 'Email',
60         'first_name' => 'Имя',
61         'last_name' => 'Фамилия',
62         'middle_name' => 'Отчество',
63         'phone' => 'Телефон',
64     ];
65 }

```

7. Отредактировать файл **views/user/_form.php** убрав лишние поля:

```

php_form.php x
1  <?php
2
3  > use ...
4
5
6  /** @var yii\web\View $this */
7  /** @var app\models\User $model */
8  /** @var yii\widgets\ActiveForm $form */
9  ?>
10
11 <div class="user-form">
12
13     <?php $form = ActiveForm::begin(); ?>
14
15     <?= $form->field($model, 'username')->textInput(['maxlength' => true]) ?>
16
17     <?= $form->field($model, 'password')->passwordInput(['maxlength' => true]) ?>
18
19     <?= $form->field($model, 'email')->textInput(['maxlength' => true]) ?>
20
21     <?= $form->field($model, 'first_name')->textInput(['maxlength' => true]) ?>
22
23     <?= $form->field($model, 'last_name')->textInput(['maxlength' => true]) ?>
24
25     <?= $form->field($model, 'middle_name')->textInput(['maxlength' => true]) ?>
26
27     <?= $form->field($model, 'phone')->textInput(['maxlength' => true]) ?>
28
29     <div class="form-group">
30         <?= Html::submitButton('Зарегистрироваться', ['class' => 'btn btn-primary']) ?>
31     </div>
32
33     <?php ActiveForm::end(); ?>
34
35 </div>

```

8. Изменить редирект и при успешной регистрации в файле **controllers/UserController.php**:

```

UserController.php x
78 public function createAction()
79 {
80     $model = new User();
81
82     if ($this->request->isPost) {
83         if ($model->load($this->request->post()) && $model->save()) {
84             return $this->redirect(['site/login']);
85         }
86     } else {
87         $model->loadDefaultValues();
88     }
89
90     return $this->render( view: 'create', [
91         'model' => $model,
92     ]);
93 }

```

- Добавить ссылку на регистрацию в главное меню (файл `views/layouts/main.php`)

```

main.php x
37 });
38 echo Nav::widget([
39     'options' => ['class' => 'navbar-nav navbar-right'],
40     'items' => [
41         ['label' => 'Home', 'url' => ['/site/index']],
42         ['label' => 'About', 'url' => ['/site/about']],
43         ['label' => 'Contact', 'url' => ['/site/contact']],
44         ['label' => 'Регистрация', 'url' => ['/user/create'], 'visible'=>Yii::$app->user->isGuest],

```

Валидация при регистрации

Дано

Настроенное рабочее место и фреймворк. Созданная база данных со связями.
Реализованная функция базовой регистрации

Надо

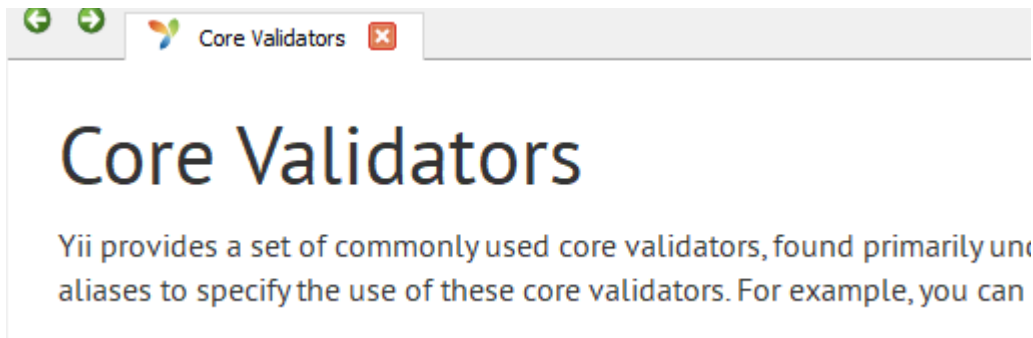
Реализовать валидации по заданию для функции регистрации

Алгоритм

- Согласно заданию необходимо реализовать следующие валидации полей регистрации:
 - ФИО - только кириллические буквы и пробелы;
 - Логин – только латиница, уникальный;
 - Email - валидный формат email-адрес, уникальный;
 - Пароль - минимум 6 символов;

- Телефон - формат +7(XXX)-XXX-XX-XX.

Открыть статью **Yii - Core Validators** в Zeal:



2. Добавить валидации полей в класс **models/User.php** (примеры кода можно **взять из статьи**):

Примечание: паттерн телефона +7(XXX)-XXX-XX-XX

`/^\+?7(\d{3})-\d{3}-\d{2}-\d{2}$/`



3. Отредактировать представление **views/user/create.php** чтобы русифицировать страницу:

```
PHP create.php x
7
8     $this->title = 'Регистрация';
9     $this->params['breadcrumbs'][] = $this->title;
10    ?>
11    <div class="user-create">
12
13        <h1><?= Html::encode($this->title) ?></h1>
14
15        <?= $this->render( view: '_form', [
16            'model' => $model,
17        ]) ?>
```

5. Реализация специфических алгоритмов из задания

Организация личного кабинета пользователя

Дано

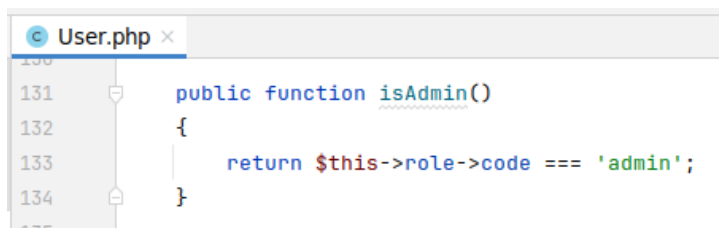
Настроенное рабочее место и фреймворк. Созданная база данных со связями. Реализованная аутентификация (вход на сайт). Созданы все модели для таблиц базы данных.

Надо

Авторизованного пользователя перенаправляет на личный кабинет с персональными заявками. Также необходимо ограничивать доступ по прямой ссылке на действие.

Алгоритм

1. Добавить функцию **isAdmin** в модель **User** (**models/User.php**)



```
131 public function isAdmin()  
132 {  
133     return $this->role->code === 'admin';  
134 }  
135
```

2. Используя Gii создать CRUD для модели Request

CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for data model.

Model Class

Search Model Class

Controller Class

View Path

Base Controller Class

Личный кабинет будет реализован из сгенерированной страницы, отображающей список заявок (адрес /request)

My Application Home About Contact Регистрация Login						
Главная / Requests						
Requests						
Create Request						
#	ID	User ID	Status ID	Auto Number	Text	
Ничего не найдено.						

- Для ограничения доступа к этой странице только зарегистрированным пользователям необходимо изменить код контроллера **RequestController**. Код можно взять из статьи **Authorization** раздел **Access Control Filter** в справке **Zeal** или в коде контроллера **SiteController**

```
16 class RequestController extends Controller
17 {
18     /** {@inheritdoc} ... */
21     public function behaviors()
22     {
23         return [
24             'access' => [
25                 'class' => AccessControl::className(),
26                 'only' => ['*'],
27                 'rules' => [
28                     [
29                         'actions' => ['index', 'create'],
30                         'allow' => true,
31                         'roles' => ['@'],
32                     ],
33                 ],
34             ],
35             'verbs' => [
36                 'class' => VerbFilter::className(),
37                 'actions' => [
38                     'delete' => ['POST'],
39                 ],
40             ],
41         ];
42     }
```

- Для реализации перенаправления нужно изменить код контроллера **SiteController**

```
SiteController.php x
79 public function actionLogin()
80 {
81     if (!Yii::$app->user->isGuest) {...}
84
85     $model = new LoginForm();
86     if ($model->load(Yii::$app->request->post()) && $model->login()) {
87         if (!Yii::$app->user->identity->isAdmin()) {
88             return $this->redirect('url: '/request');
89         }
90         return $this->goBack();
91     }
92
93     $model->password = '';
94     return $this->render('view: 'login', [...]);
97 }
```

5. Добавить пункт меню для зарегистрированного пользователя (файл views/layouts/main.php)

```
main.php x
43 ['label' => 'Contact', 'url' => ['/site/contact']],
44 ['label' => 'Регистрация', 'url' => ['/user/create'], 'visible'=>Yii::$app->user->isGuest],
45 ['label' => 'Личный кабинет', 'url' => ['/request/index'], 'visible'=>!Yii::$app->user->isGuest],
46 Yii::$app->user->isGuest ? (
47     ['label' => 'Вход', 'url' => ['/site/login']]
48 ) : (
49     '<li>'
50     . Html::beginForm(['/site/logout'], 'method: 'post')
51     . Html::submitButton(
52         content: 'Выход (' . Yii::$app->user->identity->username . ')',
```

6. На основе раздела Working with Data Keys статьи Data Providers изменяем код выборки заявок

Enter your query

Data Caching

Data Formatting

Data Providers

Data widgets

Database Access Objects

Database Migration

Dependency Injection Con...

Entry Scripts

Error Handling

Events

Extending ActiveForm on t...

Extensions

Filters

Fixtures

Fragment Caching

Functional Tests

Generating Code with GII

Working with Data Keys

When using the data items returned by a data provider, you often need to identify each data i information, you may want to use the customer ID as the key for each customer data. Data pr by [yii\data\DataProviderInterface::getModels\(\)](#). For example,

```
use yii\data\ActiveDataProvider;

$query = Post::find()->where(['status' => 1]);

$provider = new ActiveDataProvider([
    'query' => $query,
]);
```



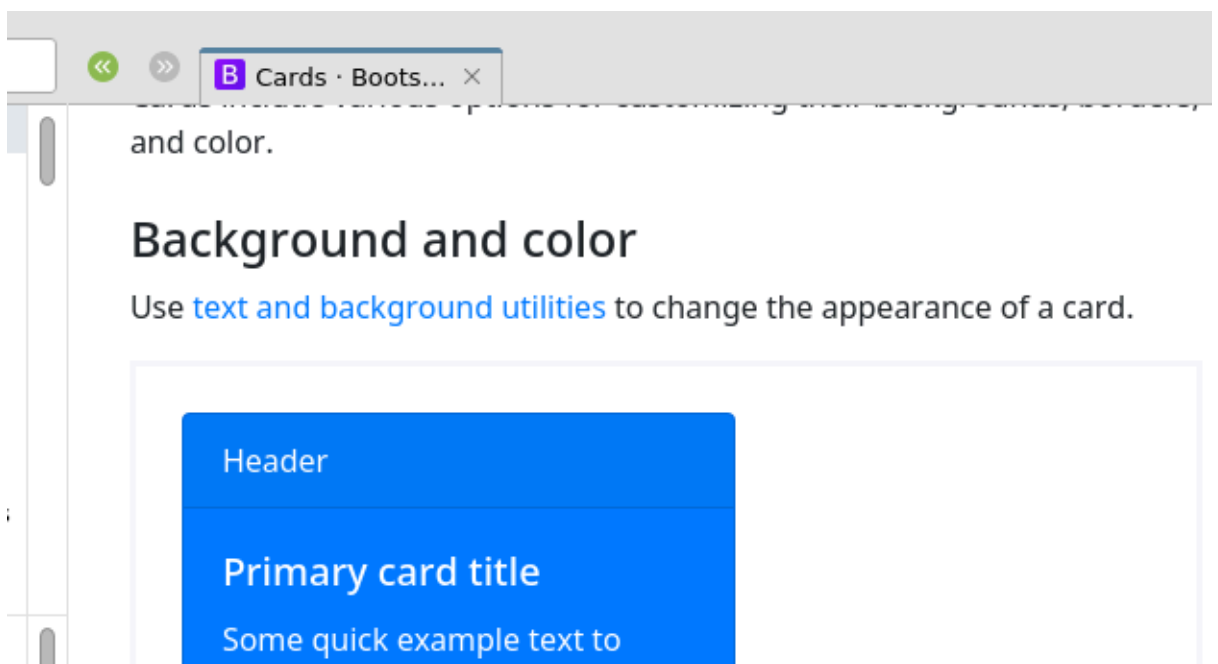
```

RequestController.php x
50  */
51  public function actionIndex()
52  {
53      $dataProvider = new ActiveDataProvider([
54          'query' => Request::find()->where(['user_id' => \Yii::$app->user->identity->getId()]),
55
56          'pagination' => [
57              'pageSize' => 5
58          ],
59          'sort' => [
60              'defaultOrder' => [
61                  'created_at' => SORT_DESC,
62              ]
63          ],
64      ]);
65  };
66
67  return $this->render( view: 'index', [
68      'dataProvider' => $dataProvider,
69  ]);
70  }

```

Теперь в личном кабинете обычного пользователя отображаются заявки только которые он подавал

- Используя код из статьи **Cards** документации **Bootstrap 4** редактируем вид таблицы заявок



```
index.php x
21 <?php
22 foreach ($dataProvider->models as $model){
23
24     $classCard = match ($model->status->code) {
25         'approve' => 'text-white bg-success',
26         'rejected' => 'text-white bg-danger',
27         'new' => 'bg-light',
28     }
29
30     ?>
31     <div class="card mb-2 <?= $classCard ?>" >
32         <div class="card-body">
33             <h5 class="card-title"><?= $model->auto_number ?></h5>
34             <h6 class="card-subtitle mb-2 text-muted"><?= $model->status->name ?></h6>
35             <p class="card-text"><?= $model->text ?></p>
36         </div>
37     </div>
38 <?php }
39
40 echo LinkPager::widget([
41     'pagination' => $dataProvider->pagination,
42 ]);
43
44 ?>
45
```

Добавление новой заявки пользователем

Дано

Настроенное рабочее место и фреймворк. Созданная база данных со связями. Реализованная аутентификация (вход на сайт). Созданы все модели для таблиц базы данных. Создан базовый CRUD для модели Request.

Надо

Организовать возможность авторизованному пользователю оставлять заявку в соответствии с заданием

Создание заявки (все поля обязательны):

- Государственный регистрационный номер;
- Описание нарушения.

Алгоритм

1. В файле **views/request/_form.php** удалить лишние поля (недоступные для заполнения пользователем)

```
1 <?php
2
3 use ...
4
5
6 /** @var yii\web\View $this */
7 /** @var app\models\Request $model */
8 /** @var yii\widgets\ActiveForm $form */
9
10
11 <div class="request-form">
12
13     <?php $form = ActiveForm::begin(); ?>
14
15     <?= $form->field($model, attribute: 'auto_number')->textInput(['maxlength' => true]) ?>
16
17     <?= $form->field($model, attribute: 'text')->textarea(['rows' => 6]) ?>
18
19     <div class="form-group">
20         <?= Html::submitButton( content: 'Подать заявку', ['class' => 'btn btn-success']) ?>
21     </div>
22
23     <?php ActiveForm::end(); ?>
24
25 </div>
```

2. Изменить модель Request (файл **models/Request.php**) для того, чтобы при добавлении заявки идентификатор пользователя добавлялся автоматически. Также нужно удалить из правила required поле **id_user**

```
Request.php
33 public function rules()
34 {
35     return [
36         [['user_id', 'status_id'], 'integer'],
37         [['user_id', 'default', 'value' => \Yii::$app->user->identity->getId()],
38         [['auto_number'], 'required'],
39         [['text'], 'string'],
40         [['created_at'], 'safe'],
41         [['auto_number'], 'string', 'max' => 255],
42         [['auto_number'], 'unique'],
43         [['status_id'], 'exist', 'skipOnError' => true,
44             'targetClass' => Status::class, 'targetAttribute' => ['status_id' => 'id']],
45         [['user_id'], 'exist', 'skipOnError' => true,
46             'targetClass' => User::class, 'targetAttribute' => ['user_id' => 'id']],
47     ];
48 }
```

3. Русифицировать страницу добавления заявки (файл **views/request/create.php**)

```
create.php x
1 <?php
2
3 use yii\helpers\Html;
4
5 /** @var yii\web\View $this */
6 /** @var app\models\Request $model */
7
8 $this->title = 'Оставить новое заявление';
9 $this->params['breadcrumbs'][] = $this->title;
10 ?>
11 <div class="request-create">
12
13     <h1><?= Html::encode($this->title) ?></h1>
14
15     <?= $this->render( view: '_form', [
16         'model' => $model,
17     ]) ?>
18
19 </div>
```

4. Русифицировать подписи к полям формы (файл **models/Request.php**)

```
Request.php x
no usages
53 public function attributeLabels()
54 {
55     return [
56         'id' => 'ID',
57         'user_id' => 'User ID',
58         'status_id' => 'Status ID',
59         'auto_number' => 'Номер автомобиля',
60         'text' => 'Описание нарушения',
61         'created_at' => 'Created At',
62     ];
63 }
```

5. Изменить редирект при создании новой заявки (файл **controllers/RequestController.php**)



```

90 public function actionCreate()
91 {
92     $model = new Request();
93
94     if ($this->request->isPost) {
95         if ($model->load($this->request->post()) && $model->save()) {
96             return $this->redirect(['index']);
97         }
98     } else {
99         $model->loadDefaultValues();
100     }
101
102     return $this->render( view: 'create', [
103         'model' => $model,
104     ]);
105 }

```

Организация панели управления сайтом администратора

Дано

Настроенное рабочее место и фреймворк. Созданная база данных со связями. Реализованная аутентификация (вход на сайт). Созданы все модели для таблиц базы данных.

Надо

Администратора перенаправляет на панель управления сайтом (по адресу /admin) со всеми заявками всех пользователей, необходимо ограничивать доступ по прямой ссылке на действия. Также необходимо реализовать одобрение и отклонение заявки

Алгоритм

1. Используя Gii создать AdminController для панели администрирования сайта

Controller Generator

This generator helps you to quickly generate a new controller class with one or several controller actions.

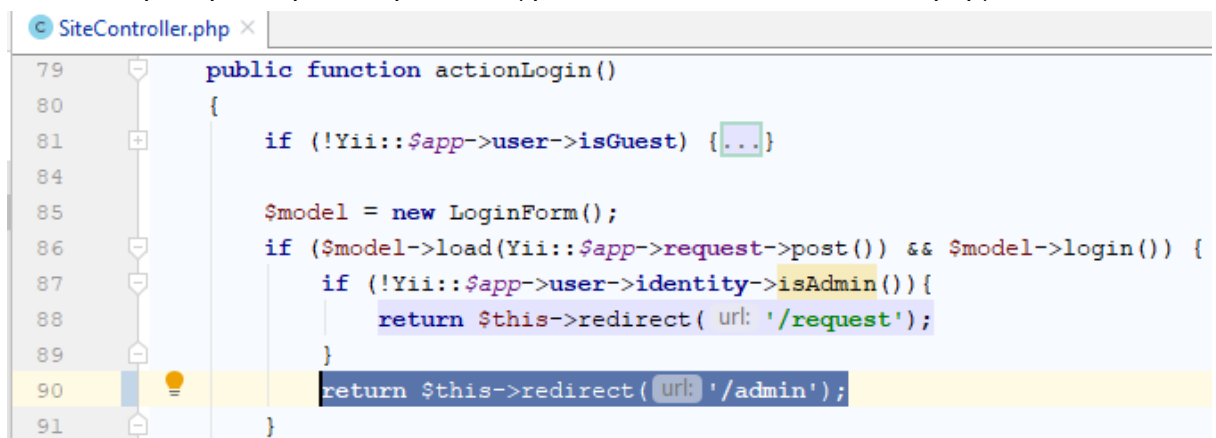
Controller Class

app\controllers\AdminController

Action IDs

index

2. Изменить редирект при авторизации (файл controllers/SiteController.php)



```
79 public function actionLogin()
80 {
81     if (!Yii::$app->user->isGuest) {...}
84
85     $model = new LoginForm();
86     if ($model->load(Yii::$app->request->post()) && $model->login()) {
87         if (!Yii::$app->user->identity->isAdmin()) {
88             return $this->redirect( url: '/request');
89         }
90         return $this->redirect( url: '/admin');
91     }
```

3. Используя код из статьи Authorization ограничить доступ к **Admin Controller** только для Администратора (файл /controller/AdminController.php)

denyCallback: specifies a PHP callable that should be called when this rule will deny the access.

Below is an example showing how to make use of the **matchCallback** option, which allows you to write

```
use yii\filters\AccessControl;

class SiteController extends Controller
{
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['special-callback'],
                'rules' => [
                    [
                        'actions' => ['special-callback'],
                        'allow' => true,
                        'matchCallback' => function ($rule, $action) {
                            return date('d-m') === '31-10';
                        }
                    ]
                ]
            ],
        ],
    }
}
```

AdminController.php

```
10 class AdminController extends \yii\web\Controller
11 {
12     public function behaviors()
13     {
14         return [
15             'access' => [
16                 'class' => AccessControl::className(),
17                 'only' => ['*'],
18                 'rules' => [
19                     [
20                         'actions' => ['index'],
21                         'roles' => ['@'],
22                         'allow' => true,
23                         'matchCallback' => function ($rule, $action) {
24                             return Yii::$app->user->identity->isAdmin();
25                         }
26                     ],
27                 ],
28             ],
29         ];
30     }
}
```

- Страница административной панели будет содержать все заявки всех пользователей. Используя код из контроллера **RequestController** нужно изменить код **AdminController**

```
AdminController.php x
no usages
8 class AdminController extends \yii\web\Controller
9 {
10     public function actionIndex()
11     {
12         $dataProvider = new ActiveDataProvider([
13             'query' => Request::find(),
14
15             'pagination' => [
16                 'pageSize' => 5
17             ],
18             'sort' => [
19                 'defaultOrder' => [
20                     'created_at' => SORT_DESC,
21                 ]
22             ],
23         ]);
24
25         return $this->render( view: 'index', [
26             'dataProvider' => $dataProvider,
27         ]);
28     }
29 }
30
31 }
```

5. Добавить метод для формирования полного имени пользователя в модель User

```
User.php x
no usages
95 public function getFullName()
96 {
97     return $this->last_name." ".$this->first_name." ".$this->middle_name;
98 }
99 }
```

6. Используя представления для заявок пользователя, исправить отображение заявок


```

index.php
13 $this->title = 'Панель администратора';
14 $this->params['breadcrumbs'][] = $this->title;
15 ?>
16 <div class="request-index">
17
18     <h1><?= Html::encode($this->title) ?></h1>
19
20     <?php
21     foreach ($dataProvider->models as $model){
22
23         $classCard = match ($model->status->code) {
24             'approve' => 'text-white bg-success',
25             'rejected' => 'text-white bg-danger',
26             'new' => 'bg-light',
27         }
28
29     ?>
30     <div class="card mb-2 <?= $classCard ?>" >
31         <div class="card-body">
32             <h5 class="card-title"><?= $model->auto_number ?></h5>
33             <h6 class="card-subtitle mb-2 text-muted"><?= $model->status->name ?></h6>
34             <h6 class="card-subtitle mb-2 text-muted"><?= $model->user->getFullName() ?></h6>
35             <p class="card-text"><?= $model->text ?></p>
36
37             <?php if ($model->status->code=='new') {
38                 echo Html::a( text: 'Подтвердить', ['admin/success', 'id' => $model->id], ['class' => 'card-link']);
39                 echo Html::a( text: 'Отклонить', ['admin/cancel', 'id' => $model->id], ['class' => 'card-link']);
40             } ?>
41         </div>
42     </div>
43 <?php }
44
45 echo LinkPager::widget([
46     'pagination' => $dataProvider->pagination,
47 ]);
48
49 ?>
50
51 </div>

```

7. Добавить ссылку на административную панель в меню сайта

```

main.php
44 [ 'label' => 'Регистрация', 'url' => ['/user/create'], 'visible' => Yii::$app->user->isGuest ],
45 [ 'label' => 'Личный кабинет', 'url' => ['/request/index'], 'visible' => !Yii::$app->user->isGuest ],
46 [ 'label' => 'Панель администрирования сайта', 'url' => ['/admin/index'],
47     'visible' => !Yii::$app->user->isGuest && Yii::$app->user->identity->isAdmin() ],
48     Yii::$app->user->isGuest ? (

```

8. Добавить права доступа для действий по изменению статуса заявки

```
AdminController.php x
10 {
11     public function behaviors()
12     {
13         return array_merge(
14             parent::behaviors(),
15             [
16                 'access' => [
17                     'class' => AccessControl::class,
18                     'only' => ['*'],
19                     'rules' => [
20                         [
21                             'allow' => true,
22                             'actions' => ['index', 'success', 'cancel'],
23                             'roles' => ['@'],
24                             'matchCallback' => function ($rule, $action) {
25                                 return \Yii::$app->user->identity->isAdmin();
26                             }
27                         ],
28                     ],
29             ],
30         );
31     }
32 }
```

9. Реализовать функции одобрения и отклонения заявки

```
AdminController.php x
57 public function actionSuccess($id)
58 {
59     $model = $this->findModel($id);
60     if ($model->status->code === 'new') {
61         $model->status_id = Status::findOne(['code' => 'approve'])->id;
62         $model->save();
63     }
64     return $this->redirect(url: 'index');
65 }
66
67 no usages
68 public function actionCancel($id)
69 {
70     $model = $this->findModel($id);
71     if ($model->status->code === 'new') {
72         $model->status_id = Status::findOne(['code' => 'rejected'])->id;
73         $model->save();
74     }
75     return $this->redirect(url: 'index');
76 }
77
78 2 usages
79 protected function findModel($id)
80 {
81     if (($model = Request::findOne(['id' => $id])) !== null) {
82         return $model;
83     }
84     throw new NotFoundException(message: 'The requested page does not exist.');
```

Добавление уведомлений

Дано

Настроенное рабочее место и фреймворк. Созданная база данных со связями. Созданы все модели для таблиц базы данных. Реализованы основные функции системы.

Надо

Добавить уведомления об основных действиях пользователя: успешная регистрация, успешная подача заявки, отклонение или подтверждение заявки у администратора.

Алгоритм

1. Добавить уведомление об успешной регистрации (файл **controllers/UserController.php**)

```

UserController.php x
78 public function actionCreate()
79 {
80     $model = new User();
81
82     if ($this->request->isPost) {
83         if ($model->load($this->request->post()) && $model->save()) {
84             Yii::$app->getSession()->setFlash( key: 'success', value: 'Поздравляем! Вы успешно зарегистрировались!');
85             return $this->redirect(['site/login']);
86         }
87     } else {
88         $model->loadDefaultValues();
89     }
90
91     return $this->render( view: 'create', [
92         'model' => $model,
93     ]);
94 }

```

2. Добавить уведомление об успешной добавлении заявки (файл **controllers/RequestController.php**)

```

RequestController.php x
90 public function actionCreate()
91 {
92     $model = new Request();
93
94     if ($this->request->isPost) {
95         if ($model->load($this->request->post()) && $model->save()) {
96             Yii::$app->getSession()->setFlash( key: 'success', value: 'Заявление зарегистрировано!');
97             return $this->redirect(['index']);
98         }
99     } else {
100         $model->loadDefaultValues();
101     }
102
103     return $this->render( view: 'create', [
104         'model' => $model,
105     ]);
106 }

```

3. Добавить уведомление об одобрении или отклонении заявки (файл **controllers/AdminController.php**)

```

AdminController.php x
57 public function actionSuccess($id)
58 {
59     $model = $this->findModel($id);
60     if ($model->status->code === 'new') {
61         $model->status_id = Status::findOne(['code' => 'approve'])->id;
62         $model->save();
63         \Yii::$app->getSession()->setFlash( key: 'success', value: 'Заявление одобрено!');
64     }
65     return $this->redirect( url: 'index');
66 }
67
68 no usages
69 public function actionCancel($id)
70 {
71     $model = $this->findModel($id);
72     if ($model->status->code === 'new') {
73         $model->status_id = Status::findOne(['code' => 'rejected'])->id;
74         $model->save();
75         \Yii::$app->getSession()->setFlash( key: 'info', value: 'Заявление отклонено!');
76     }
77     return $this->redirect( url: 'index');
78 }

```

Добавление анимаций

Дано

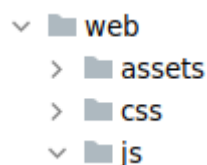
Настроенное рабочее место и фреймворк. Созданная база данных со связями. Созданы все модели для таблиц базы данных. Реализованы основные функции системы. Присутствуют уведомления системы

Надо

Добавить микро анимации на уведомления системы

Алгоритм

1. Создать директорию web/js для хранения пользовательских файлов js



2. В созданную директорию добавить пустой файл main.js




3. В файле assets/AppAsset.php указать созданный файл

```

11
12  /**
13   * Main application asset bundle.
14   *
15   * @author Qiang Xue <qiang.xue@gmail.com>
16   * @since 2.0
17   */
18  class AppAsset extends AssetBundle
19  {
20      public $basePath = '@webroot';
21      public $baseUrl = '@web';
22      public $css = [
23          'css/site.css',
24      ];
25      public $js = [
26          'js/main.js'
27      ];
28      public $depends = [
29          'yii\web\YiiAsset',
30          'yii\bootstrap5\BootstrapAsset'
31      ];
32  }

```

4. Используя код из статьи JQuery->.animate() реализовать код анимации исчезновения информационных сообщений


.animate() x

To animate the opacity, left offset, and height of the image simultaneously:

```

1 | $( "#clickme" ).on( "click", function() {
2 |     $( "#book" ).animate({
3 |         opacity: 0.25,
4 |         left: "+=50",
5 |         height: "toggle"
6 |     }, 5000, function() {
7 |         // Animation complete.
8 |     });
9 | });

```

```

1  $('alert').animate( prop: {
2      opacity: 0.25,
3  }, speed: 1000, easing: function () : void {
4      $('alert').hide('show');
5  });
6

```

Кастомизация дизайна

Дано

Настроенное рабочее место и фреймворк.

Надо

Изменить стандартный дизайн фреймворка в соответствии с предметной областью.

Алгоритм

1. Удалить из меню неиспользуемые страницы (файл **views/layouts/main.php**)

```

32  <header id="header">
33  <?php
34  NavBar::begin([
35      'brandLabel' => Yii::$app->name,
36      'brandUrl' => Yii::$app->homeUrl,
37      'options' => ['class' => 'navbar-expand-md navbar-dark bg-dark fixed-top']
38  ]);
39  echo Nav::widget([
40      'options' => ['class' => 'navbar-nav'],
41      'items' => [
42          ['label' => 'Регистрация', 'url' => ['/user/create'], 'visible' => Yii::$app->user->isGuest],
43          ['label' => 'Личный кабинет', 'url' => ['/request/index'], 'visible' => !Yii::$app->user->isGuest],
44          ['label' => 'Панель администратора', 'url' => ['/admin'],
45              'visible' => !Yii::$app->user->isGuest && Yii::$app->user->identity->isAdmin()],
46          Yii::$app->user->isGuest
47              ? ['label' => 'Login', 'url' => ['/site/login']]
48              : ['<li class="nav-item">
49                  . Html::beginForm(['/site/logout'])
50                  . Html::submitButton(
51                      content: 'Logout (' . Yii::$app->user->identity->username . ')',
52                      ['class' => 'nav-link btn btn-link logout']
53                  )
54                  . Html::endForm()
55                  . '</li>'
56              ]
57          ];
58  NavBar::end();
59  ?>
60  </header>

```

2. В настройках сайта изменить его название в соответствии с заданием и главную страницу по-умолчанию (файл **config/web.php**)

```

web.php x
6 $config = [
7     'id' => 'basic',
8     'basePath' => dirname( path: __DIR__ ),
9     'bootstrap' => ['log'],
10    'language' => 'ru-RU',
11    'defaultRoute' => 'request',
12    'name' => '«Нарушениям.Нет»',
13    'aliases' => [
14        '@bower' => '@vendor/bower-asset',
15        '@npm' => '@vendor/npm-asset',
16    ],

```

3. В директорию **web/css** добавить файл **main.css** и прописать его в файле **assets/AppAsset.php**

```

AppAsset.php x
12 /**
13  * Main application asset bundle.
14  *
15  * @author Qiang Xue <qiang.xue@gmail.com>
16  * @since 2.0
17  */
18 class AppAsset extends AssetBundle
19 {
20     public $basePath = '@webroot';
21     public $baseUrl = '@web';
22     public $css = [
23         'css/site.css',
24         'css/main.css',
25     ];
26     public $js = [

```

4. Поменять цвет навигационной панели (файл **views/layouts/main.php**)

```

main.php x
32 <header id="header">
33     <?php
34     NavBar::begin([
35         'brandLabel' => Yii::$app->name,
36         'brandUrl' => Yii::$app->homeUrl,
37         'options' => ['class' => 'navbar-expand-md navbar-dark bg-primary fixed-top']
38     ]);
39     echo Nav::widget([

```

5. Изменить цвет фона и текста футера


```

main.php x
71
72 <footer id="footer" class="mt-auto py-3 bg-danger ">
73   <div class="container">
74     <div class="row text-muted ">
75       <div class="col-md-6 text-center text-md-start text-white"> «Нарушениям.Нет» <?= date( format: 'Y') ?></div>
76     </div>
77   </div>
78 </footer>

```

6. Заменить цвета всех кнопок с btn-success на btn-primary (регистрации, добавления новой заявки, страница заявок)

```

user/_form.php x
23 <?= $form->field($model, attribute: 'last_name')->textInput(['maxlength' => true]) ?>
24
25 <?= $form->field($model, attribute: 'middle_name')->textInput(['maxlength' => true]) ?>
26
27 <?= $form->field($model, attribute: 'phone')->textInput(['maxlength' => true]) ?>
28
29 <div class="form-group">
30   <?= Html::submitButton( content: 'Зарегистрироваться', ['class' => 'btn btn-primary']) ?>
31 </div>
32

```

```

request/_form.php x
3 <?php $form = ActiveForm::begin(); ?>
4
5 <?= $form->field($model, attribute: 'auto_number')->textInput(['maxlength' => true]) ?>
6
7 <?= $form->field($model, attribute: 'text')->textarea(['rows' => 6]) ?>
8
9 <div class="form-group">
10  <?= Html::submitButton( content: 'Подать заявку', ['class' => 'btn btn-primary']) ?>
11 </div>
12
13 <?php ActiveForm::end(); ?>

```

```

index.php x
11 <?>
12 <div class="request-index">
13
14   <h1><?= Html::encode($this->title) ?></h1>
15
16   <p>
17     <?= Html::a( text: 'Оставить новое заявление', ['create'], ['class' => 'btn btn-primary']) ?>
18   </p>
19

```

7. Русифицировать страницу входа (файл **views/site/login.php**)

```

login.php x
11 $this->title = 'Вход';
12 $this->params['breadcrumbs'][] = $this->title;
13 <?>
14 <div class="site-login">
15 <h1><?= Html::encode($this->title) ?></h1>
16
17 <p>Пожалуйста, заполните данные для входа:</p>
18
19 <div class="row">
20 <div class="col-lg-5">
21
22 <?php $form = ActiveForm::begin([
23     'id' => 'login-form',
24     'fieldConfig' => [
25         'template' => "{label}\n{input}\n{error}",
26         'labelOptions' => ['class' => 'col-lg-1 col-form-label mr-lg-3'],
27         'inputOptions' => ['class' => 'col-lg-3 form-control'],
28         'errorOptions' => ['class' => 'col-lg-7 invalid-feedback'],
29     ],
30 ]); ?>
31
32 <?= $form->field($model, attribute: 'username')->textInput(['autofocus' => true]) ?>
33
34 <?= $form->field($model, attribute: 'password')->passwordInput() ?>
35
36 <?= $form->field($model, attribute: 'rememberMe')->checkbox([
37     'template' => "<div class=\"custom-control custom-checkbox\">{input} {label}</div>\n<div class=\"col-lg-8\">{error}</div>",
38 ]) ?>
39
40 <div class="form-group">
41 <div>
42 <?= Html::submitButton( content: 'Вход', ['class' => 'btn btn-primary', 'name' => 'login-button']) ?>
43 </div>
44 </div>

```

- В классе **models/LoginForm.php** добавить русификацию полей логина и пароля

```

LoginForm.php x
no usages
37 public function attributeLabels()
38 {
39     return [
40         'username' => 'Логин',
41         'password' => 'Пароль',
42         'rememberMe' => 'Запомнить',
43     ];
44 }

```

- Заменить импорты виджетов для форм регистрации и добавления новой заявки

```

request/_form.php x
1 <?php
2
3 /*use yii\helpers\Html;
4 use yii\widgets\ActiveForm;*/
5 use yii\bootstrap5\ActiveForm;
6 use yii\bootstrap5\Html;

```

```

PHP _form.php x
1  <?php
2
3  use yii\bootstrap5\ActiveForm;
4  use yii\bootstrap5\Html;

```

10. Для кастомизации отображения карточек заявок нужно добавить нужные стили (файл **web/css/main.css**)

```

CSS main.css x
1  .bg-success-card {
2      background-color: lightskyblue;
3  }
4  .bg-danger-card {
5      background-color: lightpink;
6  }

```

11. Изменить названия классов в файлах **views/request/index.php**, **views/admin/index.php**

```

PHP request/index.php x
21  <?php
22  foreach ($dataProvider->models as $model){
23
24      $classCard = match ($model->status->code) {
25          'approve' => 'text-white bg-success-card',
26          'rejected' => 'text-white bg-danger-card',
27          'new' => 'bg-light',
28      }
29

```

```

PHP admin/index.php x
22
23      $classCard = match ($model->status->code) {
24          'approve' => 'text-white bg-success-card',
25          'rejected' => 'text-white bg-danger-card',
26          'new' => 'bg-light'
27      }
28

```

Использования предоставленных медиаматериалов

Дано

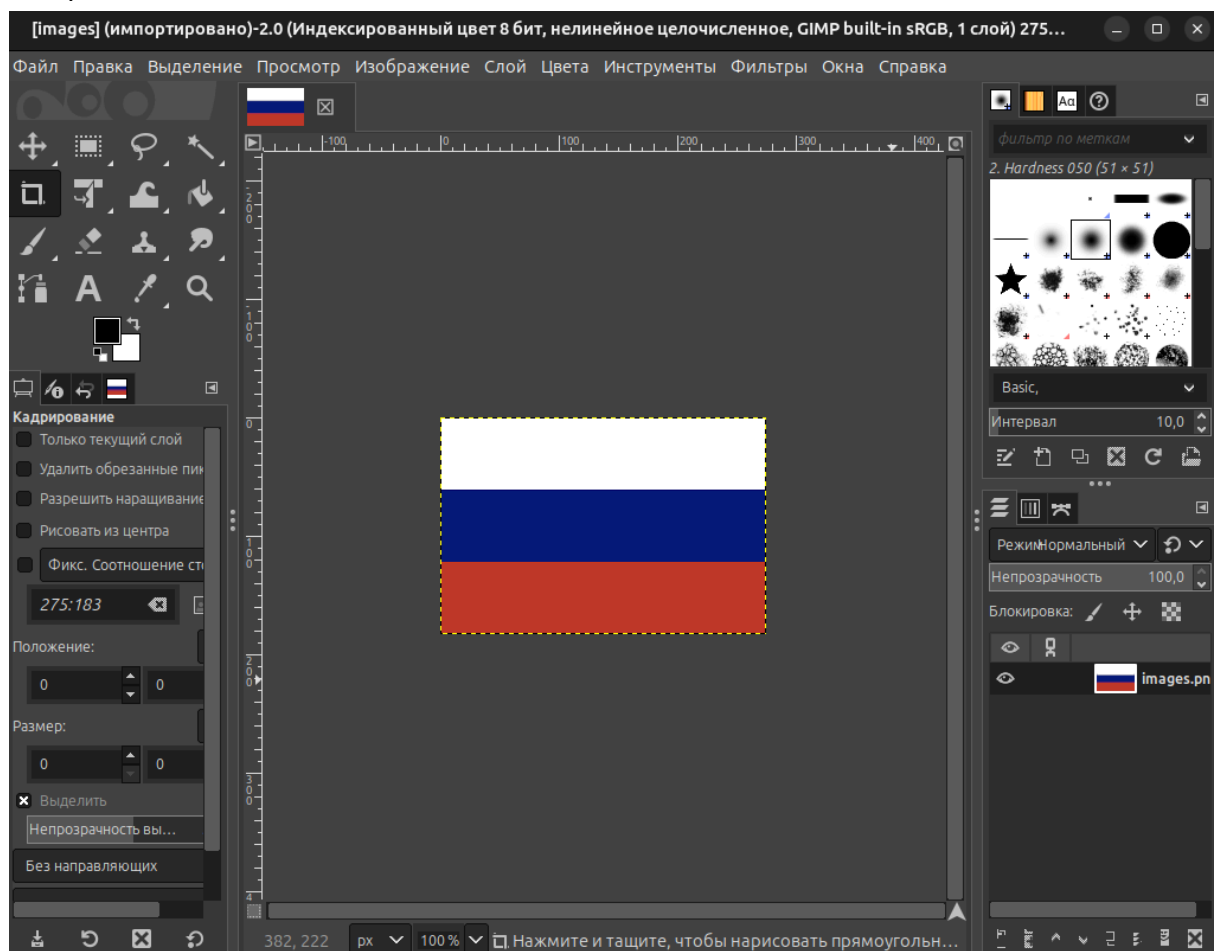
Настроенное рабочее место и фреймворк.

Надо

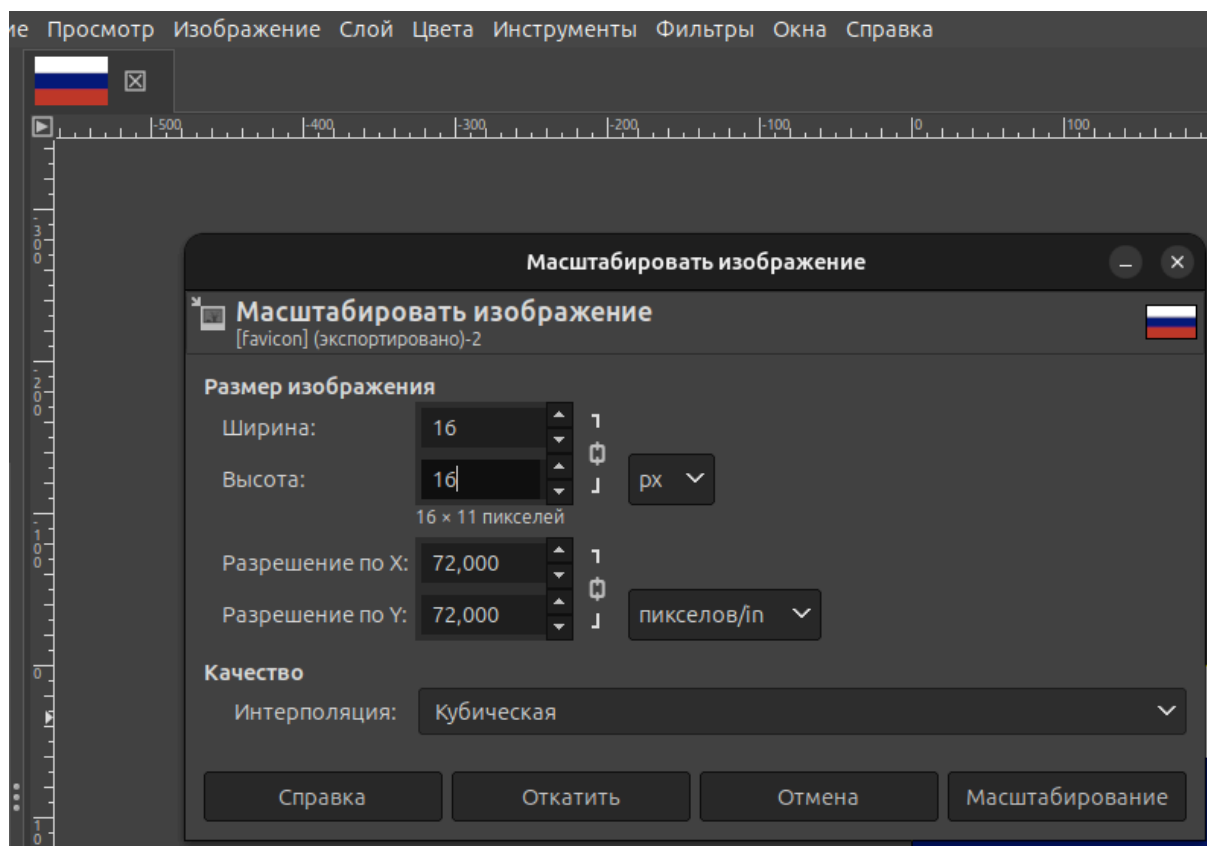
Использовать в дизайне обработанные медиа-материалы.

Алгоритм

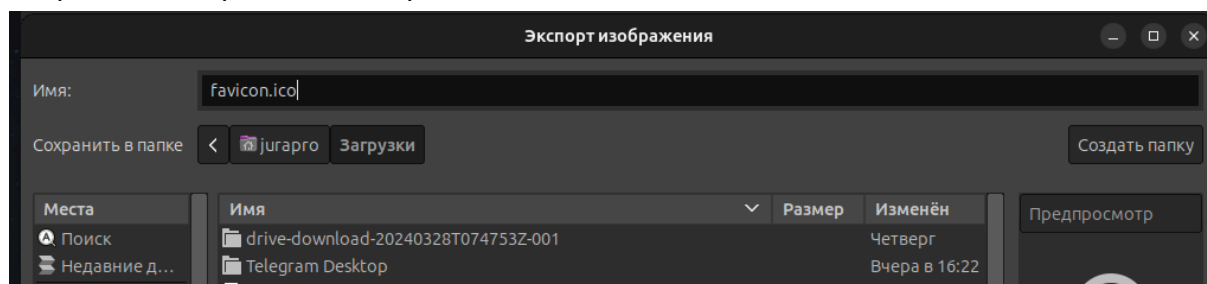
1. Открыть изображение из медиаматериалов в программе обработки изображений



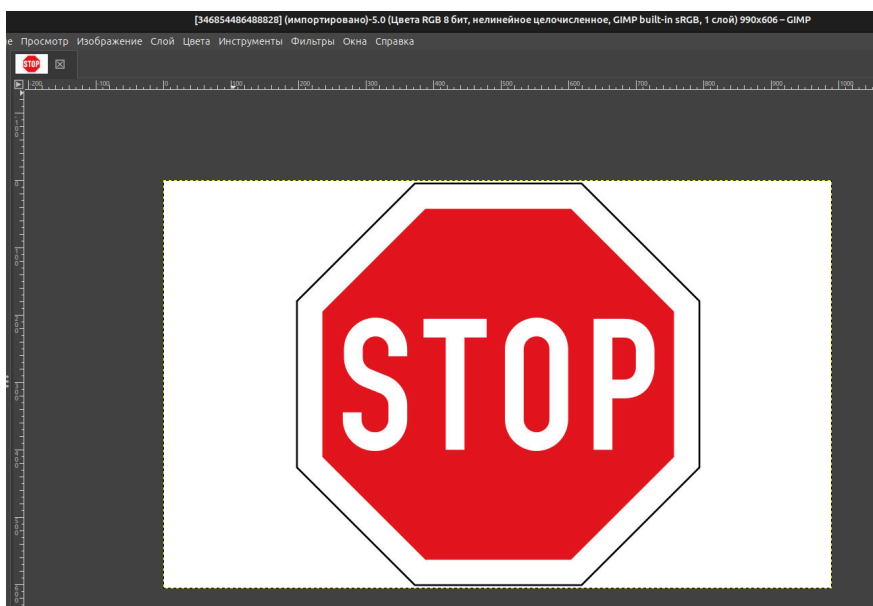
2. Изменить размер изображения до 16x16 пикселей



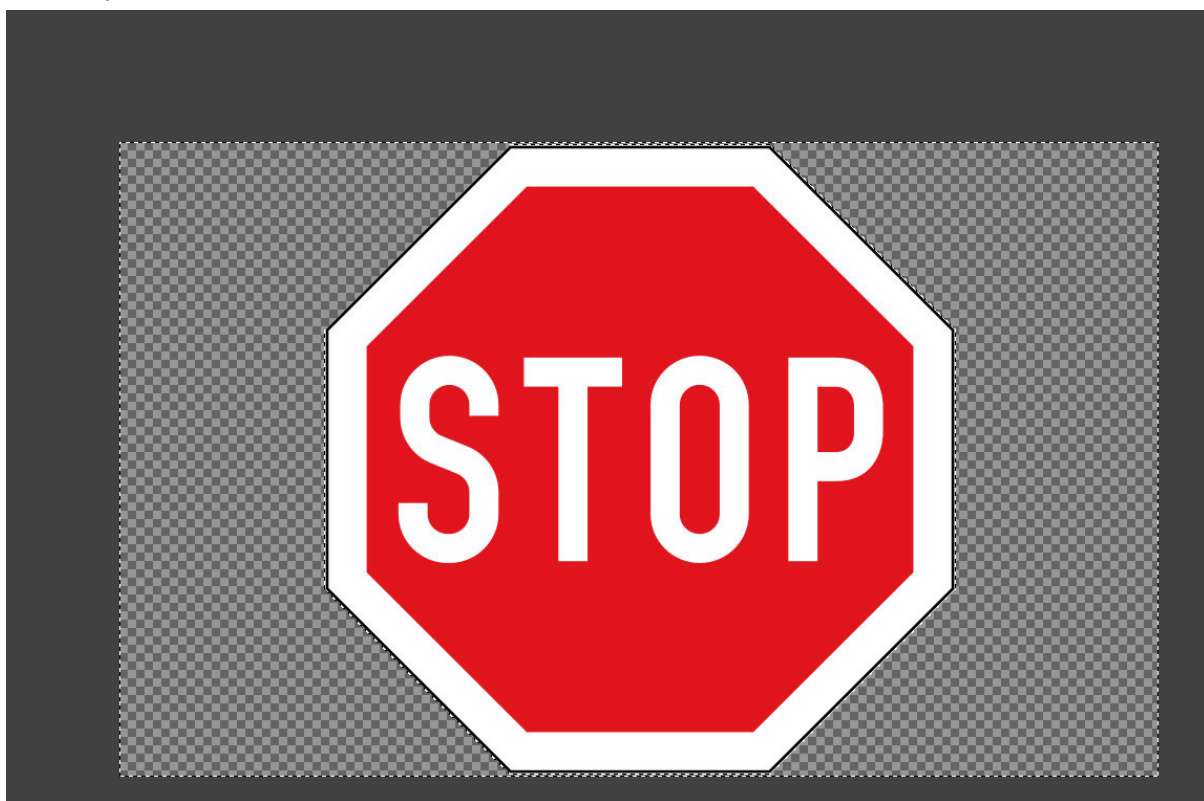
3. Сохранить изображение как фавикон



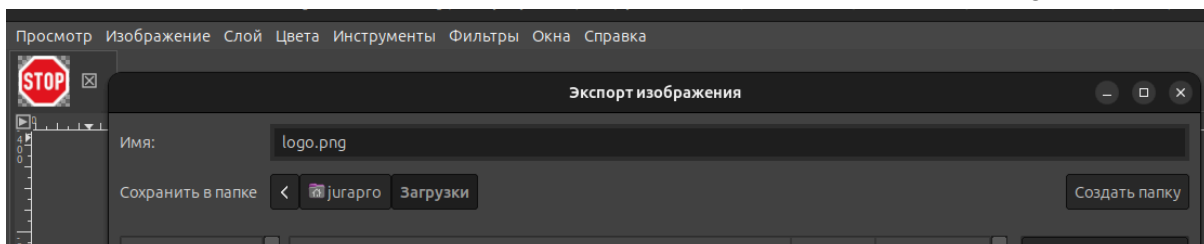
4. Заменить соответствующий файл в проекте (файл **web/favicon.ico**)
5. Открыть изображение из медиаматериалов подходящее для логотипа в программе обработки изображений



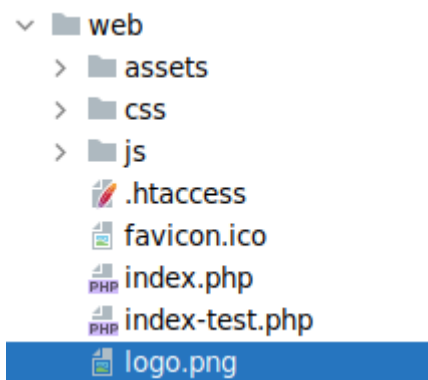
6. Нажать Слой → Прозрачность → Добавить альфа-канал, чтобы добавить канал альфа. Выделить белый фон. Затем вызвать Правка → Очистить или нажмите клавишу Del.



7. Уменьшить размер изображения до 100 пикселей и сохранить файл как png



8. Сохранить файл в директорию web



9. Добавить логотип в навигационную панель сайта (файл **views/layouts/main.php**)



Сохранение дизайна согласно заданию

Дано

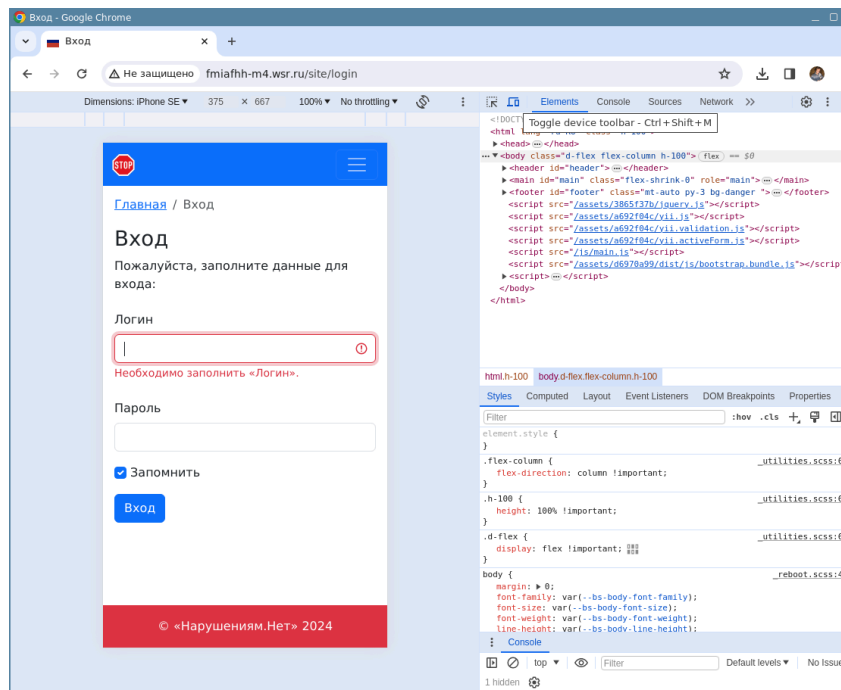
Настроенное рабочее место и фреймворк. Готовый дизайн сайта

Надо

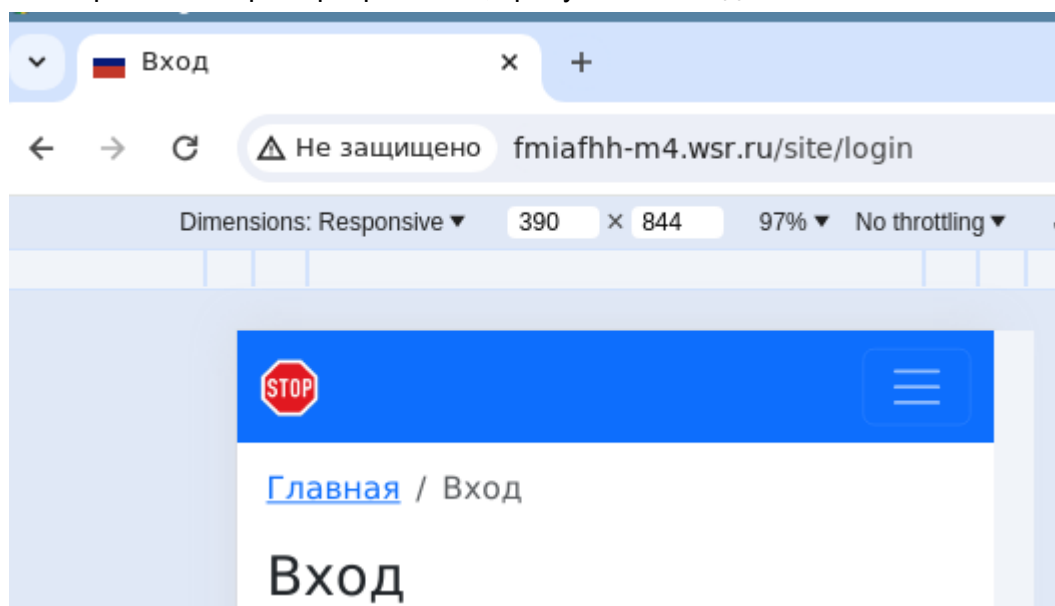
Сохранить дизайн сайта согласно заданию

Алгоритм

1. Открыть необходимую страницу в браузере
2. Открыть в браузере консоль разработчика (shift + ctl + i)



3. В настройках выбрать разрешение, требуемое по заданию



4. Используя программу для скриншота экрана, сделать снимок страницы

STOP

≡

[Главная](#) / Вход

Вход

Пожалуйста, заполните данные для входа:

Логин

ⓘ

Необходимо заполнить «Логин».



Пароль

☒ Запомнить

Вход

© «Нарушениям.Нет» 2024

5. Повторить для всех страниц



[Главная](#) / Регистрация

Регистрация

Логин

Пароль

Email

Имя



Фамилия

Отчество

Телефон

[Зарегистрироваться](#)


© «Нарушениям.Нет» 2024




[Главная](#) / Оставить новое заявление

Оставить новое заявление

Номер автомобиля

911

Описание нарушения

Описание нарушения

Подать заявку

© «Нарушениям.Нет» 2024

