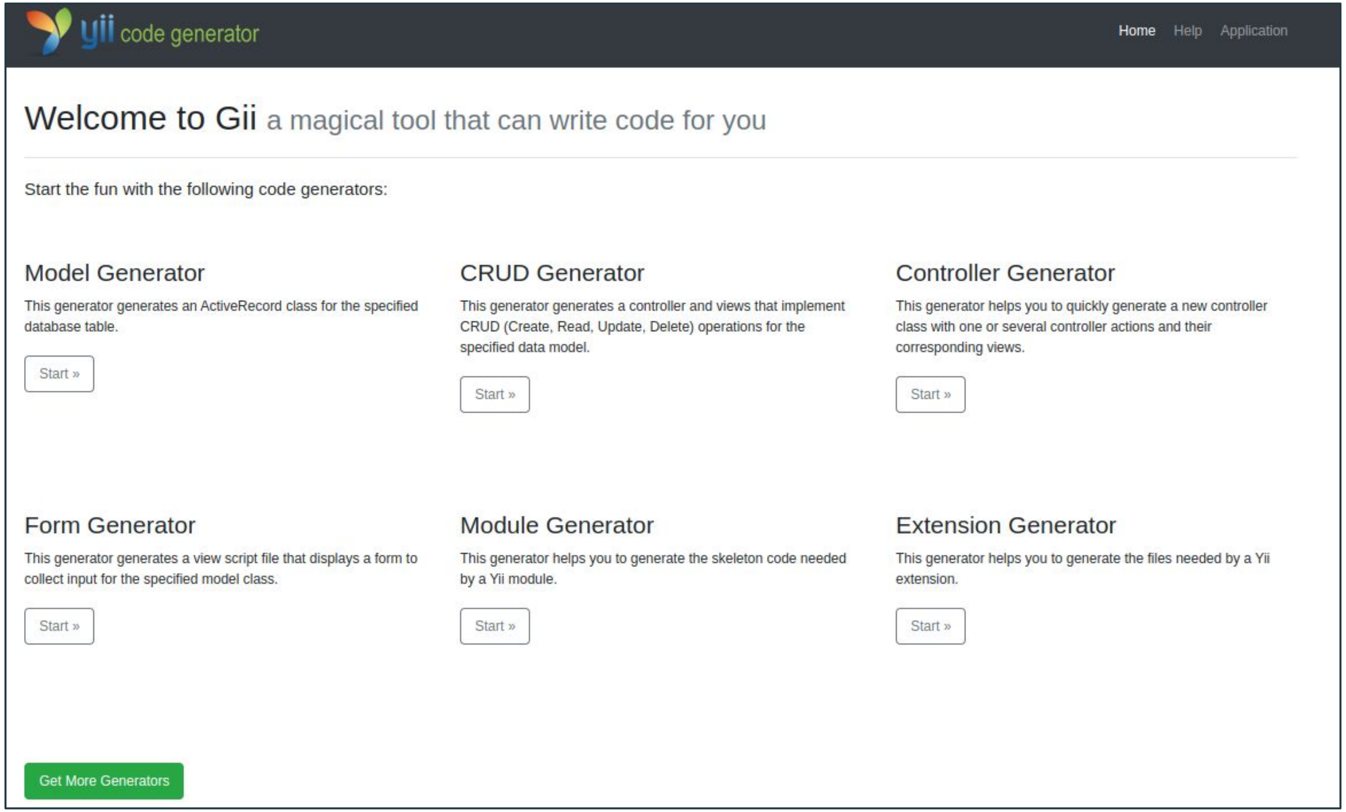
Тема 3. «Реализация регистрации и аутентификации»

Грушевский Ю.В.

Gii



Welcome to Gii a magical tool that can write code for you

Start the fun with the following code generators:

Model Generator

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

Start »

CRUD Generator

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

Start »

Controller Generator

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

Start »

Form Generator

This generator generates a view script file that displays a form to collect input for the specified model class.

Start »

Module Generator

This generator helps you to generate the skeleton code needed by a Yii module.

Start »

Extension Generator

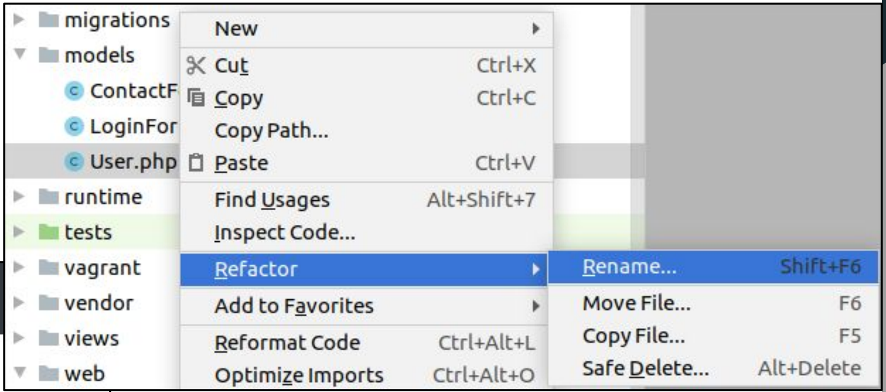
This generator helps you to generate the files needed by a Yii extension.

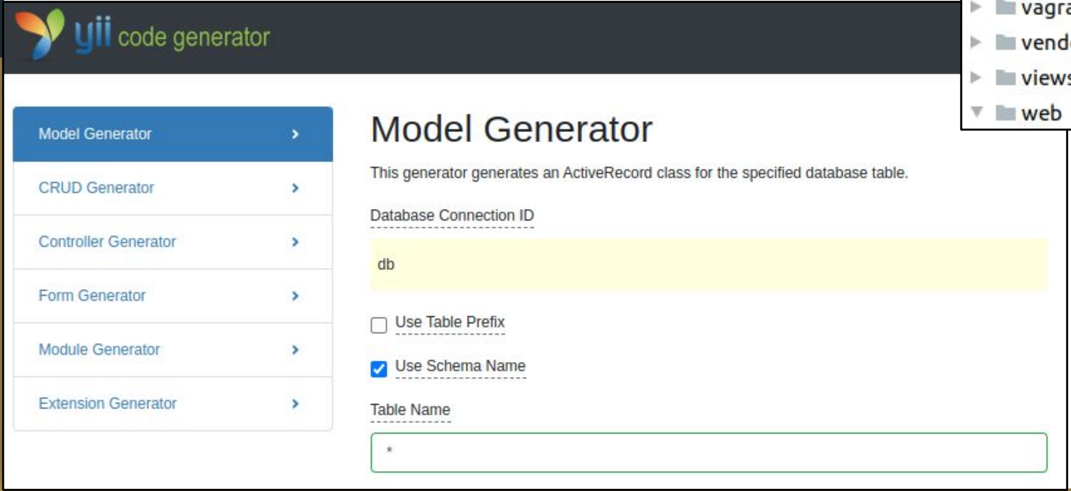
Start »

Get More Generators

Аутентификация

Генерация всех моделей





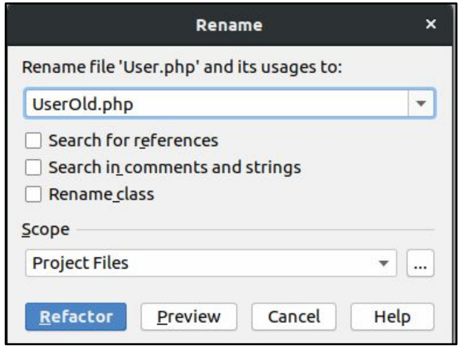
Model Generator

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

Database Connection ID: db

- [ ] Use Table Prefix
- [x] Use Schema Name

Table Name: *



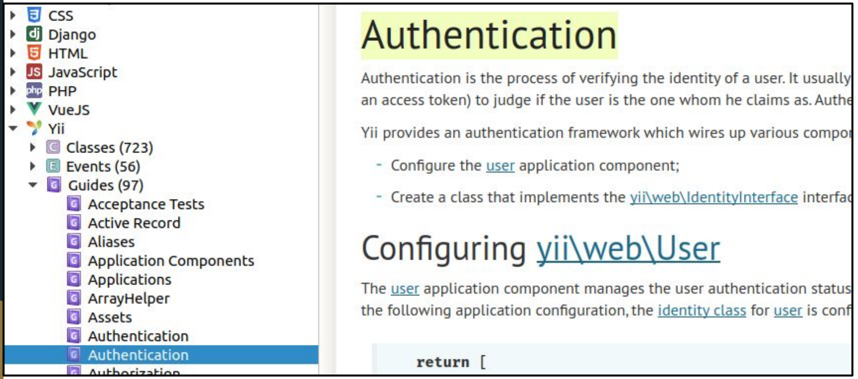
Rename file 'User.php' and its usages to:

UserOld.php

- [ ] Search for references
- [ ] Search in comments and strings
- [ ] Rename class

Scope: Project Files

Правка стандартной логики



- CSS
- Django
- HTML
- JavaScript
- PHP
- VueJS
- Yii
  - Classes (723)
  - Events (56)
  - Guides (97)
    - Acceptance Tests
    - Active Record
    - Aliases
    - Application Components
    - Applications
    - ArrayHelper
    - Assets
    - Authentication
    - Authentication
    - Authorization

Authentication

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

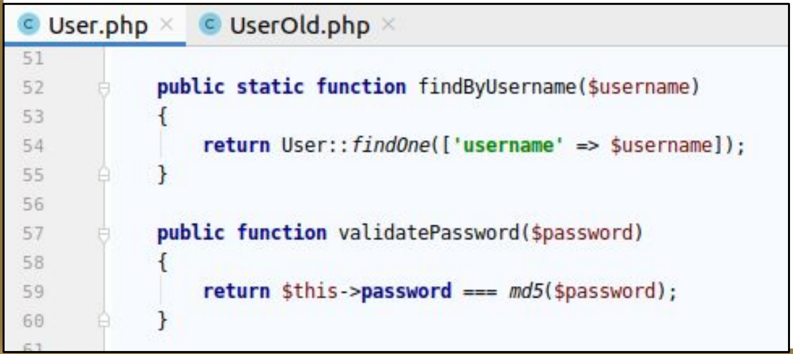
Yii provides an authentication framework which wires up various compo

- Configure the user application component;
- Create a class that implements the yii\web\IdentityInterface interfa

Configuring yii\web\User

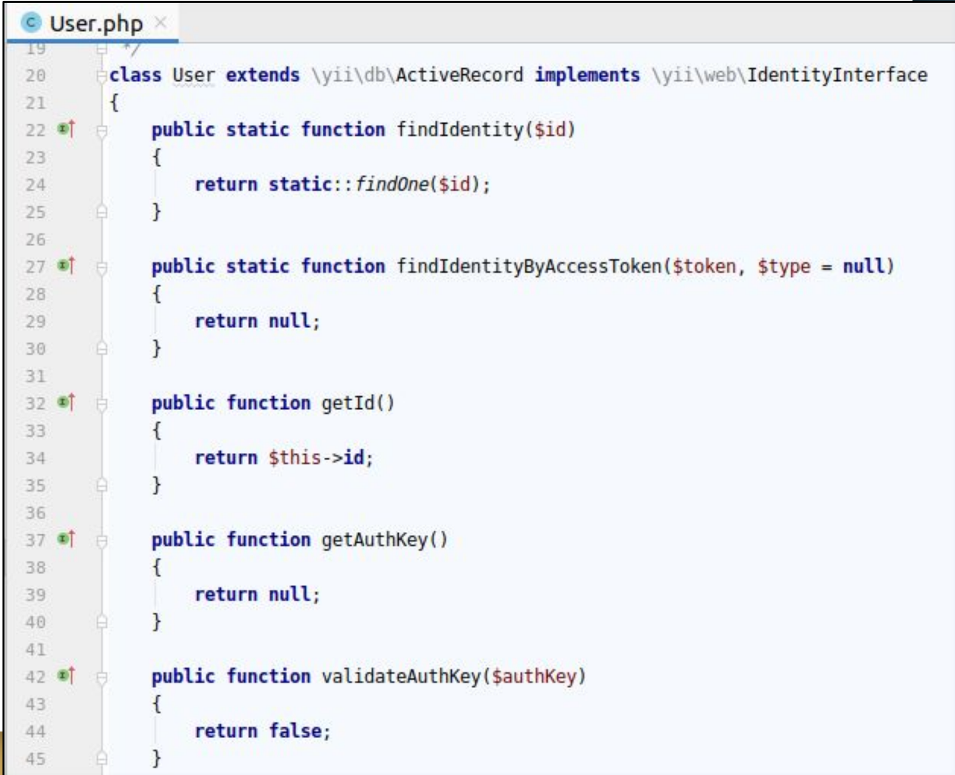
The user application component manages the user authentication status
the following application configuration, the identity class for user is conf

```
return [
```



User.php × UserOld.php ×

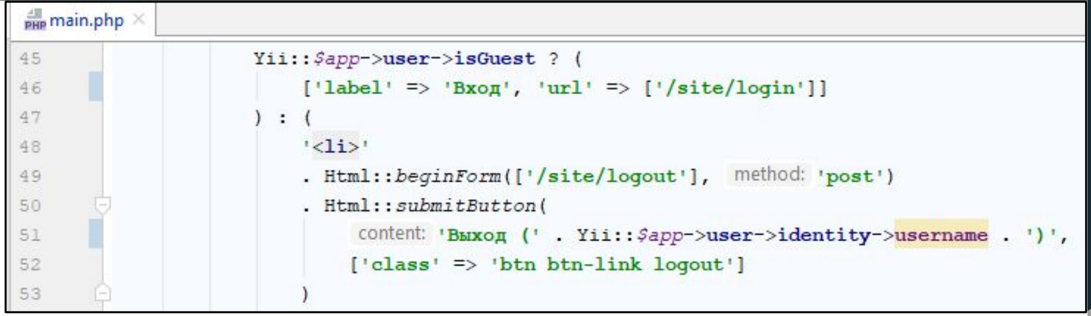
```
    public static function findByUsername($username)
    {
        return User::findOne(['username' => $username]);
    }

    public function validatePassword($password)
    {
        return $this->password === md5($password);
    }
```



User.php ×

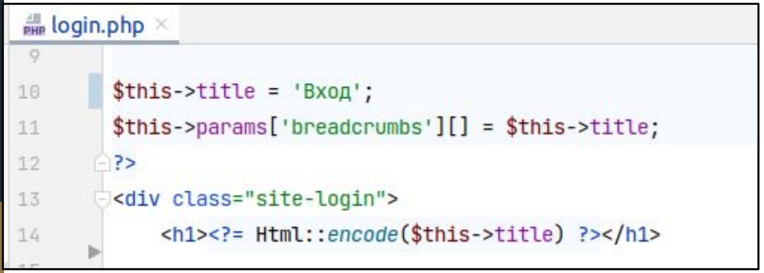
```
 */
class User extends \yii\db\ActiveRecord implements \yii\web\IdentityInterface
{
    public static function findIdentity($id)
    {
        return static::findOne($id);
    }

    public static function findIdentityByAccessToken($token, $type = null)
    {
        return null;
    }

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

    public function getAuthKey()
    {
        return null;
    }

    public function validateAuthKey($authKey)
    {
        return false;
    }
```

Русификация



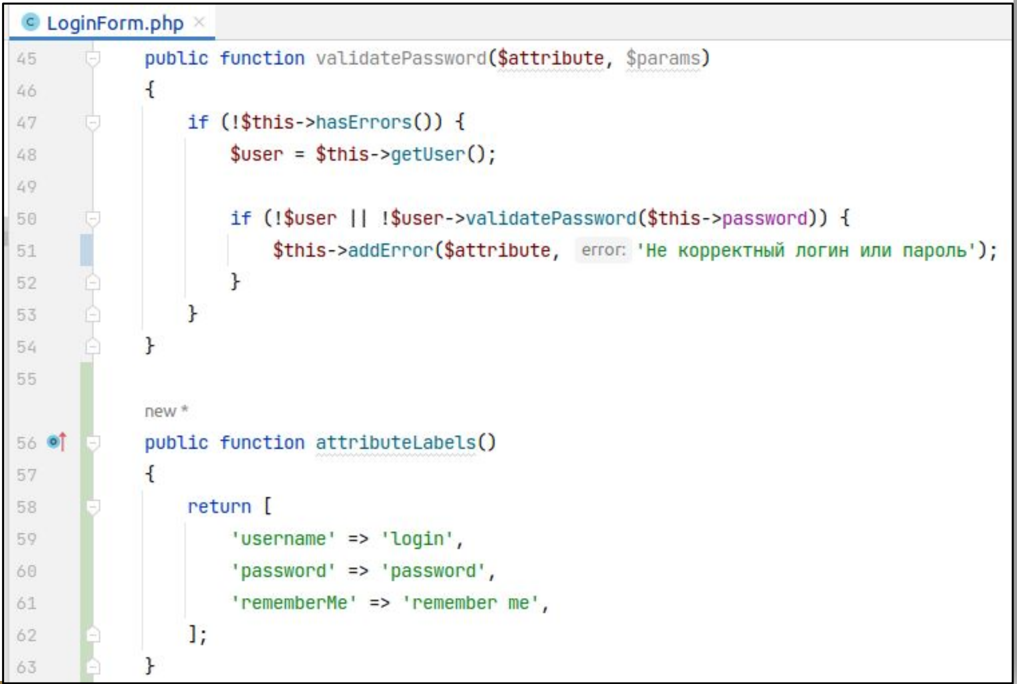
main.php

```
            Yii::$app->user->isGuest ? (
                ['label' => 'Вход', 'url' => ['/site/login']]
            ) : (
                '<li>'
                . Html::beginForm(['/site/logout'], method: 'post')
                . Html::submitButton(
                    content: 'Выход (' . Yii::$app->user->identity->username . ')',
                    ['class' => 'btn btn-link logout']
                )
```



login.php

```
$this->title = 'Вход';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="site-login">
    <h1><?= Html::encode($this->title) ?></h1>
```

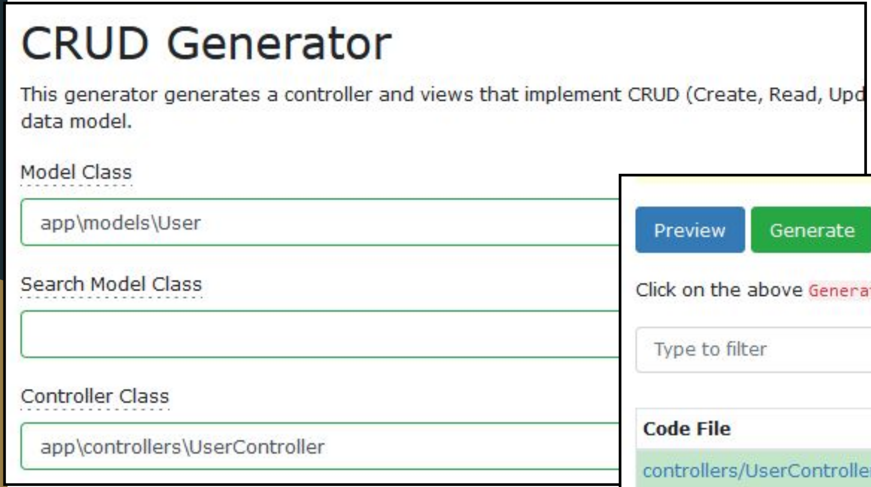


LoginForm.php

```
    public function validatePassword($attribute, $params)
    {
        if (!$this->hasErrors()) {
            $user = $this->getUser();

            if (!$user || !$user->validatePassword($this->password)) {
                $this->addError($attribute, error: 'Не корректный логин или пароль');
            }
        }
    }

    new *
    public function attributeLabels()
    {
        return [
            'username' => 'login',
            'password' => 'password',
            'rememberMe' => 'remember me',
        ];
    }
```

Базовая регистрация

GRUD модели User



CRUD Generator

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

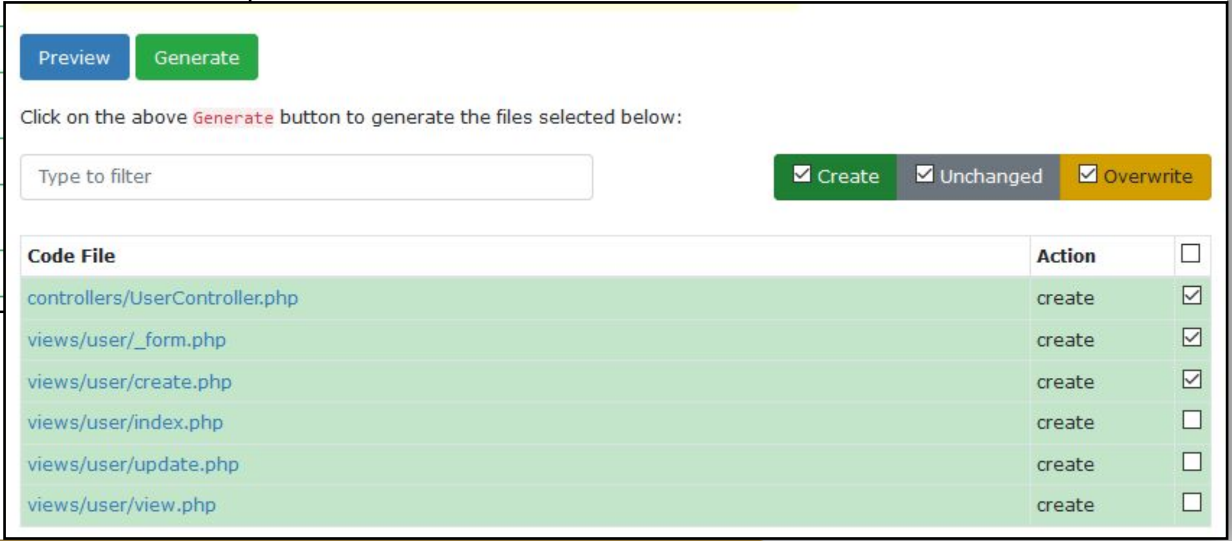
Model Class

app\models\User

Search Model Class

Controller Class

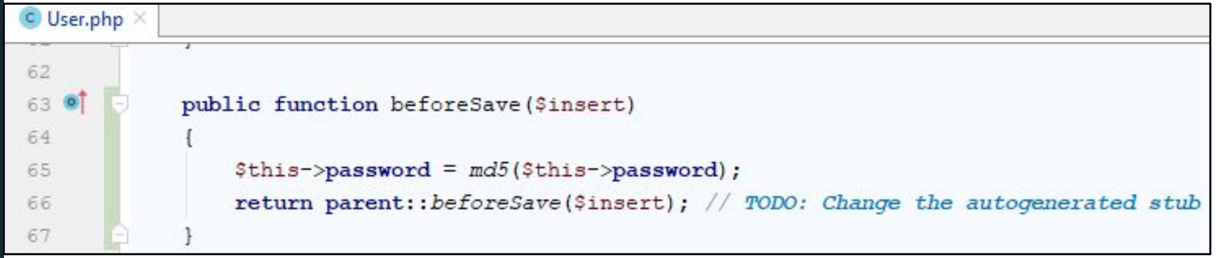
app\controllers\UserController



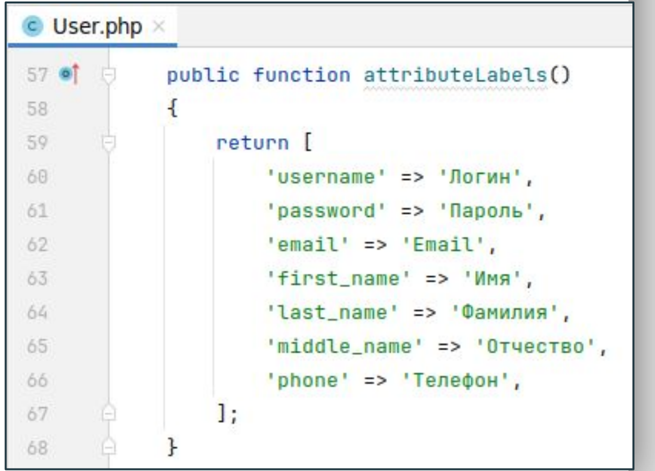
Click on the above `Generate` button to generate the files selected below:

Code File	Action	☐
controllers/UserController.php	create	☑
views/user/_form.php	create	☑
views/user/create.php	create	☑
views/user/index.php	create	☐
views/user/update.php	create	☐
views/user/view.php	create	☐

Доработка модели User

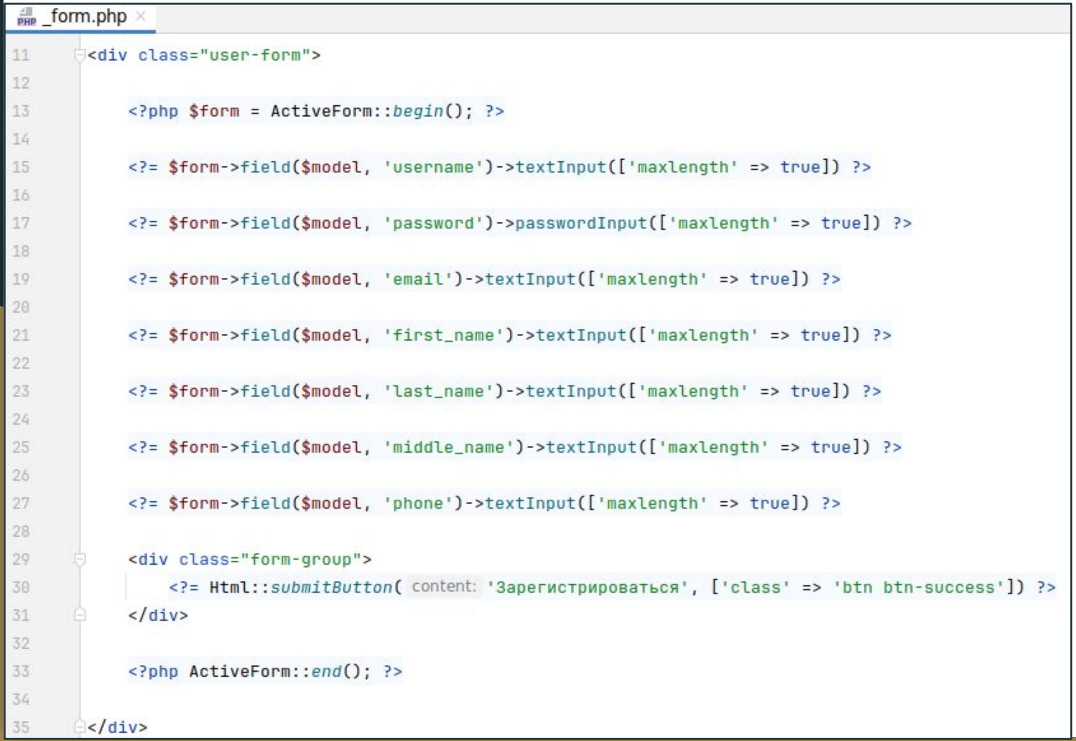

User.php

```
    public function beforeSave($insert)
    {
        $this->password = md5($this->password);
        return parent::beforeSave($insert); // TODO: Change the autogenerated stub
    }
```


User.php

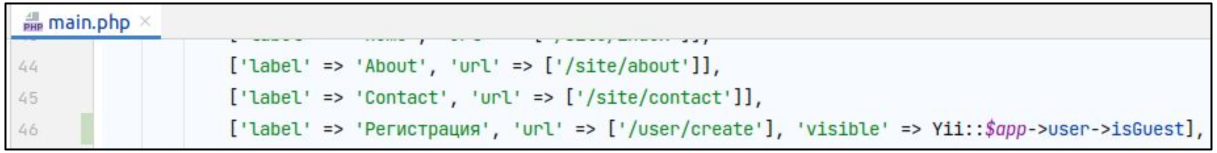
```
    public function attributeLabels()
    {
        return [
            'username' => 'Логин',
            'password' => 'Пароль',
            'email' => 'Email',
            'first_name' => 'Имя',
            'last_name' => 'Фамилия',
            'middle_name' => 'Отчество',
            'phone' => 'Телефон',
        ];
    }
```

Доработка представления


_form.php

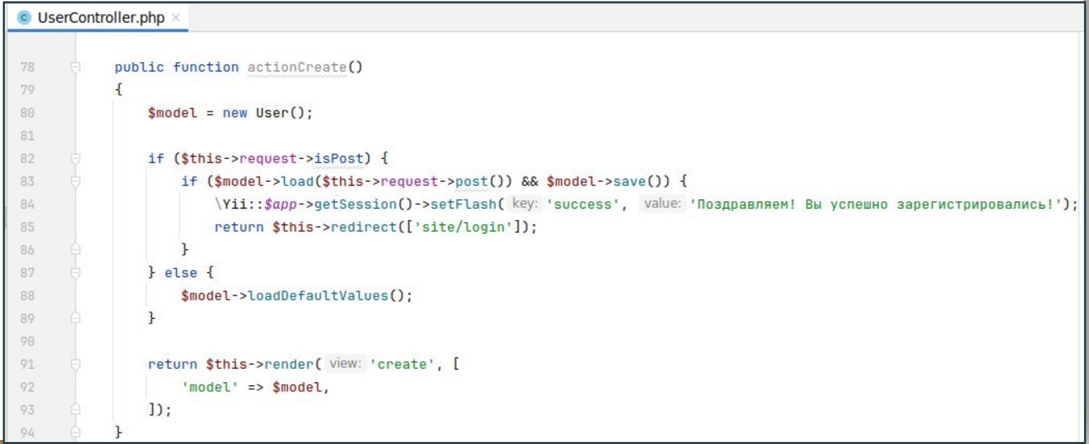
```
<div class="user-form">

    <?php $form = ActiveForm::begin(); ?>

    <?= $form->field($model, 'username')->textInput(['maxlength' => true]) ?>

    <?= $form->field($model, 'password')->passwordInput(['maxlength' => true]) ?>

    <?= $form->field($model, 'email')->textInput(['maxlength' => true]) ?>

    <?= $form->field($model, 'first_name')->textInput(['maxlength' => true]) ?>

    <?= $form->field($model, 'last_name')->textInput(['maxlength' => true]) ?>

    <?= $form->field($model, 'middle_name')->textInput(['maxlength' => true]) ?>

    <?= $form->field($model, 'phone')->textInput(['maxlength' => true]) ?>

    <div class="form-group">
        <?= Html::submitButton( content: 'Зарегистрироваться', ['class' => 'btn btn-success']) ?>
    </div>

    <?php ActiveForm::end(); ?>

</div>
```

Доработка контроллера и добавление ссылки



main.php

```
            ['label' => 'About', 'url' => ['/site/about']],
            ['label' => 'Contact', 'url' => ['/site/contact']],
            ['label' => 'Регистрация', 'url' => ['/user/create'], 'visible' => Yii::$app->user->isGuest],
```

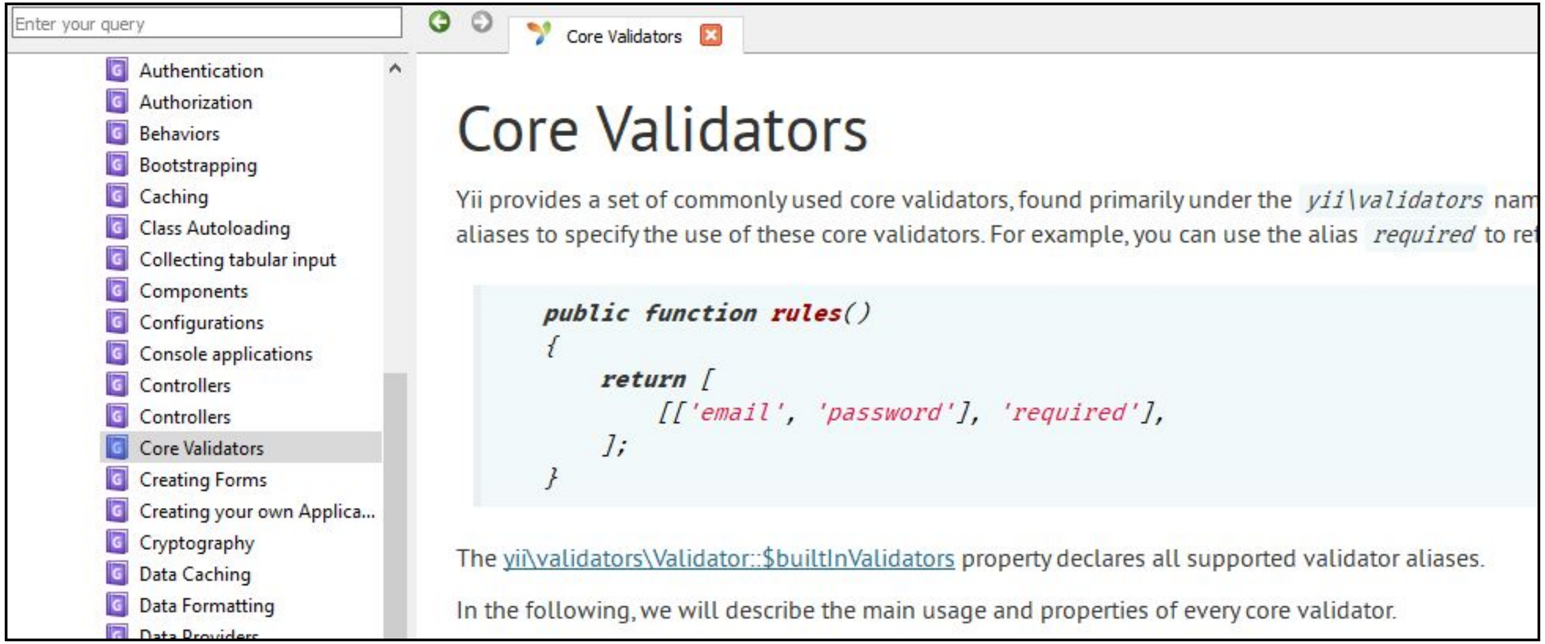


UserController.php

```
    public function actionCreate()
    {
        $model = new User();

        if ($this->request->isPost) {
            if ($model->load($this->request->post()) && $model->save()) {
                \Yii::$app->getSession()->setFlash( key: 'success', value: 'Поздравляем! Вы успешно зарегистрировались!');
                return $this->redirect(['site/login']);
            }
        } else {
            $model->loadDefaultValues();
        }

        return $this->render( view: 'create', [
            'model' => $model,
        ]);
    }
```

Реализация требуемых валидаций

Вспомогательная статья



Core Validators

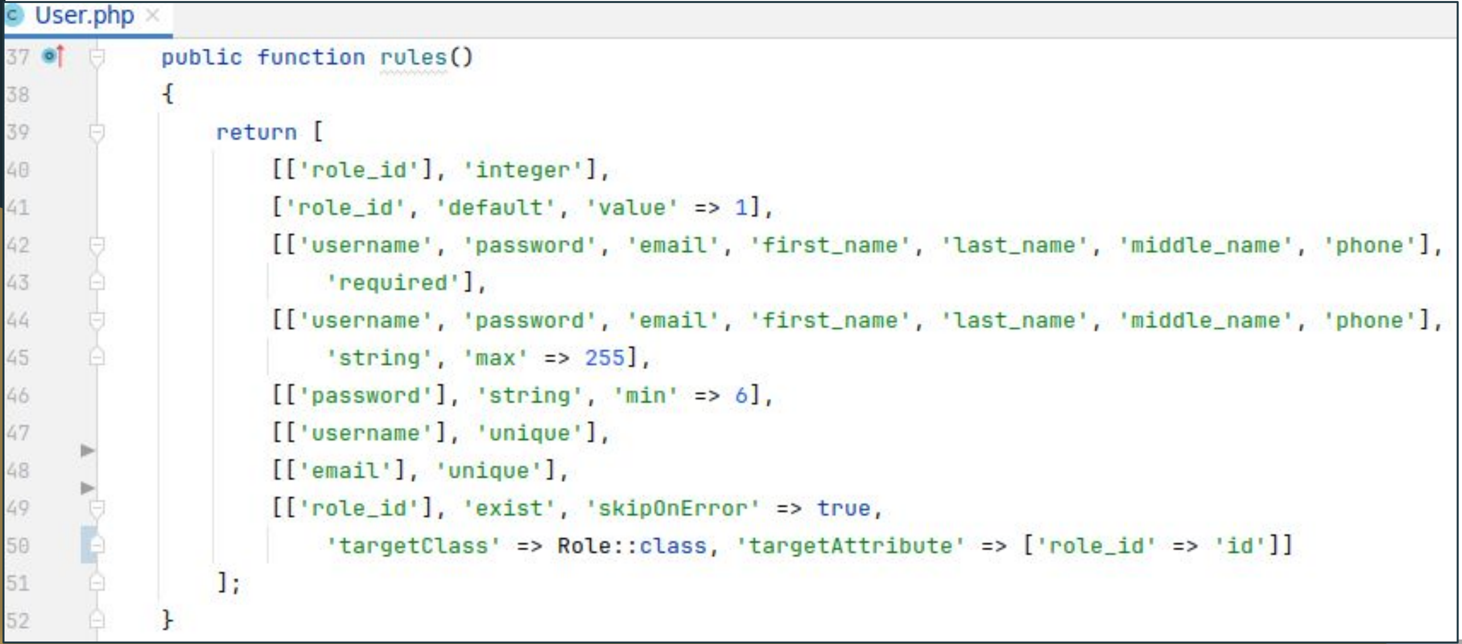
Yii provides a set of commonly used core validators, found primarily under the *yii\validators* nam
aliases to specify the use of these core validators. For example, you can use the alias *required* to ret

```
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.

Ограничения в модели


User.php

```
    public function rules()
    {
        return [
            [['role_id'], 'integer'],
            ['role_id', 'default', 'value' => 1],
            [['username', 'password', 'email', 'first_name', 'last_name', 'middle_name', 'phone'],
                'required'],
            [['username', 'password', 'email', 'first_name', 'last_name', 'middle_name', 'phone'],
                'string', 'max' => 255],
            [['password'], 'string', 'min' => 6],
            [['username'], 'unique'],
            [['email'], 'unique'],
            [['role_id'], 'exist', 'skipOnError' => true,
                'targetClass' => Role::class, 'targetAttribute' => ['role_id' => 'id']]
        ];
    }
```

Ограничения

- 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.

Задание (90 мин)

Пользовательские валидации

Доработайте модель User добавив ограничения требуемые по заданию