Лабораторна робота №5

**Тема:** Створення соціальної мережі на Yii2

**Мета:** Створити соціальну мережу з допомогою Yii2

Хід роботи

До створення соціальноої мережі на Yii2 я підійшов по принципу ‘database first’, тобто перш за все сформував базу даних.

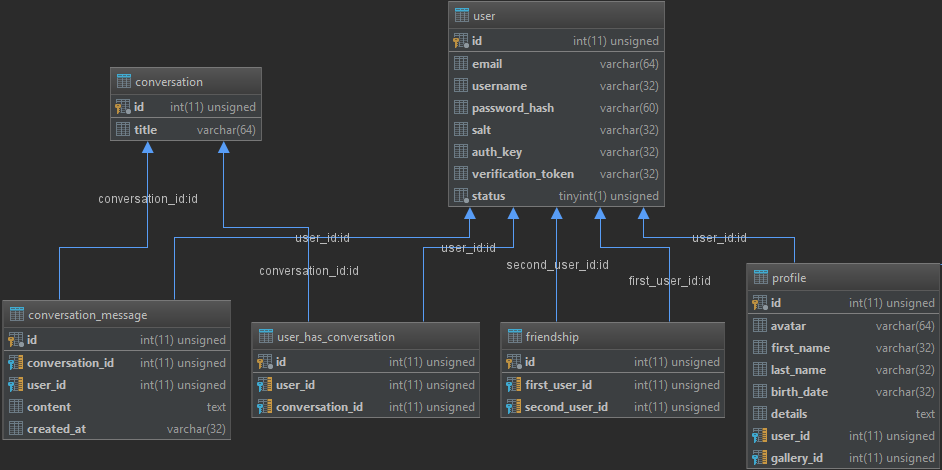


Рисунок 1 - Схема БД

Створив базу за допомогою мігарцій:

Міграції:

create\_user\_table.php

**<?php  
  
use** yii\db\Migration;  
  
*/\*\*  
 \* Handles the creation of table `user`.  
 \*/***class** m181107\_171852\_create\_user\_table **extends** Migration  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeUp()  
 {  
 $this->createTable('user', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'email' => $this->string(64),  
 'username' => $this->string(32),  
 'password\_hash' => $this->string(60),  
 'salt' => $this->string(32),  
 'auth\_key' => $this->string(32),  
 'verification\_token' => $this->string(32),  
 'status' => $this->tinyInteger(1)->unsigned()->notNull()->defaultValue(2),  
 ]);  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeDown()  
 {  
 $this->dropTable('user');  
 }  
}

create\_chat\_tables.php

**<?php  
  
use** yii\db\Migration;  
  
*/\*\*  
 \* Class m181108\_174737\_create\_chat\_tables  
 \*/***class** m181108\_174737\_create\_chat\_tables **extends** Migration  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeUp()  
 {  
 $this->createTable('conversation', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'title' => $this->string(64),  
 ]);  
  
 $this->createTable('user\_has\_conversation', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'user\_id' => $this->integer(11)->unsigned(),  
 'conversation\_id' => $this->integer(11)->unsigned()  
 ]);  
  
 $this->createIndex(  
 'idx-user\_has\_conversation-user\_id',  
 'user\_has\_conversation',  
 'user\_id'  
 );  
  
 $this->createIndex(  
 'idx-user\_has\_conversation-conversation\_id',  
 'user\_has\_conversation',  
 'conversation\_id'  
 );  
  
 $this->createTable('conversation\_message', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'conversation\_id' => $this->integer(11)->unsigned(),  
 'user\_id' => $this->integer(11)->unsigned(),  
 'content' => $this->text(),  
 'created\_at' => $this->string(32)  
 ]);  
  
 $this->createIndex(  
 'idx-conversation\_message-conversation\_id',  
 'conversation\_message',  
 'conversation\_id'  
 );  
  
 $this->createIndex(  
 'idx-conversation\_message-user\_id',  
 'conversation\_message',  
 'user\_id'  
 );  
  
  
 $this->addForeignKey(  
 'fk-conversation\_message-user\_id',  
 'conversation\_message',  
 'user\_id',  
 'user',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
  
 $this->addForeignKey(  
 'fk-conversation\_message-conversation\_id',  
 'conversation\_message',  
 'conversation\_id',  
 'conversation',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
  
 $this->addForeignKey(  
 'fk-user\_has\_conversation-user\_id',  
 'user\_has\_conversation',  
 'user\_id',  
 'user',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
  
 $this->addForeignKey(  
 'fk-user\_has\_conversation-conversation\_id',  
 'user\_has\_conversation',  
 'conversation\_id',  
 'conversation',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeDown()  
 {  
 $this->dropForeignKey('fk-conversation\_message-conversation\_id', 'conversation\_message');  
 $this->dropIndex('idx-conversation\_message-conversation\_id', 'conversation\_message');  
  
 $this->dropForeignKey('fk-conversation\_message-user\_id', 'conversation\_message');  
 $this->dropIndex('idx-conversation\_message-user\_id', 'conversation\_message');  
  
 $this->dropForeignKey('fk-user\_has\_conversation-conversation\_id', 'user\_has\_conversation');  
 $this->dropIndex('idx-user\_has\_conversation-conversation\_id', 'user\_has\_conversation');  
  
 $this->dropForeignKey('fk-user\_has\_conversation-user\_id', 'user\_has\_conversation');  
 $this->dropIndex('idx-user\_has\_conversation-user\_id', 'user\_has\_conversation');  
  
 $this->dropTable('conversation\_message');  
 $this->dropTable('user\_has\_conversation');  
 $this->dropTable('conversation');  
 }  
}

create\_friendship\_table.php

**<?php  
  
use** yii\db\Migration;  
  
*/\*\*  
 \* Handles the creation of table `friendship`.  
 \* Has foreign keys to the tables:  
 \*  
 \* - `user`  
 \* - `user`  
 \*/***class** m181119\_084318\_create\_friendship\_table **extends** Migration  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeUp()  
 {  
 $this->createTable('friendship', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'first\_user\_id' => $this->integer(11)->unsigned(),  
 'second\_user\_id' => $this->integer(11)->unsigned(),  
 ]);  
  
 $this->createIndex(  
 'idx-friendship-users\_id',  
 'friendship',  
 [  
 'first\_user\_id',  
 'second\_user\_id'  
 ],  
 **true** );  
  
 // add foreign key for table `user`  
 $this->addForeignKey(  
 'fk-friendship-first\_user\_id',  
 'friendship',  
 'first\_user\_id',  
 'user',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
  
 // add foreign key for table `user`  
 $this->addForeignKey(  
 'fk-friendship-second\_user\_id',  
 'friendship',  
 'second\_user\_id',  
 'user',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeDown()  
 {  
 // drops foreign key for table `user`  
 $this->dropForeignKey(  
 'fk-friendship-first\_user\_id',  
 'friendship'  
 );  
  
 // drops foreign key for table `user`  
 $this->dropForeignKey(  
 'fk-friendship-second\_user\_id',  
 'friendship'  
 );  
  
 $this->dropIndex(  
 'idx-friendship-users\_id',  
 'friendship'  
 );  
  
 $this->dropTable('friendship');  
 }  
}

create\_profile\_table.php

**<?php  
  
use** yii\db\Migration;  
  
*/\*\*  
 \* Handles the creation of table `profile`.  
 \* Has foreign keys to the tables:  
 \*  
 \* - `user`  
 \* - `gallery`  
 \*/***class** m181119\_085007\_create\_profile\_table **extends** Migration  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeUp()  
 {  
 $this->createTable('profile', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'avatar' => $this->string(64),  
 'first\_name' => $this->string(32),  
 'last\_name' => $this->string(32),  
 'birth\_date' => $this->string(32),  
 'details' => $this->text(),  
 'user\_id' => $this->integer(11)->unsigned(),  
 'gallery\_id' => $this->integer(11)->unsigned(),  
 ]);  
  
 // creates index for column `user\_id`  
 $this->createIndex(  
 'idx-profile-user\_id',  
 'profile',  
 'user\_id'  
 );  
  
 // add foreign key for table `user`  
 $this->addForeignKey(  
 'fk-profile-user\_id',  
 'profile',  
 'user\_id',  
 'user',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
  
 // creates index for column `gallery\_id`  
 $this->createIndex(  
 'idx-profile-gallery\_id',  
 'profile',  
 'gallery\_id'  
 );  
  
 // add foreign key for table `gallery`  
 $this->addForeignKey(  
 'fk-profile-gallery\_id',  
 'profile',  
 'gallery\_id',  
 'gallery',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeDown()  
 {  
 // drops foreign key for table `user`  
 $this->dropForeignKey(  
 'fk-profile-user\_id',  
 'profile'  
 );  
  
 // drops index for column `user\_id`  
 $this->dropIndex(  
 'idx-profile-user\_id',  
 'profile'  
 );  
  
 // drops foreign key for table `gallery`  
 $this->dropForeignKey(  
 'fk-profile-gallery\_id',  
 'profile'  
 );  
  
 // drops index for column `gallery\_id`  
 $this->dropIndex(  
 'idx-profile-gallery\_id',  
 'profile'  
 );  
  
 $this->dropTable('profile');  
 }  
}

Після створення таблиць і встановлення звязків можна згенерувати моделі в Gii Code Generator:

- в адресній строці після index.php дописуємо рендер ‘ ?r=gii ’;

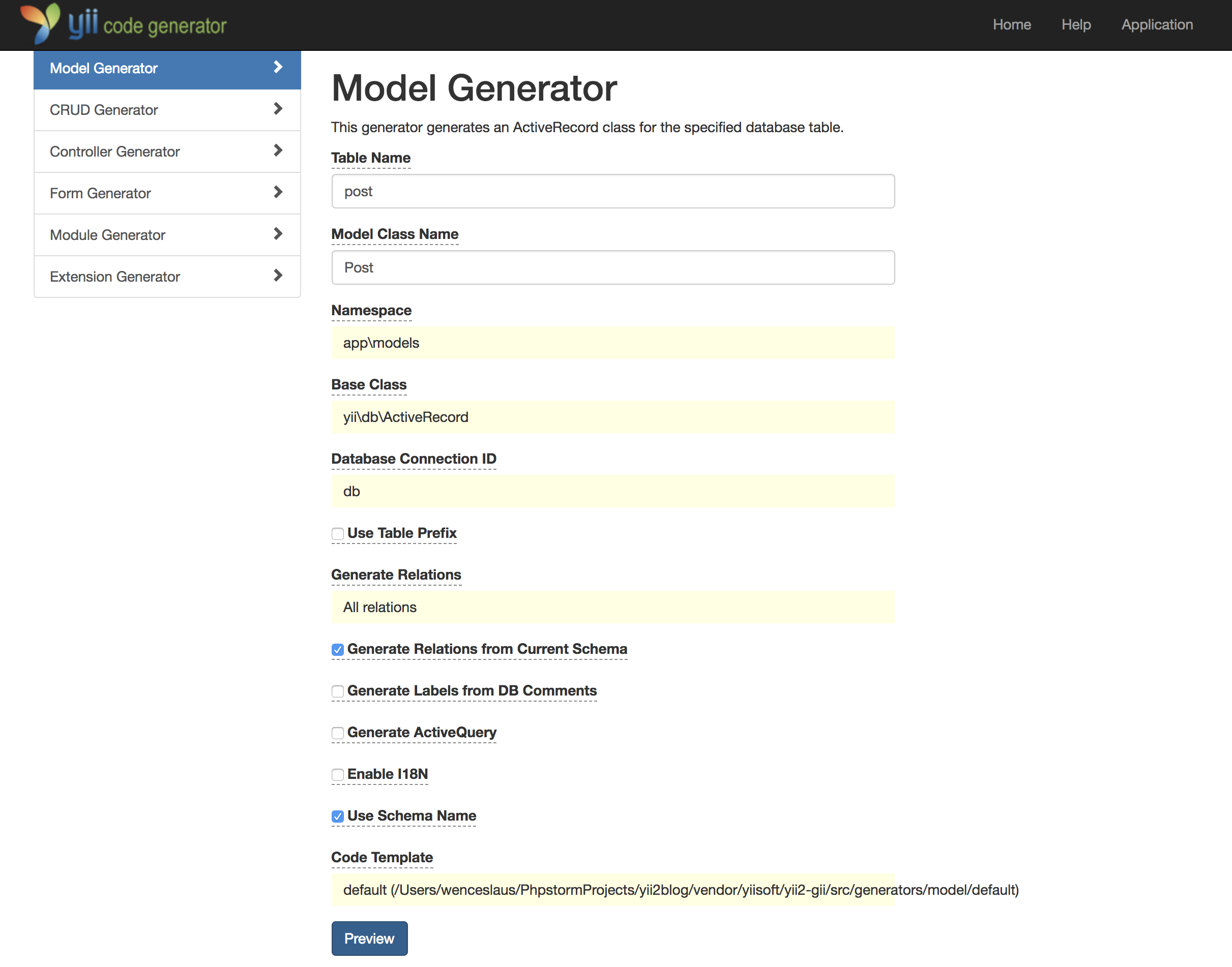


Рисунок 2. Генерація моделі

На заголовки типу Table Name, Model Class Name і т.д. можна навестись для отримання інструкцій і слідуючи їм згенеруємо моделі для всіх таблиць.

Далі ми генеруємо CRUD (абр. Create Read Update Delete), CRUD це інтерфейс для зручного керування даними. Разом з ним згенеруються і контролери для моделей, отже окремо їх генерувати не потрібно.

Як правило CRUD будуть користуватись редактори чи адміністратори.

Генерувати CRUD нам потрібно тільки для моделей: **Conversation**,, **User**.

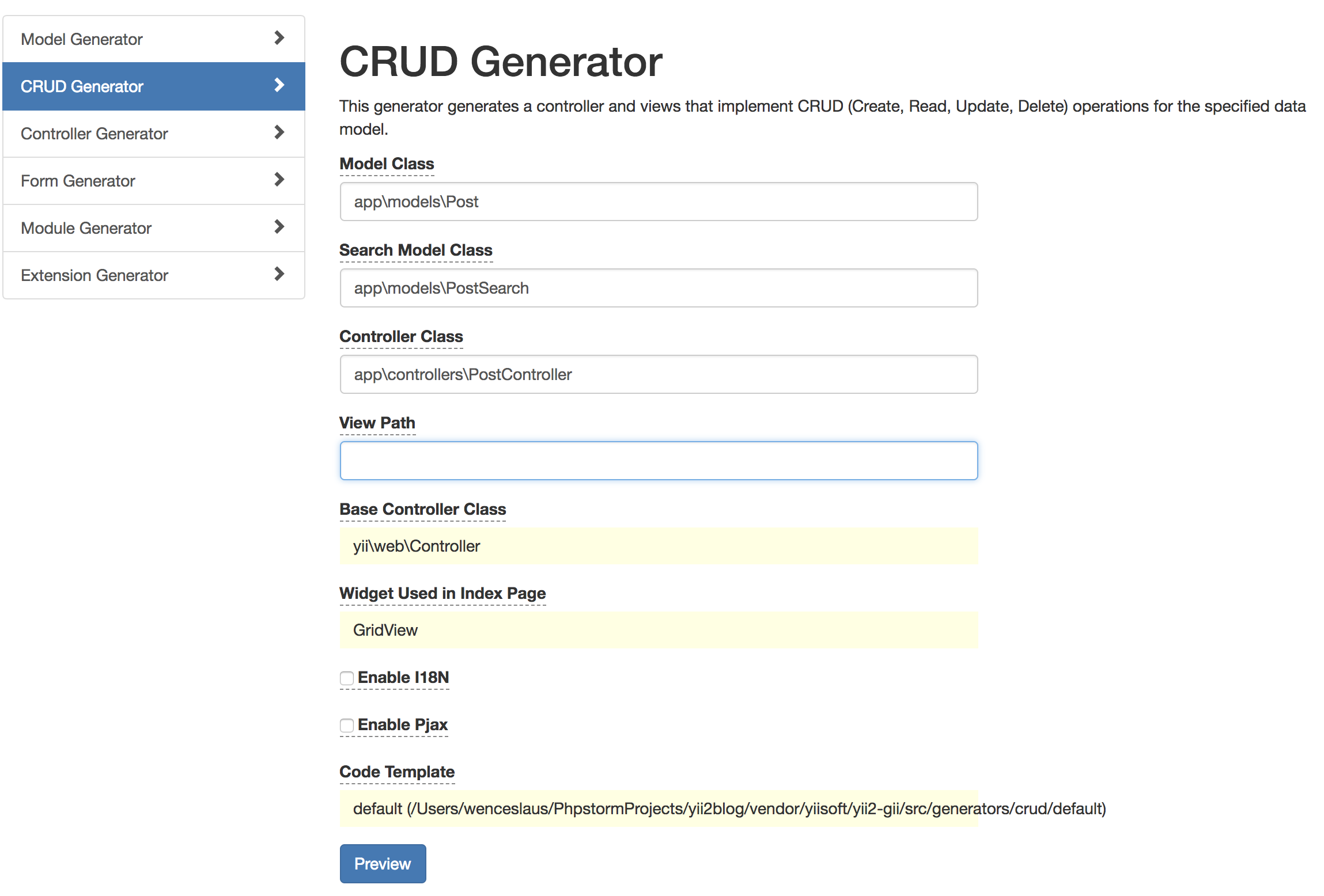


Рисунок 3. Генерація CRUD

Тепер можна рефакторити згенеровану основу блога. Я буду розглядати код по елементам блогу.

# Користувач.

Рефакторимо модель User, добавляємо всі потрібні нам методи для автентифікації та ін.

models/User.php

**<?php  
  
namespace** app\models;  
  
**use** Yii;  
**use** yii\base\Model;  
**use** yii\base\NotSupportedException;  
**use** yii\db\ActiveRecord;  
**use** yii\helpers\ArrayHelper;  
**use** yii\web\IdentityInterface;  
**use** yii\web\UploadedFile;  
  
*/\*\*  
 \* This is the model class for table "user".  
 \*  
 \** ***@property*** *string $id  
 \** ***@property*** *string $email  
 \** ***@property*** *string $username  
 \** ***@property*** *string $password\_hash  
 \** ***@property*** *string $salt  
 \** ***@property*** *string $auth\_key  
 \** ***@property*** *string $access\_token  
 \** ***@property*** *string $verification\_token  
 \** ***@property*** *int $status  
 \*  
 \** ***@property*** *Profile $profile  
 \** ***@property*** *User[] $followers  
 \** ***@property*** *Friendship[] $friendships  
 \** ***@property*** *Conversation[] $conversations  
 \** ***@property*** *ConversationMessage[] $conversationMessages  
 \** ***@property*** *UserHasConversation[] $userHasConversations  
 \*/***class** User **extends** ActiveRecord **implements** IdentityInterface  
{  
 **const** *STATUS\_BLOCKED* = 0;  
 **const** *STATUS\_ACTIVE* = 1;  
 **const** *STATUS\_WAIT* = 2;  
  
 **public** $role;  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public static function** tableName()  
 {  
 **return** 'user';  
 }  
  
 */\*\*  
 \** ***@param*** *int|string $id  
 \** ***@return*** *User|null|IdentityInterface  
 \*/* **public static function** findIdentity($id)  
 {  
 **return static**::*findOne*($id);  
 }  
  
 */\*\*  
 \** ***@param*** *mixed $token  
 \** ***@param*** *null $type  
 \** ***@return*** *void|IdentityInterface  
 \** ***@throws*** *NotSupportedException  
 \*/* **public static function** findIdentityByAccessToken($token, $type = **null**)  
 {  
 **throw new** NotSupportedException('findIdentityByAccessToken is not implemented.');  
 }  
  
 */\*\*  
 \** ***@param*** *$verification\_token  
 \** ***@return*** *User|null  
 \*/* **public static function** findIdentityByVerificationToken($verification\_token)  
 {  
 **return static**::*findOne*(['verification\_token' => $verification\_token]);  
 }  
  
 */\*\*  
 \** ***@param*** *$username  
 \** ***@return*** *User|null  
 \*/* **public static function** findByUsername($username)  
 {  
 **return static**::*findOne*(['username' => $username]);  
 }  
  
 **public function** afterFind()  
 {  
 $roles = $this->getRoles();  
 $role = array\_shift($roles);  
 $this->role = $role->name;  
 **parent**::*afterFind*();  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\rbac\Role[]  
 \*/* **public function** getRoles()  
 {  
 **return** Yii::*$app*->authManager->getRolesByUser($this->id);  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** rules()  
 {  
 **return** [  
 [['email'], 'string', 'max' => 64],  
 [['username', 'salt', 'auth\_key', 'verification\_token'], 'string', 'max' => 32],  
 [['password\_hash'], 'string', 'max' => 60],  
 [['status'], 'in', 'range' => array\_keys(**self**::*getStatuses*())],  
 [['role'], 'string'],  
 ];  
 }  
  
 */\*\*  
 \* Get statuses array  
 \*  
 \** ***@return*** *array  
 \*/* **public function** getStatuses()  
 {  
 **return** [  
 **self**::*STATUS\_BLOCKED* => 'blocked',  
 **self**::*STATUS\_ACTIVE* => 'active',  
 **self**::*STATUS\_WAIT* => 'waiting for confirm',  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** attributeLabels()  
 {  
 **return** [  
 'id' => 'ID',  
 'email' => 'Email',  
 'username' => 'Username',  
 'password\_hash' => 'Password Hash',  
 'salt' => 'Salt',  
 'auth\_key' => 'Auth Key',  
 'verification\_token' => 'Verification Token',  
 'status' => 'Status',  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *array  
 \*/* **public function** getRolesArray()  
 {  
 **return** ArrayHelper::*map*(Yii::*$app*->authManager->getRoles(), 'name', 'name');  
 }  
  
 */\*\*  
 \* Check if user is logged  
 \*  
 \** ***@return*** *bool  
 \*/* **public function** isLogged()  
 {  
 **return** Yii::*$app*->user->identity->getId() === $this->id;  
 }  
  
 */\*\*  
 \** ***@param*** *$id  
 \** ***@return*** *bool  
 \*/* **public function** hasFollower($id)  
 {  
 **return** $this->getFriendships()->where(['second\_user\_id' => $id])->exists();  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getFriendships()  
 {  
 **return** $this->hasMany(Friendship::*className*(), ['first\_user\_id' => 'id']);  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/  
  
 /\*\*  
 \** ***@param*** *$id  
 \** ***@return*** *bool  
 \*/* **public function** hasConversation($id)  
 {  
 **return** $this->getConversations()->where(['id' => $id])->exists();  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getConversations()  
 {  
 **return** $this->hasMany(Conversation::*className*(), ['id' => 'conversation\_id'])  
 ->via('userHasConversations');  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getFollowers()  
 {  
 **return** $this->hasMany(User::*className*(), ['id' => 'second\_user\_id'])->via('friendships');  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getProfile()  
 {  
 **return** $this->hasOne(Profile::*className*(), ['user\_id' => 'id']);  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getConversationMessages()  
 {  
 **return** $this->hasMany(ConversationMessage::*className*(), ['user\_id' => 'id']);  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getUserHasConversations()  
 {  
 **return** $this->hasMany(UserHasConversation::*className*(), ['user\_id' => 'id']);  
 }  
  
 */\*\*  
 \* Get status name  
 \*  
 \** ***@return*** *mixed  
 \*/* **public function** getStatusName()  
 {  
 **return self**::*getStatuses*()[$this->status];  
 }  
  
 */\*\*  
 \** ***@return*** *int|string  
 \*/* **public function** getId()  
 {  
 **return** $this->id;  
 }  
  
 **public function** getUsername()  
 {  
 **return** $this->username;  
 }  
  
 */\*\*  
 \** ***@param*** *string $authKey  
 \** ***@return*** *bool  
 \*/* **public function** validateAuthKey($authKey)  
 {  
 **return** $this->getAuthKey() === $authKey;  
 }  
  
 */\*\*  
 \** ***@return*** *string  
 \*/* **public function** getAuthKey()  
 {  
 **return** $this->auth\_key;  
 }  
  
 */\*\*  
 \** ***@param*** *$password\_hash  
 \** ***@return*** *string  
 \** ***@throws*** *\yii\base\Exception  
 \*/* **public function** setPassword($password\_hash)  
 {  
 **return** $this->password\_hash = Yii::*$app*->security->generatePasswordHash($password\_hash);  
 }  
  
 */\*\*  
 \** ***@param*** *$password\_hash  
 \** ***@return*** *bool  
 \*/* **public function** validatePassword($password\_hash)  
 {  
 **return** Yii::*$app*->security->validatePassword($password\_hash, $this->password\_hash);  
 }  
  
 */\*\*  
 \** ***@param*** *bool $insert  
 \** ***@return*** *bool  
 \** ***@throws*** *\yii\base\Exception  
 \*/* **public function** beforeSave($insert)  
 {  
 **if** (**parent**::*beforeSave*($insert)) {  
 **if** ($this->isNewRecord) {  
 $this->generateAuthKey();  
 $this->generateVerificationToken();  
 }  
 **return true**;  
 }  
 **return false**;  
 }  
  
 */\*\*  
 \** ***@return*** *string  
 \** ***@throws*** *\yii\base\Exception  
 \*/* **public function** generateAuthKey()  
 {  
 **return** $this->auth\_key = Yii::*$app*->security->generateRandomString();  
 }  
  
 */\*\*  
 \** ***@return*** *string  
 \** ***@throws*** *\yii\base\Exception  
 \*/* **public function** generateVerificationToken()  
 {  
 **return** $this->verification\_token = Yii::*$app*->security->generateRandomString();  
 }  
  
 **public  
 function** sendConfirmEmail()  
 {  
 **return** Yii::*$app*->mailer->compose(['html' => 'user-verify-html'], ['user' => $this])  
 ->setFrom([Yii::*$app*->params['adminEmail'] => Yii::*$app*->name])  
 ->setTo($this->email)  
 ->setSubject('Email confirmation for ' . Yii::*$app*->name)  
 ->send();  
 }  
  
 **public  
 function** verify()  
 {  
 $this->verification\_token = **null**;  
 $this->status = **self**::*STATUS\_ACTIVE*;  
 **return** $this->save();  
 }  
  
 */\*\*  
 \** ***@param*** *bool $runValidation  
 \** ***@param*** *null $attributeNames  
 \** ***@return*** *bool  
 \** ***@throws*** *\Exception  
 \*/* **public function** save($runValidation = **true**, $attributeNames = **null**)  
 {  
 **if** (**parent**::*save*($runValidation, $attributeNames)) {  
 **if** (!**isset**($this->profile)) {  
 $profile = **new** Profile();  
 $profile->setDefaultImage();  
 $profile->link('user', $this);  
 }  
  
 $authManager = Yii::*$app*->authManager;  
 $role = !**isset**($this->role) ? 'user' : $this->role;  
 **if** (!$authManager->getAssignment($role, $this->id)) {  
 **if** (count($authManager->getAssignments($this->id))) {  
 $authManager->revokeAll($this->id);  
 }  
  
 $userRole = $authManager->getRole($role);  
 $authManager->assign($userRole, $this->id);  
 }  
 **return true**;  
 }  
 **return false**;  
 }  
}

Далі впроваджуємо контролер користувача. Добавляємо в поведінку всі потрібні правила доступу до дій(action) і відповідно потрібні нам дії:

controllers/UserController.php

**<?php  
  
namespace** app\controllers;  
  
**use** app\models\Friendship;  
**use** app\models\Profile;  
**use** Yii;  
**use** app\models\User;  
**use** app\models\UserSearch;  
**use** yii\base\Model;  
**use** yii\data\ActiveDataProvider;  
**use** yii\filters\AccessControl;  
**use** yii\web\Controller;  
**use** yii\web\ForbiddenHttpException;  
**use** yii\web\NotFoundHttpException;  
**use** yii\filters\VerbFilter;  
**use** yii\web\UploadedFile;  
  
*/\*\*  
 \* UserController implements the CRUD actions for User model.  
 \*/***class** UserController **extends** Controller  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** behaviors()  
 {  
 **return** [  
 'access' => [  
 'class' => AccessControl::*className*(),  
 'rules' => [  
 [  
 'allow' => **true**,  
 'roles' => ['admin'],  
 ],  
 [  
 'allow' => **true**,  
 'actions' => ['update', 'index', 'display', 'followers', 'find', 'follow', 'unfollow', 'reset-image', 'view'],  
 'roles' => ['moderator'],  
 ],  
 [  
 'allow' => **true**,  
 'actions' => ['update'],  
 'permissions' => ['updateOwnProfile'],  
 'roleParams' => [  
 'user\_id' => Yii::*$app*->request->get('id'),  
 ]  
 ],  
 [  
 'allow' => **true**,  
 'actions' => ['display', 'followers', 'find', 'follow', 'unfollow', 'reset-image'],  
 'roles' => ['user'],  
 ],  
 ]  
 ],  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Lists all User models.  
 \** ***@return*** *mixed  
 \*/* **public function** actionIndex()  
 {  
 $searchModel = **new** UserSearch();  
 $dataProvider = $searchModel->search(Yii::*$app*->request->queryParams);  
  
 **return** $this->render('index', [  
 'searchModel' => $searchModel,  
 'dataProvider' => $dataProvider,  
 ]);  
 }  
  
 */\*\*  
 \* Displays a single User model.  
 \** ***@param*** *string $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* **public function** actionView($id)  
 {  
 **return** $this->render('view', [  
 'model' => $this->findModel($id),  
 ]);  
 }  
  
 */\*\*  
 \* Finds the User model based on its primary key value.  
 \* If the model is not found, a 404 HTTP exception will be thrown.  
 \** ***@param*** *string $id  
 \** ***@return*** *User the loaded model  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* **protected function** findModel($id)  
 {  
 **if** (($model = User::*findOne*($id)) !== **null**) {  
 **return** $model;  
 }  
  
 **throw new** NotFoundHttpException('The requested page does not exist.');  
 }  
  
 */\*\*  
 \** ***@return*** *string|\yii\web\Response  
 \** ***@throws*** *\Exception  
 \*/* **public function** actionCreate()  
 {  
 $user = **new** User();  
 $profile = **new** Profile();  
  
 **if** ($user->load(Yii::*$app*->request->post()) && $profile->load(Yii::*$app*->request->post())) {  
 **if** ($user->save() && $profile->save()) {  
 **return** $this->redirect(['view', 'id' => $user->id]);  
 }  
 }  
  
 **return** $this->render('create', [  
 'user' => $user,  
 'profile' => $profile,  
 ]);  
 }  
  
 */\*\*  
 \* Updates an existing User model.  
 \* If update is successful, the browser will be redirected to the 'view' page.  
 \** ***@param*** *string $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* **public function** actionUpdate($id)  
 {  
 */\*\** ***@var*** *User $user \*/* $user = $this->findModel($id);  
 */\*\** ***@var*** *Profile $profile \*/* $profile = Profile::*find*()->where(['user\_id' => $id])->one();  
  
 **if** ($user->load(Yii::*$app*->request->post()) && $profile->load(Yii::*$app*->request->post())) {  
 **if** ($user->save() && $profile->save()) {  
 **return** $this->redirect(['display', 'id' => $user->id]);  
 }  
 }  
  
 **return** $this->render('update', [  
 'user' => $user,  
 'profile' => $profile  
 ]);  
 }  
  
 */\*\*  
 \* Set profile image to default image  
 \*  
 \** ***@param*** *$id  
 \** ***@return*** *\yii\web\Response  
 \** ***@throws*** *NotFoundHttpException  
 \*/* **public function** actionResetImage($id)  
 {  
 */\*\** ***@var*** *User $user \*/* $user = $this->findModel($id);  
 $user->profile->setDefaultImage();  
 $user->profile->save();  
 **return** $this->redirect(['user/update', 'id' => $id]);  
 }  
  
 */\*\*  
 \** ***@param*** *$id  
 \** ***@return*** *\yii\web\Response  
 \** ***@throws*** *NotFoundHttpException  
 \** ***@throws*** *\Throwable  
 \** ***@throws*** *\yii\db\StaleObjectException  
 \*/* **public function** actionDelete($id)  
 {  
 $this->findModel($id)->delete();  
  
 **return** $this->redirect(['index']);  
 }  
  
 */\*\*  
 \* User side profile view  
 \*  
 \** ***@param*** *$id  
 \** ***@return*** *string  
 \** ***@throws*** *NotFoundHttpException  
 \*/* **public function** actionDisplay($id)  
 {  
 $model = $this->findModel($id);  
  
 $dataProvider = **new** ActiveDataProvider([  
 'query' => $model->getFollowers(),  
 'pagination' => [  
 'pageSize' => 3  
 ]  
 ]);  
  
 **return** $this->render('user', [  
 'model' => $model->profile,  
 'dataProvider' => $dataProvider,  
 ]);  
 }  
  
 */\*\*  
 \* Follow user  
 \*  
 \** ***@param*** *$id  
 \** ***@return*** *\yii\web\Response  
 \** ***@throws*** *NotFoundHttpException  
 \*/* **public function** actionFollow($id)  
 {  
 $model = $this->findModel($id);  
 */\*\** ***@var*** *User $user \*/* $user = Yii::*$app*->user->identity;  
  
 $friendship = **new** Friendship();  
 $friendship->link('firstUser', $user);  
 $friendship->link('secondUser', $model);  
  
 **return** $this->redirect(['user/display', 'id' => $id]);  
 }  
  
 */\*\*  
 \* Unfollow user  
 \*  
 \** ***@param*** *$id  
 \** ***@return*** *bool|\yii\web\Response  
 \** ***@throws*** *\Throwable  
 \** ***@throws*** *\yii\db\StaleObjectException  
 \*/* **public function** actionUnfollow($id)  
 {  
 */\*\** ***@var*** *User $user \*/* $user = Yii::*$app*->user->identity;  
 **if** ($user->getFriendships()->where(['second\_user\_id' => $id])->one()->delete()) {  
 **return** $this->redirect(['user/display', 'id' => $id]);  
 }  
 **return false**;  
 }  
  
 */\*\*  
 \* Shows user`s followers  
 \*  
 \** ***@param*** *$id  
 \** ***@return*** *string  
 \** ***@throws*** *NotFoundHttpException  
 \*/* **public function** actionFollowers($id)  
 {  
 $user = $this->findModel($id);  
  
 $searchModel = **new** UserSearch();  
 $dataProvider = $searchModel->search(Yii::*$app*->request->queryParams, $id);  
  
 **return** $this->render('followers', [  
 'user' => $user,  
 'dataProvider' => $dataProvider,  
 'searchModel' => $searchModel,  
 ]);  
 }  
  
 **public function** actionFind()  
 {  
 $searchModel = **new** UserSearch();  
 $dataProvider = $searchModel->search(Yii::*$app*->request->queryParams);  
  
 **return** $this->render('find', [  
 'dataProvider' => $dataProvider,  
 'searchModel' => $searchModel,  
 ]);  
 }  
}

Тепер можна провести переілк всіх видів(view)

views/user/…

index.php

**<?php  
  
use** yii\helpers\Html;  
**use** yii\grid\GridView;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$searchModel app\models\UserSearch \*/  
/\** ***@var*** *$dataProvider yii\data\ActiveDataProvider \*/*$this->title = 'Users';  
$this->params['breadcrumbs'][] = $this->title;  
**?>**<div class="user-index">  
  
 <h1>**<?=** Html::*encode*($this->title) **?>**</h1>  
 **<?php** // echo $this->render('\_search', ['model' => $searchModel]); **?>** <p>  
 **<?=** Html::*a*('Create User', ['create'], ['class' => 'btn btn-success']) **?>** </p>  
  
 **<?=** GridView::*widget*([  
 'dataProvider' => $dataProvider,  
 'filterModel' => $searchModel,  
 'columns' => [  
 ['class' => 'yii\grid\SerialColumn'],  
  
 'id',  
 'email:email',  
 'username',  
  
 ['class' => 'yii\grid\ActionColumn'],  
 ],  
 ]); **?>**</div>

update.php

**<?php  
  
use** yii\helpers\Html;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$user app\models\User \*/  
/\** ***@var*** *$profile app\models\Profile \*/*$this->title = 'Profile settings';  
**?>**<div class="user-update">  
 <h1>**<?=** Html::*encode*($this->title) **?>**</h1>  
 **<?=** $this->render('\_form', [  
 'user' => $user,  
 'profile' => $profile,  
 ]) **?>**</div>

create.php

**<?php  
  
use** yii\helpers\Html;  
  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$user app\models\User \*/  
/\** ***@var*** *$profile app\models\Profile \*/*$this->title = 'Create User';  
$this->params['breadcrumbs'][] = ['label' => 'Users', 'url' => ['index']];  
$this->params['breadcrumbs'][] = $this->title;  
**?>**<div class="user-create">  
  
 <h1>**<?=** Html::*encode*($this->title) **?>**</h1>  
  
 **<?=** $this->render('\_form', [  
 'user' => $user,  
 'profile' => $profile,  
 ]) **?>**</div>

\_form.php

**<?php  
  
use** yii\helpers\Html;  
**use** yii\widgets\ActiveForm;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$user app\models\User \*/  
/\** ***@var*** *$profile app\models\Profile \*/  
/\** ***@var*** *$form yii\widgets\ActiveForm \*/***?>**<div class="user-form">  
  
 <div class="row">  
 <hr>  
 **<?php** $form = ActiveForm::*begin*(); **?>** <div class="col-md-4">  
 **<?php  
  
 if** (**isset**($profile->avatar)) {  
 **echo** Html::*img*($profile->avatar, ['class' => 'center-block', 'id' => 'user-avatar', 'style' => 'width:50%; height:20%']);  
 }  
 **?>  
  
 <?=** $form->field($profile, 'imageFile')->widget(\kartik\file\FileInput::*className*(), [  
 'options' => [  
 'accept' => 'image/\*',  
 ],  
 ]) **?>  
  
 <?=** Html::*a*('Set default image', ['user/reset-image', 'id' => $profile->id], ['class' => 'btn btn-danger']) **?>** </div>  
 <div class="col-md-8">  
  
 **<? if** (Yii::*$app*->authManager->checkAccess(Yii::*$app*->user->identity->getId(), 'admin')): **?>  
 <?=** $form->field($user, 'role')->dropDownList($user->getRolesArray()) **?>** <hr>  
 **<?php endif**; **?>  
  
 <? if** (Yii::*$app*->authManager->checkAccess(Yii::*$app*->user->identity->getId(), 'moderator')): **?>  
 <?=** $form->field($user, 'status')->dropDownList($user->getStatuses()) **?>** <hr>  
 **<?php endif**; **?>  
  
  
  
 <?=** $form->field($user, 'email')->textInput(['maxlength' => **true**]) **?>  
  
 <?=** $form->field($user, 'username')->textInput(['maxlength' => **true**]) **?>  
  
 <?=** $form->field($profile, 'first\_name')->textInput(['maxlength' => **true**]) **?>  
  
 <?=** $form->field($profile, 'last\_name')->textInput(['maxlength' => **true**]) **?>  
  
 <?=** $form->field($profile, 'birth\_date')->widget(\kartik\date\DatePicker::*className*(), [  
 'options' => ['placeholder' => 'Enter birth date ...'],  
 'pluginOptions' => [  
 'autoclose' => **true** ]  
 ]) **?>  
  
 <?=** $form->field($profile, 'details')->textarea(['rows' => 6]) **?>** <div class="form-group">  
 **<?=** Html::*submitButton*('Save', ['class' => 'btn btn-success']) **?>** </div>  
 </div>  
 **<?php** ActiveForm::*end*(); **?>** </div>  
  
</div>

\_search.php

**<?php  
  
use** yii\helpers\Html;  
**use** yii\widgets\ActiveForm;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\UserSearch \*/  
/\** ***@var*** *$form yii\widgets\ActiveForm \*/***?>**<div class="user-search">  
  
 **<?php** $form = ActiveForm::*begin*([  
 'method' => 'get',  
 ]); **?>  
  
 <?=** $form->field($model, 'email') **?>  
  
 <?=** $form->field($model, 'username') **?>  
  
 <?=** $form->field($model, 'first\_name') **?>  
  
 <?=** $form->field($model, 'last\_name') **?>** <div class="form-group">  
 **<?=** Html::*submitButton*('Search', ['class' => 'btn btn-primary']) **?>  
 <?=** Html::*resetButton*('Reset', ['class' => 'btn btn-default']) **?>** </div>  
  
 **<?php** ActiveForm::*end*(); **?>**</div>

view.php

**<?php  
  
use** yii\helpers\Html;  
**use** yii\widgets\DetailView;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\User \*/*$this->title = $model->id;  
$this->params['breadcrumbs'][] = ['label' => 'Users', 'url' => ['index']];  
$this->params['breadcrumbs'][] = $this->title;  
**?>**<div class="user-view">  
  
 <h1>**<?=** Html::*encode*($this->title) **?>**</h1>  
  
 <p>  
 **<?=** Html::*a*('Update', ['update', 'id' => $model->id], ['class' => 'btn btn-primary']) **?>  
 <?=** Html::*a*('Delete', ['delete', 'id' => $model->id], [  
 'class' => 'btn btn-danger',  
 'data' => [  
 'confirm' => 'Are you sure you want to delete this item?',  
 'method' => 'post',  
 ],  
 ]) **?>** </p>  
  
 **<?=** DetailView::*widget*([  
 'model' => $model,  
 'attributes' => [  
 'id',  
 'email:email',  
 'username',  
 'password\_hash',  
 'salt',  
 'auth\_key',  
 'verification\_token',  
 'status',  
 ],  
 ]) **?>**</div>

user.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 19.11.2018  
 \* Time: 14:30  
 \*/***use** yii\helpers\Html;  
**use** yii\widgets\DetailView;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\Profile \*/  
/\** ***@var*** *$user \app\models\User \*/  
/\** ***@var*** *$dataProvider \yii\data\ActiveDataProvider \*/***?>**<div class="profile-view">  
 **<?php** $user = Yii::*$app*->user->identity;  
  
 **?>** <div class="row">  
 <!--left col-->  
 <div class="col-sm-3">  
 <div class="row text-center">  
 **<?=** Html::*img*($model->avatar, ['class' => 'img-thumbnail img-responsive center-block', 'id' => 'user-avatar']) **?>** <h1>**<?=** $model->getName() **?>**</h1>  
  
 **<?php if** (!($model->user->isLogged())): **?>** <div class="btn-group">  
 **<?php if** ($user->hasFollower($model->user\_id)): **?>  
 <?=** Html::*a*('Unfollow', ['user/unfollow', 'id' => $model->user\_id], ['class' => 'btn btn-danger']) **?>  
 <?php else**: **?>  
 <?=** Html::*a*('Follow', ['user/follow', 'id' => $model->user\_id], ['class' => 'btn btn-success']) **?>  
 <?php endif**; **?>  
  
 <?=** Html::*a*('Send message', ['conversation/get-conversation', 'id' => $model->user\_id], ['class' => 'btn btn-info']) **?>** </div>  
 **<?php else**: **?>  
 <?=** Html::*a*('Settings', ['user/update', 'id' => $model->user\_id], ['class' => 'btn btn-danger']) **?>  
 <?php endif**; **?>** </div>  
  
 <br>  
  
 <div class="row">  
 <div class="panel panel-default target">  
 <div class="panel-heading">Profile info</div>  
 <div class="panel-body">  
 **<?=** DetailView::*widget*([  
 'model' => $model,  
 'options' => ['class' => 'list-group', 'tag' => 'ul'],  
 'template' => '<li class="list-group-item text-right"><span class="pull-left"><strong>{label}</strong></span>{value}</li>',  
 'attributes' => [  
 [  
 'attribute' => 'birth\_date',  
 'visible' => **isset**($model->birth\_date)  
 ],  
 [  
 'label' => 'Followers',  
 'value' => count($model->user->friendships)  
 ],  
 [  
 'label' => 'Role',  
 'value' => **function** (\app\models\Profile $model) {  
 $roles = \yii\helpers\ArrayHelper::*getColumn*($model->user->getRoles(), 'name');  
 **return** implode(',', $roles);  
 },  
 ],  
 ],  
 ]) **?>** </div>  
 </div>  
 </div>  
 </div>  
  
 <div class="col-sm-9">  
 **<?php if** (**isset**($model->details)): **?>** <div class="panel panel-default">  
 <div class="panel-heading">**<?= isset**($model->first\_name) ? $model->getFullName() : $model->getUsername() **?>** 's Bio  
 </div>  
 <div class="panel-body"> **<?=** $model->details **?>** </div>  
 </div>  
 **<?php endif**; **?>** <div class="panel panel-default target">  
 **<?=** Html::*a*('Followers', ['user/followers', 'id' => $model->user\_id], ['class' => 'panel-heading center-block']) **?>** <div class="panel-body">  
 **<?=** \yii\widgets\ListView::*widget*([  
 'dataProvider' => $dataProvider,  
 'itemOptions' => [  
 'tab' => 'div',  
 'class' => 'col-md-4'  
 ],  
 'itemView' => **function** (\app\models\User $model, $key, $index, \yii\widgets\ListView $widget) {  
 **return** $widget->view->render('\_user\_follower', [  
 'model' => $model,  
 ]);  
 },  
 'layout' => '<div class="row">{items}</div> <div class="container">{pager}{summary}</div>',  
 ]); **?>** </div>  
 </div>  
 </div>  
 </div>  
</div>

followers.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 28.11.2018  
 \* Time: 10:30  
 \*/***use** yii\widgets\ActiveForm;  
**use** yii\widgets\ListView;  
**use** yii\helpers\Html;  
  
*/\** ***@var*** *$this \yii\web\View \*/  
/\** ***@var*** *$dataProvider \yii\data\ActiveDataProvider \*/  
/\** ***@var*** *$user \app\models\User \*/  
/\** ***@var*** *$searchModel app\models\UserSearch \*/*$this->title = 'Followers';  
$this->params['breadcrumbs'][] = ['label' => $user->profile->getName(), 'url' => ['display', 'id' => $user->getId()]];  
$this->params['breadcrumbs'][] = $this->title;  
**?>**<div class="user-followers">  
  
 **<?php** $form = ActiveForm::*begin*([  
 'method' => 'get',  
 ]); **?>  
  
 <?=** $form->field($searchModel, 'full\_name')->textInput(['placeholder' => 'Start typing any name of word'])->label(**false**) **?>** <div class="form-group">  
 **<?=** Html::*submitButton*('Search', ['class' => 'btn btn-primary']) **?>  
 <?=** Html::*resetButton*('Reset', ['class' => 'btn btn-default']) **?>** </div>  
  
 **<?php** ActiveForm::*end*(); **?>** <p>  
 **<?=** Html::*a*('Add user', ['user/find'], ['class' => 'btn btn-success']) **?>** </p>  
  
 **<?=** ListView::*widget*([  
 'dataProvider' => $dataProvider,  
 'itemOptions' => [  
 'tab' => 'div',  
 'class' => 'col-md-3'  
 ],  
 'itemView' => **function** (\app\models\User $model, $key, $index, \yii\widgets\ListView $widget) {  
 **return** $widget->view->render('\_user\_follower', [  
 'model' => $model,  
 ]);  
 },  
 'layout' => '<div class="row">{items}</div> <div class="container">{pager}{summary}</div>',  
 ]);  
 **?>**</div>

find.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 28.11.2018  
 \* Time: 10:30  
 \*/***use** yii\widgets\ActiveForm;  
**use** yii\widgets\ListView;  
**use** yii\helpers\Html;  
  
*/\** ***@var*** *$this \yii\web\View \*/  
/\** ***@var*** *$dataProvider \yii\data\ActiveDataProvider \*/  
/\** ***@var*** *$searchModel app\models\UserSearch \*/*$this->title = 'Find user';  
$this->params['breadcrumbs'][] = $this->title;  
**?>**<div class="user-find">  
  
 **<?=** $this->render('\_search', ['model' => $searchModel]) **?>  
  
 <?=** ListView::*widget*([  
 'dataProvider' => $dataProvider,  
 'itemOptions' => [  
 'tab' => 'div',  
 'class' => 'col-md-3'  
 ],  
 'itemView' => **function** (\app\models\User $model, $key, $index, \yii\widgets\ListView $widget) {  
 **return** $widget->view->render('\_user\_follower', [  
 'model' => $model,  
 ]);  
 },  
 'layout' => '<div class="row">{items}</div> <div class="container">{pager}{summary}</div>',  
 ]);  
 **?>**</div>

\_user\_follower.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 20.11.2018  
 \* Time: 14:07  
 \*/***use** yii\helpers\Html;  
  
*/\*\*  
 \** ***@var*** *$model \app\models\User  
 \** ***@var*** *$user \app\models\User  
 \*/***?>**<div class="thumbnail">  
 **<?=** Html::*a*(  
 Html::*img*($model->profile->avatar, ['class' => 'follower-avatar']),  
 ['user/display', 'id' => $model->id]  
 ); **?>** <div class="caption">  
 <p class="pull-left">  
 **<?=** $model->profile->getName() **?>** </p>  
  
 **<?php** $user = Yii::*$app*->user->identity;  
 **?>** <div class="btn-group pull-right">  
 <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown">  
 <i class="glyphicon glyphicon-option-vertical"></i>  
 </button>  
 <ul class="dropdown-menu" role="menu">  
 **<?php if** (!($model->isLogged())): **?>** <li>  
 **<?php if** ($user->hasFollower($model->id)): **?>  
 <?=** Html::*a*('Unfollow', ['user/unfollow', 'id' => $model->id]) **?>  
 <?php else**: **?>  
 <?=** Html::*a*('Follow', ['user/follow', 'id' => $model->id]) **?>  
 <?php endif**; **?>** </li>  
 <li>  
 **<?=** Html::*a*('Send message', ['conversation/get-conversation', 'id' => $model->id]) **?>** </li>  
 **<?php endif**; **?>** <li>  
 **<?=** Html::*a*('Show followers', ['user/followers', 'id' => $model->id]) **?>** </li>  
 </ul>  
 </div>  
  
 <div class="clearfix"></div>  
 </div>  
</div>

# Бесіда.

Рефакторимо модель і добавляємо потрібні методи, в поведінки(behaviours) дописуємо трігери присвоювання автора, редактора та дати створення/редагування.

models/Conversation.php

**<?php  
  
namespace** app\models;  
  
**use** phpDocumentor\Reflection\Types\Integer;  
**use** Yii;  
**use** yii\base\Theme;  
**use** yii\db\Expression;  
  
*/\*\*  
 \* This is the model class for table "conversation".  
 \*  
 \** ***@property*** *string $id  
 \** ***@property*** *string $title  
 \*  
 \** ***@property*** *User[] $participants  
 \** ***@property*** *integer[] $user\_ids  
 \** ***@property*** *ConversationMessage[] $conversationMessages  
 \** ***@property*** *UserHasConversation[] $userHasConversations  
 \*/***class** Conversation **extends** \yii\db\ActiveRecord  
{  
 */\*\*  
 \* Find dialog conversation between two users  
 \*  
 \** ***@param*** *$userId  
 \** ***@param*** *$currentUserId  
 \** ***@return*** *Conversation|array|null|\yii\db\ActiveRecord  
 \*/* **public static function** findDialog($userId, $currentUserId)  
 {  
 $subQuery = **self**::*find*()  
 ->select(['c.\*', 'mc' => **new** Expression('COUNT(m.user\_id)')])  
 ->from(['c' => **self**::*tableName*()])  
 ->leftJoin(['u1' => UserHasConversation::*tableName*()], 'u1.conversation\_id = c.id')  
 ->leftJoin(['u2' => UserHasConversation::*tableName*()], 'u2.conversation\_id = c.id')  
 ->leftJoin(['m' => UserHasConversation::*tableName*()], 'm.conversation\_id = c.id')  
 ->groupBy('c.id')  
 ->andWhere(['u1.user\_id' => (int)$userId])  
 ->andWhere(['u2.user\_id' => (int)$currentUserId]);  
  
 $query = **self**::*find*()  
 ->from(['c' => $subQuery])  
 ->where(['mc' => 2]);  
  
 **return** $query->one();  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public static function** tableName()  
 {  
 **return** 'conversation';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** rules()  
 {  
 **return** [  
 [['title', 'user\_ids'], 'required'],  
 [['title'], 'string', 'max' => 64],  
 [['user\_ids'], 'each', 'rule' => ['integer']],  
 ];  
 }  
  
 **public function** behaviors()  
 {  
 **return** [  
 [  
 'class' => \voskobovich\behaviors\ManyToManyBehavior::*className*(),  
 'relations' => [  
 'user\_ids' => 'participants',  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** attributeLabels()  
 {  
 **return** [  
 'id' => 'ID',  
 'title' => 'Title',  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getConversationMessages()  
 {  
 **return** $this->hasMany(ConversationMessage::*className*(), ['conversation\_id' => 'id']);  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getUserHasConversations()  
 {  
 **return** $this->hasMany(UserHasConversation::*className*(), ['conversation\_id' => 'id']);  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getParticipants()  
 {  
 **return** $this->hasMany(User::*className*(), ['id' => 'user\_id'])->via('userHasConversations');  
 }  
  
 **public function** beforeSave($insert)  
 {  
 $this->user\_ids = array\_merge($this->user\_ids, [Yii::*$app*->user->getId()]);  
 **return parent**::*beforeSave*($insert);  
 }  
}

Тепер рефакторимо контролер. Відповідно встановлюємо в поведінки правила доступу і т.д.

controllers/ConversationController.php

**<?php  
  
namespace** app\controllers;  
  
**use** app\models\ConversationForm;  
**use** app\models\ConversationMessage;  
**use** app\models\User;  
**use** app\models\UserHasConversation;  
**use** Yii;  
**use** app\models\Conversation;  
**use** app\models\ConversationSearch;  
**use** yii\data\ActiveDataProvider;  
**use** yii\filters\AccessControl;  
**use** yii\web\Controller;  
**use** yii\web\NotFoundHttpException;  
**use** yii\filters\VerbFilter;  
**use** yii\web\Response;  
  
*/\*\*  
 \* ConversationController implements the CRUD actions for Conversation model.  
 \*/***class** ConversationController **extends** Controller  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** behaviors()  
 {  
 **return** [  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 'access' => [  
 'class' => AccessControl::*className*(),  
 'rules' => [  
 [  
 'allow' => **true**,  
 'roles' => ['admin'],  
 ],  
 [  
 'allow' => **true**,  
 'actions' => ['index', 'display', 'get-conversation', 'check'],  
 'roles' => ['moderator'],  
 ],  
 [  
 'allow' => **true**,  
 'actions' => ['update'],  
 'permissions' => ['updateOwnConversation'],  
 'roleParams' => [  
 'conversation\_id' => Yii::*$app*->request->get('id'),  
 ]  
 ],  
 [  
 'allow' => **true**,  
 'actions' => ['create', 'delete', 'display', 'get-conversation', 'check'],  
 'roles' => ['user'],  
 ],  
 ],  
 ]  
 ];  
 }  
  
 */\*\*  
 \* Lists all Conversation models.  
 \** ***@return*** *mixed  
 \*/* **public function** actionIndex()  
 {  
 $searchModel = **new** ConversationSearch();  
 $dataProvider = **new** ActiveDataProvider([  
 'query' => Conversation::*find*()  
 ->with('participants')  
 ]);  
  
 **return** $this->render('index', [  
 'searchModel' => $searchModel,  
 'dataProvider' => $dataProvider,  
 ]);  
 }  
  
 */\*\*  
 \* Displays a single Conversation model.  
 \** ***@param*** *string $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* **public function** actionView($id)  
 {  
 **return** $this->render('view', [  
 'model' => $this->findModel($id),  
 ]);  
 }  
  
 */\*\*  
 \* Finds the Conversation model based on its primary key value.  
 \* If the model is not found, a 404 HTTP exception will be thrown.  
 \** ***@param*** *string $id  
 \** ***@return*** *Conversation the loaded model  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* **protected function** findModel($id)  
 {  
 **if** (($model = Conversation::*findOne*($id)) !== **null**) {  
 **return** $model;  
 }  
  
 **throw new** NotFoundHttpException('The requested page does not exist.');  
 }  
  
 */\*\*  
 \** ***@param*** *$id  
 \** ***@return*** *bool|Response  
 \*/* **public function** actionGetConversation($id)  
 {  
 $user = User::*findOne*($id);  
  
 $currentUserId = Yii::*$app*->user->identity->getId();  
  
 $conversation = Conversation::*findDialog*($user->id, $currentUserId);  
  
 **if** ($conversation == **null**) {  
 $conversation = **new** Conversation([  
 'title' => $user->profile->getName(),  
 'user\_ids' => [$currentUserId, $id],  
 ]);  
 $conversation->save();  
 }  
  
 **return** $this->redirect(['conversation/display', 'id' => $conversation->id]);  
 }  
  
 */\*\*  
 \** ***@param*** *bool|integer $id  
 \** ***@return*** *string  
 \*/* **public function** actionDisplay($id = **false**)  
 {  
 **if** ($id === **false**) {  
 */\*\** ***@var*** *User $user \*/* $user = Yii::*$app*->user->identity;  
 $conversation = $user->getConversations()->one();  
 **if** ($conversation) {  
 $id = $conversation->id;  
 }  
 }  
  
 $model = **new** ConversationMessage();  
 $model->conversation\_id = $id;  
  
 **if** ($model->load(Yii::*$app*->request->post())) {  
 $model->user\_id = Yii::*$app*->user->identity->getId();  
 **if** ($model->save()) {  
 $model = **new** ConversationMessage();  
 }  
 }  
  
 **return** $this->render('display', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Creates a new Conversation model.  
 \* If creation is successful, the browser will be redirected to the 'view' page.  
 \** ***@return*** *mixed  
 \*/* **public function** actionCreate()  
 {  
 $model = **new** Conversation();  
  
 **if** ($model->load(Yii::*$app*->request->post()) && $model->save()) {  
 **return** $this->redirect(['conversation/display', 'id' => $model->id]);  
 }  
  
 **return** $this->render('create', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Updates an existing Conversation model.  
 \* If update is successful, the browser will be redirected to the 'view' page.  
 \** ***@param*** *string $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* **public function** actionUpdate($id)  
 {  
 $model = $this->findModel($id);  
  
 **if** ($model->load(Yii::*$app*->request->post()) && $model->save()) {  
 **return** $this->redirect(['view', 'id' => $model->id]);  
 }  
  
 **return** $this->render('update', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \** ***@param*** *$id  
 \** ***@return*** *Response  
 \** ***@throws*** *NotFoundHttpException  
 \** ***@throws*** *\Throwable  
 \** ***@throws*** *\yii\db\StaleObjectException  
 \*/* **public function** actionDelete($id)  
 {  
 $this->findModel($id)->delete();  
  
 **return** $this->redirect(['conversation/display']);  
 }  
  
 **public function** actionCheck($id, $lastId)  
 {  
 **if** (Yii::*$app*->request->isAjax) {  
 Yii::*$app*->response->format = Response::*FORMAT\_JSON*;  
 */\*\** ***@var*** *array $message \*/* $message = ConversationMessage::*find*()  
 ->select(['id'])  
 ->where(['conversation\_id' => $id,])  
 ->orderBy(['id' => *SORT\_DESC*])  
 ->asArray()  
 ->one();  
 **if** ($message['id'] != $lastId) {  
 **return true**;  
 }  
  
 **return false**;  
 }  
  
 **return** $this->redirect(['conversation/display', 'id' => $id]);  
 }  
}

Перелік видів:

views/conversation/…

\_form.php

**<?php  
  
use** yii\helpers\Html;  
**use** yii\widgets\ActiveForm;  
**use** yii\helpers\ArrayHelper;  
**use** kartik\select2\Select2;  
**use** \app\models\User;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\Conversation \*/  
/\** ***@var*** *$form yii\widgets\ActiveForm \*/***?>**<div class="conversation-form">  
  
 **<?php** $form = ActiveForm::*begin*(); **?>  
  
 <?=** $form->field($model, 'title')->textInput(['maxlength' => **true**]) **?>  
  
 <?=** $form->field($model, 'user\_ids')->widget(Select2::*className*(), [  
 'value' => '',  
 'data' => ArrayHelper::*map*(User::*find*()->where(['NOT', ['id' => Yii::*$app*->user->identity->getId()]])->all(), 'id', 'username'),  
 'options' => [  
 'placeholder' => 'Select user',  
 'multiple' => **true**,  
 ]  
 ]); **?>** <div class="form-group">  
 **<?=** Html::*submitButton*('Save', ['class' => 'btn btn-success']) **?>** </div>  
  
 **<?php** ActiveForm::*end*(); **?>**</div>

\_form\_message.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 11.11.2018  
 \* Time: 13:34  
 \*/***use** yii\widgets\ActiveForm;  
**use** yii\widgets\Pjax;  
**use** yii\helpers\Html;  
  
*/\** ***@var*** *$this \yii\web\View \*/  
/\** ***@var*** *$model \app\models\ConversationMessage \*/  
/\** ***@var*** *$id integer \*/***?>**<div class="reply">  
  
 **<?php** Pjax::*begin*(['id' => 'new\_message']) **?>  
 <?php** $form = ActiveForm::*begin*([  
 'options' => ['data-pjax' => **true**],  
 ]); **?>** <div class="col-md-10 side">  
 **<?=** $form->field($model, 'content')->textarea(['maxlength' => **true**, 'placeholder' => 'Write a message...'])->label(**false**) **?>** </div>  
 <div class="col-md-2 side">  
  
 <div class="form-group">  
 **<?=** Html::*submitButton*('<i class="glyphicon glyphicon-send"></i>', ['class' => 'btn btn-success', 'id' => 'send\_btn']) **?>** </div>  
 </div>  
 <div class="clearfix"></div>  
  
 **<?php** ActiveForm::*end*(); **?>  
 <?php** Pjax::*end*(); **?>**</div>

\_message.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 11.11.2018  
 \* Time: 12:19  
 \*/***use** yii\helpers\Html;  
  
*/\*\*  
 \** ***@var*** *\app\models\ConversationMessage $model  
 \*/***?>**<div class="message pull-left well well-sm col-md-8 ">  
  
 <p class="message-author">  
 **<?=** Html::*a*(Html::*encode*($model->user->username), ['user/view', 'id' => $model->user->id]) **?>** </p>  
 **<?=** nl2br(Html::*encode*($model->content)) **?>** <p class="create-at text-muted pull-right">**<?=** Yii::*$app*->formatter->asDatetime($model->created\_at, 'short') **?>**</p>  
</div>  
<div class="clearfix"></div>

\_own\_message.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 11.11.2018  
 \* Time: 12:40  
 \*/***use** app\models\ConversationMessage;  
**use** Yii\helpers\Html;  
  
*/\*\*  
 \** ***@var*** *ConversationMessage $model  
 \*/***?>**<div class="message pull-right well well-sm col-md-8">  
 **<?=** nl2br(Html::*encode*($model->content)) **?>** <p class="text-muted create-at pull-right">**<?=** Yii::*$app*->formatter->asDatetime($model->created\_at, 'short') **?>**</p>  
</div>  
<div class="clearfix"></div>

\_search.php

**<?php  
  
use** yii\helpers\Html;  
**use** yii\widgets\ActiveForm;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\ConversationSearch \*/  
/\** ***@var*** *$form yii\widgets\ActiveForm \*/***?>**<div class="conversation-search">  
  
 **<?php** $form = ActiveForm::*begin*([  
 'action' => ['index'],  
 'method' => 'get',  
 ]); **?>  
  
 <?=** $form->field($model, 'id') **?>  
  
 <?=** $form->field($model, 'title') **?>** <div class="form-group">  
 **<?=** Html::*submitButton*('Search', ['class' => 'btn btn-primary']) **?>  
 <?=** Html::*resetButton*('Reset', ['class' => 'btn btn-default']) **?>** </div>  
  
 **<?php** ActiveForm::*end*(); **?>**</div>

\_user\_item.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 11.11.2018  
 \* Time: 19:19  
 \*/***use** app\models\User;  
**use** yii\helpers\Html;  
  
*/\*\*  
 \** ***@var*** *$model User  
 \*/***?>**<article class="item" data-key="**<?=** $model->id; **?>**">  
 <p>  
 **<?=** Html::*a*($model->username, ['user/view', 'id' => $model->id]) **?>** </p>  
</article>

create.php

**<?php  
  
use** yii\helpers\Html;  
**use** yii\helpers\ArrayHelper;  
**use** \kartik\select2\Select2;  
  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\Conversation \*/*$this->title = 'Create Conversation';  
$this->params['breadcrumbs'][] = ['label' => 'Conversations', 'url' => ['index']];  
$this->params['breadcrumbs'][] = $this->title;  
**?>**<div class="conversation-create">  
  
 <h1>**<?=** Html::*encode*($this->title) **?>**</h1>  
  
 **<?=** $this->render('\_form', [  
 'model' => $model,  
 ]) **?>**</div>

display.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 11.11.2018  
 \* Time: 11:56  
 \*/***use** yii\helpers\Html;  
**use** yii\web\JqueryAsset;  
**use** yii\widgets\ActiveForm;  
**use** yii\widgets\ListView;  
**use** yii\widgets\Pjax;  
**use** yii\helpers\Url;  
**use** app\widgets\UserConversations;  
**use** app\widgets\ConversationMessages;  
**use** app\widgets\ConversationUsers;  
  
*/\** ***@var*** *$this \yii\web\View \*/  
/\** ***@var*** *$model \app\models\ConversationMessage \*/  
/\** ***@var*** *$dataProvider yii\data\ActiveDataProvider \*/*$actionUrl = Url::*to*(['conversation/check', 'id' => $model->conversation\_id]);  
$this->registerJs(<<<JS  
 window.actionUrl = '$actionUrl';  
  
 $("#menu-toggle").click(function(e) {  
 e.preventDefault();  
 $('#page-content-wrapper').toggleClass('col-md-7');  
 $('#sidebar-wrapper').toggleClass('col-md-2');  
 $('#sidebar-wrapper').toggleClass('toggled');  
 $('#page-content-wrapper').toggleClass('col-md-9');  
 });  
JS  
);  
$this->registerJsFile('js/conversation/main.js', ['depends' => JqueryAsset::*className*()]);  
**?>**<div class="conversation-display">  
 <div class="row">  
  
 <!-- left column-->  
 <div class="col-md-3 conversations side">  
 <div class="header">  
 <p>Conversations</p>  
 </div>  
 <p id="new-conversation">  
 **<?=** Html::*a*('Create new', ['conversation/create'], ['class' => 'btn btn-success center-block']) **?>** </p>  
  
 **<?php** Pjax::*begin*(['id' => 'conversations']) **?>** <div class="user-conversations">  
 **<?=** UserConversations::*widget*([  
 'currentId' => $model->conversation\_id,  
 ]) **?>** </div>  
 **<?php** Pjax::*end*(); **?>** </div>  
  
 **<?php if** ($model->conversation): **?>** <div id="page-content-wrapper" class="col-md-9 conversation side">  
 <!-- conversation header-->  
 <div class="header">  
 <div class="pull-left">  
 <p>**<?=** $model->conversation->title **?>**</p>  
 <span class="text-muted">**<?=** count($model->conversation->participants) **?>** members</span>  
 </div>  
  
 <div class="buttons pull-right">  
 <div class="btn-group">  
 <a href="#menu-toggle" class="btn btn-default" id="menu-toggle">Members</a>  
 </div>  
  
 <div class="btn-group">  
 <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown">  
 <i class="glyphicon glyphicon-option-vertical"></i>  
 </button>  
 <ul class="dropdown-menu" role="menu">  
 <li>  
 **<?=** Html::*a*('Update', ['update', 'id' => $model->conversation\_id]) **?>** </li>  
 <li class="divider"></li>  
 <li>  
 **<?=** Html::*a*('Delete', ['delete', 'id' => $model->conversation\_id], [  
 'data' => [  
 'confirm' => 'Are you sure you want to delete this item?',  
 'method' => 'post',  
 ],  
 ]) **?>** </li>  
 </ul>  
 </div>  
 </div>  
  
 <div class="clearfix"></div>  
 </div>  
  
 <!-- messages content-->  
 <div class="dialog-messages">  
 **<?php** Pjax::*begin*(['id' => 'messages']) **?>  
 <?php echo** ConversationMessages::*widget*([  
 'conversation' => $model->conversation  
 ]) **?>  
 <?php** Pjax::*end*(); **?>** </div>  
  
 <div>  
 **<?=** $this->render('\_form\_message', [  
 'model' => $model,  
 ]) **?>** </div>  
  
 </div>  
  
 <!-- right column-->  
 <div id="sidebar-wrapper" class="hide">  
 <div class="conversation-users">  
 <div class="header">  
 <p>Members</p>  
 </div>  
 **<?=** ConversationUsers::*widget*([  
 'users' => $model->conversation->participants  
 ]) **?>** </div>  
 </div>  
  
 **<?php else**: **?>** <div class="col-md-9">  
 <h4>Start the first conversation</h4>  
 </div>  
 **<?php endif**; **?>** </div>  
</div>

index.php

**<?php  
  
use** yii\helpers\Html;  
**use** yii\grid\GridView;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$searchModel app\models\ConversationSearch \*/  
/\** ***@var*** *$dataProvider yii\data\ActiveDataProvider \*/*$this->title = 'Conversations';  
$this->params['breadcrumbs'][] = $this->title;  
**?>**<div class="conversation-index">  
  
 <h1>**<?=** Html::*encode*($this->title) **?>**</h1>  
 **<?php** // echo $this->render('\_search', ['model' => $searchModel]); **?>** <p>  
 **<?=** Html::*a*('Create Conversation', ['create'], ['class' => 'btn btn-success']) **?>** </p>  
  
 **<?=** GridView::*widget*([  
 'dataProvider' => $dataProvider,  
 'filterModel' => $searchModel,  
 'columns' => [  
 ['class' => 'yii\grid\SerialColumn'],  
  
 'id',  
 'title',  
 [  
 'label' => 'Participants',  
 'attribute' => 'user',  
 'value' => **function** (\app\models\Conversation $model) {  
 **return** implode(', ', array\_map(**function** (\app\models\User $item) {  
 **return** $item->username;  
 }, $model->participants));  
 },  
 ],  
  
 ['class' => 'yii\grid\ActionColumn'],  
 ],  
 ]); **?>**</div>

update.php

**<?php  
  
use** yii\helpers\Html;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\Conversation \*/*$this->title = 'Update Conversation: ' . $model->title;  
$this->params['breadcrumbs'][] = ['label' => 'Conversations', 'url' => ['index']];  
$this->params['breadcrumbs'][] = ['label' => $model->title, 'url' => ['view', 'id' => $model->id]];  
$this->params['breadcrumbs'][] = 'Update';  
**?>**<div class="conversation-update">  
  
 <h1>**<?=** Html::*encode*($this->title) **?>**</h1>  
  
 **<?=** $this->render('\_form', [  
 'model' => $model,  
 ]) **?>**</div>

view.php

**<?php  
  
use** yii\helpers\Html;  
**use** yii\widgets\DetailView;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\Conversation \*/*$this->title = $model->title;  
$this->params['breadcrumbs'][] = ['label' => 'Conversations', 'url' => ['index']];  
$this->params['breadcrumbs'][] = $this->title;  
**?>**<div class="conversation-view">  
  
 <h1>**<?=** Html::*encode*($this->title) **?>**</h1>  
  
 <p>  
 **<?=** Html::*a*('Update', ['update', 'id' => $model->id], ['class' => 'btn btn-primary']) **?>  
 <?=** Html::*a*('Delete', ['delete', 'id' => $model->id], [  
 'class' => 'btn btn-danger',  
 'data' => [  
 'confirm' => 'Are you sure you want to delete this item?',  
 'method' => 'post',  
 ],  
 ]) **?>** </p>  
  
 **<?=** DetailView::*widget*([  
 'model' => $model,  
 'attributes' => [  
 'id',  
 'title',  
 ],  
 ]) **?>**</div>

# RBAC.

За допомогою міграції ініціалізую можливі ролі користувачів ( *“admin”, “moderator”, “user”* ) і створюю профіль адміністратора.

Облікові дані адміністратора:

Username: admin;

Password: admin.

init\_rbac.php

**<?php  
  
use** app\models\User;  
**use** app\rbac\rules\UpdateOwnConversation;  
**use** app\rbac\rules\UpdateOwnProfile;  
**use** yii\db\Migration;  
  
*/\*\*  
 \* Class m181205\_165245\_init\_rbac  
 \*/***class** m181205\_165245\_init\_rbac **extends** Migration  
{  
 */\*\*  
 \** ***@return*** *bool  
 \** ***@throws*** *\yii\base\Exception  
 \** ***@throws*** *Exception  
 \*/* **public function** safeUp()  
 {  
 $authManager = \Yii::*$app*->authManager;  
  
 $user = $authManager->createRole('user');  
 $authManager->add($user);  
  
 $moderator = $authManager->createRole('moderator');  
 $authManager->add($moderator);  
 $authManager->addChild($moderator, $user);  
  
 $admin = $authManager->createRole('admin');  
 $authManager->add($admin);  
 $authManager->addChild($admin, $moderator);  
 $authManager->addChild($admin, $user);  
  
 //update own profile rule  
 $updateProfileRule = **new** UpdateOwnProfile();  
 $authManager->add($updateProfileRule);  
  
 $updateOwnProfile = $authManager->createPermission('updateOwnProfile');  
 $updateOwnProfile->ruleName = $updateProfileRule->name;  
 $authManager->add($updateOwnProfile);  
  
 $authManager->addChild($user, $updateOwnProfile);  
  
 //update own conversation rule  
 $updateConversationRule = **new** UpdateOwnConversation();  
 $authManager->add($updateConversationRule);  
  
 $updateOwnConversation = $authManager->createPermission('updateOwnConversation');  
 $updateOwnConversation->ruleName = $updateConversationRule->name;  
 $authManager->add($updateOwnConversation);  
  
 $authManager->addChild($user, $updateOwnConversation);  
  
 $user = **new** User();  
 $user->username = 'admin';  
 $user->email = 'admin@admin.com';  
 $user->role = 'admin';  
 $user->status = User::*STATUS\_ACTIVE*;  
 $user->setPassword('admin');  
  
 **if** ($user->save()) {  
 **return true**;  
 }  
 **return false**;  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeDown()  
 {  
 $this->delete('user', ['id' => 1]);  
  
 $auth = Yii::*$app*->authManager;  
  
 $auth->removeAll();  
 }  
}

Також створив додаткові правила для оновлення інформації про користувача і бесіди

UpdateOwnConversation.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 03.12.2018  
 \* Time: 19:28  
 \*/***namespace** app\rbac\rules;  
  
  
**use** app\models\User;  
**use** yii\rbac\Item;  
**use** yii\rbac\Rule;  
  
**class** UpdateOwnConversation **extends** Rule  
{  
 **public** $name = 'updateConversation';  
  
 */\*\*  
 \* Executes the rule.  
 \*  
 \** ***@param*** *string|int $user the user ID. This should be either an integer or a string representing  
 \* the unique identifier of a user. See [[\yii\web\User::id]].  
 \** ***@param*** *Item $item the role or permission that this rule is associated with  
 \** ***@param*** *array $params parameters passed to [[CheckAccessInterface::checkAccess()]].  
 \** ***@return*** *bool a value indicating whether the rule permits the auth item it is associated with.  
 \*/* **public function** execute($user, $item, $params)  
 {  
 $model = User::*findOne*($user);  
 **return isset**($params['conversation\_id']) ? $model->hasConversation($params['conversation\_id']) : **false**;  
 }  
}

UpdateOwnProfile.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 03.12.2018  
 \* Time: 19:28  
 \*/***namespace** app\rbac\rules;  
  
  
**use** yii\rbac\Item;  
**use** yii\rbac\Rule;  
  
**class** UpdateOwnProfile **extends** Rule  
{  
 **public** $name = 'updateProfile';  
  
 */\*\*  
 \* Executes the rule.  
 \*  
 \** ***@param*** *string|int $user the user ID. This should be either an integer or a string representing  
 \* the unique identifier of a user. See [[\yii\web\User::id]].  
 \** ***@param*** *Item $item the role or permission that this rule is associated with  
 \** ***@param*** *array $params parameters passed to [[CheckAccessInterface::checkAccess()]].  
 \** ***@return*** *bool a value indicating whether the rule permits the auth item it is associated with.  
 \*/* **public function** execute($user, $item, $params)  
 {  
 **return isset**($params['user\_id']) ? $params['user\_id'] == $user : **false**;  
 }  
}

Є також допоміжні моделі, сущності для яких не потрібно контролерів і видів:

models/LoginForm.php

**<?php  
  
namespace** app\models;  
  
**use** Yii;  
**use** yii\base\Model;  
  
*/\*\*  
 \* LoginForm is the model behind the login form.  
 \*  
 \** ***@property*** *User|null $user This property is read-only.  
 \*  
 \*/***class** LoginForm **extends** Model  
{  
 **public** $username;  
 **public** $password;  
 **public** $rememberMe = **true**;  
  
 */\*\*  
 \** ***@return*** *array the validation rules.  
 \*/* **public function** rules()  
 {  
 **return** [  
 // username and password are both required  
 ['username', 'validateStatus'],  
 [['username', 'password'], 'required'],  
 // rememberMe must be a boolean value  
 ['rememberMe', 'boolean'],  
 // password is validated by validatePassword()  
 ['password', 'validatePassword']  
 ];  
 }  
  
 */\*\*  
 \* Validates the password.  
 \* This method serves as the inline validation for password.  
 \*  
 \** ***@param*** *string $attribute the attribute currently being validated  
 \*/* **public function** validatePassword($attribute)  
 {  
 **if** (!$this->hasErrors()) {  
 $user = $this->getUser();  
  
 **if** (!$user || !$user->validatePassword($this->password)) {  
 $this->addError($attribute, 'Incorrect username or password.');  
 }  
 }  
 }  
  
 */\*\*  
 \* Finds user by [[username]]  
 \*  
 \** ***@return*** *User|null  
 \*/* **public function** getUser()  
 {  
 **return** User::*findByUsername*($this->username);  
 }  
  
 */\*\*  
 \*Validates the user status  
 \*  
 \*/* **public function** validateStatus()  
 {  
 $user = $this->getUser();  
 $status = $user->status;  
 **if** ($status == User::*STATUS\_BLOCKED*) {  
 $this->addError('status', 'Your account has been blocked');  
 Yii::*$app*->session->setFlash('danger', 'Your account has been blocked');  
 } **elseif** ($status == User::*STATUS\_WAIT*) {  
 $this->addError('status', 'Your account is waiting to be confirmed. Check your e-mail');  
 Yii::*$app*->session->setFlash('danger', 'Your account is waiting to be confirmed. Check your e-mail');  
 $user->sendConfirmEmail();  
 }  
 }  
  
 */\*\*  
 \* Logs in a user using the provided username and password.  
 \** ***@return*** *bool whether the user is logged in successfully  
 \*/* **public function** login()  
 {  
 **if** ($this->validate()) {  
 **return** Yii::*$app*->user->login($this->getUser(), $this->rememberMe ? 60 : 0);  
 }  
 **return false**;  
 }  
}

models/SignupForm.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 07.11.2018  
 \* Time: 22:56  
 \*/***namespace** app\models;  
  
**use** Yii;  
**use** yii\base\Model;  
  
**class** SignupForm **extends** Model  
{  
 **public** $username;  
 **public** $email;  
 **public** $password;  
  
 **public function** rules()  
 {  
 **return** [  
 [['username', 'email', 'password'], 'required'],  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *User|bool  
 \** ***@throws*** *\yii\base\Exception  
 \** ***@throws*** *\Exception  
 \*/* **public function** signup()  
 {  
 **if** ($this->validate()) {  
 $user = $this->getUser();  
 **if** ($user->save()) {  
 Yii::*$app*->session->setFlash('success', 'Check your email to confirm the registration.');  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
  
 */\*\*  
 \** ***@return*** *User  
 \** ***@throws*** *\yii\base\Exception  
 \*/* **public function** getUser()  
 {  
 $model = **new** User();  
 $model->username = $this->username;  
 $model->email = $this->email;  
 $model->setPassword($this->password);  
 **return** $model;  
 }  
}

models/ConversationForm.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: vyach  
 \* Date: 12.11.2018  
 \* Time: 22:08  
 \*/***namespace** app\models;  
  
  
**use** Yii;  
**use** yii\base\Model;  
  
**class** ConversationForm **extends** Model  
{  
 **public** $title;  
 **public** $participants;  
  
 **public function** rules()  
 {  
 **return** [  
 [['title', 'users'], 'required'],  
 [['title'], 'string', 'max' => 64],  
 [['users'], 'each', 'rule' => ['integer']],  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *bool  
 \*/* **public function** addUsers()  
 {  
 **if** ($this->validate()) {  
 $model = $this->getConversation();  
 **if** ($model->save()) {  
 $this->participants[] = Yii::*$app*->user->identity->getId();  
 **foreach** ($this->participants **as** $user\_id) {  
 $model->link('participants', User::*findOne*($user\_id), ['conversation\_id' => $model->id]);  
 }  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
  
 */\*\*  
 \** ***@return*** *Conversation  
 \*/* **public function** getConversation()  
 {  
 $model = **new** Conversation();  
 $model->title = $this->title;  
 **return** $model;  
 }  
}

models/UserSearch.php

**<?php  
  
namespace** app\models;  
  
**use** Faker\Provider\th\_TH\Internet;  
**use** Yii;  
**use** yii\base\Model;  
**use** yii\data\ActiveDataProvider;  
**use** app\models\User;  
  
*/\*\*  
 \*  
 \* UserSearch represents the model behind the search form of `app\models\User`.  
 \*/***class** UserSearch **extends** User  
{  
 **public** $first\_name;  
  
 **public** $last\_name;  
  
 **public** $full\_name;  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** rules()  
 {  
 **return** [  
 [['id'], 'integer'],  
 [['email', 'username', 'first\_name', 'last\_name', 'full\_name'], 'safe'],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** scenarios()  
 {  
 // bypass scenarios() implementation in the parent class  
 **return** Model::*scenarios*();  
 }  
  
 */\*\*  
 \* Creates data provider instance with search query applied  
 \*  
 \** ***@param*** *array $params  
 \*  
 \** ***@param*** *$id  
 \** ***@return*** *ActiveDataProvider  
 \*/* **public function** search($params, $id = **null**)  
 {  
 **if** ($id) {  
 $query = User::*findOne*($id)->getFollowers();  
 } **else** {  
 $query = User::*find*();  
 }  
  
 // add conditions that should always apply here  
  
 $dataProvider = **new** ActiveDataProvider([  
 'query' => $query,  
 ]);  
  
 $this->load($params);  
  
 **if** (!$this->validate()) {  
 // uncomment the following line if you do not want to return any records when validation fails  
 // $query->where('0=1');  
 **return** $dataProvider;  
 }  
  
 // grid filtering conditions  
 $query->andFilterWhere([  
 'id' => $this->id,  
 'status' => $this->status,  
 ]);  
  
 $query->andFilterWhere(['like', 'email', $this->email])  
 ->andFilterWhere(['like', 'username', $this->username]);  
  
 $query->joinWith('profile')  
 ->andFilterWhere(['like', 'profile.first\_name', $this->first\_name])  
 ->andFilterWhere(['like', 'profile.last\_name', $this->last\_name]);  
  
 $query->andFilterWhere(['OR',  
 ['like', 'profile.first\_name', $this->full\_name],  
 ['like', 'profile.last\_name', $this->full\_name],  
 ['like', 'user.username', $this->full\_name],  
 ]);  
  
 **return** $dataProvider;  
 }  
}

models/Profile.php

**<?php  
  
namespace** app\models;  
  
**use** Yii;  
**use** yii\web\UploadedFile;  
  
*/\*\*  
 \* This is the model class for table "profile".  
 \*  
 \** ***@property*** *string $id  
 \** ***@property*** *string $avatar  
 \** ***@property*** *string $first\_name  
 \** ***@property*** *string $last\_name  
 \** ***@property*** *string $birth\_date  
 \** ***@property*** *string $details  
 \** ***@property*** *string $user\_id  
 \** ***@property*** *string $gallery\_id  
 \*  
 \** ***@property*** *Gallery $gallery  
 \** ***@property*** *User $user  
 \*/***class** Profile **extends** \yii\db\ActiveRecord  
{  
 **public** $imageFile;  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public static function** tableName()  
 {  
 **return** 'profile';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** rules()  
 {  
 **return** [  
 [['avatar'], 'string'],  
 [['imageFile'], 'image', 'extensions' => ['png', 'jpg']],  
 [['user\_id', 'gallery\_id'], 'integer'],  
 [['details'], 'string'],  
 [['first\_name', 'last\_name', 'birth\_date'], 'string', 'max' => 32],  
 [['first\_name', 'last\_name', 'birth\_date', 'details'], 'default', 'value' => **null**],  
 [['gallery\_id'], 'exist', 'skipOnError' => **true**, 'targetClass' => Gallery::*className*(), 'targetAttribute' => ['gallery\_id' => 'id']],  
 [['user\_id'], 'exist', 'skipOnError' => **true**, 'targetClass' => User::*className*(), 'targetAttribute' => ['user\_id' => 'id']],  
 ];  
 }  
  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** attributeLabels()  
 {  
 **return** [  
 'id' => 'ID',  
 'first\_name' => 'First Name',  
 'last\_name' => 'Last Name',  
 'birth\_date' => 'Birthday',  
 'details' => 'Details',  
 'user\_id' => 'User ID',  
 'gallery\_id' => 'Gallery ID',  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getGallery()  
 {  
 **return** $this->hasOne(Gallery::*className*(), ['id' => 'gallery\_id']);  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getUser()  
 {  
 **return** $this->hasOne(User::*className*(), ['id' => 'user\_id']);  
 }  
  
 **public function** getName()  
 {  
 **return isset**($this->first\_name) ? $this->getFullName() : $this->getUsername();  
 }  
  
 */\*\*  
 \** ***@return*** *string  
 \*/* **public function** getFullName()  
 {  
 **return** $this->first\_name . ' ' . $this->last\_name;  
 }  
  
 */\*\*  
 \** ***@return*** *string  
 \*/* **public function** getUsername()  
 {  
 **return** $this->user->username;  
 }  
  
 **public function** beforeSave($insert)  
 {  
 **if** (**parent**::*beforeSave*($insert)) {  
 **if** ($imageFile = UploadedFile::*getInstance*($this, 'imageFile')) {  
 $this->avatar = '/uploads/' . $imageFile->baseName . '.' . $imageFile->extension;  
 $imageFile->saveAs(substr($this->avatar, 1));  
 }  
 **return true**;  
 }  
 **return false**;  
 }  
  
 **public function** setDefaultImage()  
 {  
 $this->avatar = '/uploads/standart.png';  
 }  
}

models/ConversationMessage.php

**<?php  
  
namespace** app\models;  
  
**use** Yii;  
**use** yii\behaviors\TimestampBehavior;  
**use** yii\db\ActiveRecord;  
  
*/\*\*  
 \* This is the model class for table "conversation\_message".  
 \*  
 \** ***@property*** *string $id  
 \** ***@property*** *string $conversation\_id  
 \** ***@property*** *string $user\_id  
 \** ***@property*** *string $content  
 \** ***@property*** *string $created\_at  
 \*  
 \** ***@property*** *Conversation $conversation  
 \** ***@property*** *User $user  
 \*/***class** ConversationMessage **extends** ActiveRecord  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public static function** tableName()  
 {  
 **return** 'conversation\_message';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** rules()  
 {  
 **return** [  
 [['conversation\_id', 'user\_id'], 'integer'],  
 [['content'], 'string'],  
 [['created\_at'], 'string', 'max' => 32],  
 [['conversation\_id'], 'exist', 'skipOnError' => **true**, 'targetClass' => Conversation::*className*(), 'targetAttribute' => ['conversation\_id' => 'id']],  
 [['user\_id'], 'exist', 'skipOnError' => **true**, 'targetClass' => User::*className*(), 'targetAttribute' => ['user\_id' => 'id']],  
 ];  
 }  
  
 **public function** behaviors()  
 {  
 **return** [  
 'timestamp' => [  
 'class' => TimestampBehavior::*className*(),  
 'attributes' => [  
 ActiveRecord::*EVENT\_BEFORE\_INSERT* => ['created\_at'],  
 ],  
 'value' => date('Y-m-d H:i:s'),  
 ],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** attributeLabels()  
 {  
 **return** [  
 'id' => 'ID',  
 'conversation\_id' => 'Conversation ID',  
 'user\_id' => 'User ID',  
 'content' => 'Content',  
 'created\_at' => 'Created At',  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getConversation()  
 {  
 **return** $this->hasOne(Conversation::*className*(), ['id' => 'conversation\_id']);  
 }  
  
 **public function** getAuthor()  
 {  
 **return** $this->user->getUsername();  
 }  
  
 */\*\*  
 \** ***@param*** *User $user  
 \** ***@return*** *bool  
 \*/* **public function** isOwner($user)  
 {  
 **if** ($user === $this->getUser()) {  
 **return true**;  
 }  
 **if** ($user === **null**) {  
 **return false**;  
 }  
 **if** ($this->getUser() === **null**) {  
 **return false**;  
 }  
 **return** $user->getId() === $this->user->getId();  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getUser()  
 {  
 **return** $this->hasOne(User::*className*(), ['id' => 'user\_id']);  
 }  
}

models/Friendship.php

**<?php  
  
namespace** app\models;  
  
**use** Yii;  
  
*/\*\*  
 \* This is the model class for table "friendship".  
 \*  
 \** ***@property*** *int $id  
 \** ***@property*** *string $first\_user\_id  
 \** ***@property*** *string $second\_user\_id  
 \*  
 \** ***@property*** *User $firstUser  
 \** ***@property*** *User $secondUser  
 \*/***class** Friendship **extends** \yii\db\ActiveRecord  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public static function** tableName()  
 {  
 **return** 'friendship';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** rules()  
 {  
 **return** [  
 [['first\_user\_id', 'second\_user\_id'], 'integer'],  
 [['first\_user\_id', 'second\_user\_id'], 'unique', 'targetAttribute' => ['first\_user\_id', 'second\_user\_id']],  
 [['first\_user\_id'], 'exist', 'skipOnError' => **true**, 'targetClass' => User::*className*(), 'targetAttribute' => ['first\_user\_id' => 'id']],  
 [['second\_user\_id'], 'exist', 'skipOnError' => **true**, 'targetClass' => User::*className*(), 'targetAttribute' => ['second\_user\_id' => 'id']],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** attributeLabels()  
 {  
 **return** [  
 'id' => 'ID',  
 'first\_user\_id' => 'First User ID',  
 'second\_user\_id' => 'Second User ID',  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getFirstUser()  
 {  
 **return** $this->hasOne(User::*className*(), ['id' => 'first\_user\_id']);  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getSecondUser()  
 {  
 **return** $this->hasOne(User::*className*(), ['id' => 'second\_user\_id']);  
 }  
}

Зарефакторимо розположення(layout) і добавимо власне. В них ми добавимо посилання на CRUD для постів і користувачів, відповідно для редакторів і адміністраторів.

views/layout/main.php

**<?php***/\** ***@var*** *$this \yii\web\View \*/  
  
/\** ***@var*** *$content string \*/***use** app\widgets\Alert;  
**use** yii\helpers\Html;  
**use** yii\bootstrap\Nav;  
**use** yii\bootstrap\NavBar;  
**use** yii\widgets\Breadcrumbs;  
**use** app\assets\AppAsset;  
  
AppAsset::*register*($this);  
**?>  
<?php** $this->beginPage() **?>**<!DOCTYPE html>  
<html lang="**<?=** Yii::*$app*->language **?>**">  
<head>  
 <meta charset="**<?=** Yii::*$app*->charset **?>**">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 **<?=** Html::*csrfMetaTags*() **?>** <title>**<?=** Html::*encode*($this->title) **?>**</title>  
 **<?php** $this->head() **?>**</head>  
<body>  
**<?php** $this->beginBody() **?>**<div class="wrap">  
 **<?php** NavBar::*begin*([  
 'brandLabel' => Yii::*$app*->name,  
 'brandUrl' => Yii::*$app*->homeUrl,  
 'options' => [  
 'class' => 'navbar-inverse navbar-fixed-top',  
 ],  
 ]);  
 **echo** Nav::*widget*([  
 'options' => ['class' => 'navbar-nav navbar-right'],  
 'items' => [  
 Yii::*$app*->user->isGuest ? '' : ['label' => 'Followers', 'url' => ['/user/followers', 'id' => Yii::*$app*->user->identity->getId()]],  
 ['label' => 'Conversations', 'url' => ['/conversation/display']],  
 Yii::*$app*->user->can('admin') || Yii::*$app*->user->can('moderator') ?  
 [  
 'label' => 'Administration',  
 'items' => [  
 ['label' => 'Conversations', 'url' => ['/conversation/index']],  
 ['label' => 'Users', 'url' => ['/user/index']],  
 ]  
 ] : '',  
 Yii::*$app*->user->isGuest ? ['label' => 'Sign Up', 'url' => ['/site/signup']] : '',  
 Yii::*$app*->user->isGuest ? '' : ['label' => Yii::*$app*->user->identity->profile->getName(), 'url' => ['/user/display', 'id' => Yii::*$app*->user->identity->getId()]],  
 Yii::*$app*->user->isGuest ? (['label' => 'Login', 'url' => ['/site/login']]) : (  
 '<li>'  
 . Html::*beginForm*(['/site/logout'], 'post')  
 . Html::*submitButton*(  
 'Logout (' . Yii::*$app*->user->identity->profile->getName() . ')',  
 ['class' => 'btn btn-link logout']  
 )  
 . Html::*endForm*()  
 . '</li>'  
 )  
 ],  
 ]);  
 NavBar::*end*();  
 **?>** <div class="container">  
 **<?=** Breadcrumbs::*widget*([  
 'links' => **isset**($this->params['breadcrumbs']) ? $this->params['breadcrumbs'] : [],  
 ]) **?>  
 <?=** Alert::*widget*() **?>  
 <?=** $content **?>** </div>  
</div>  
  
<footer class="footer">  
 <div class="container">  
 <p class="pull-left">&copy; My Company **<?=** date('Y') **?>**</p>  
  
 <p class="pull-right">**<?=** Yii::*powered*() **?>**</p>  
 </div>  
</footer>  
  
**<?php** $this->endBody() **?>**</body>  
</html>  
**<?php** $this->endPage() **?>**

Додаткові файли

site.css

html,  
body {  
 height: 100%;  
}  
  
.wrap {  
 min-height: 100%;  
 height: auto;  
 margin: 0 auto -60px;  
 padding: 0 0 60px;  
}  
  
.wrap > .container {  
 padding: 70px 15px 20px;  
}  
  
.footer {  
 height: 60px;  
 background-color: #f5f5f5;  
 border-top: 1px solid #ddd;  
 padding-top: 20px;  
}  
  
.jumbotron {  
 text-align: center;  
 background-color: transparent;  
}  
  
.jumbotron .btn {  
 font-size: 21px;  
 padding: 14px 24px;  
}  
  
.not-set {  
 color: #c55;  
 font-style: italic;  
}  
  
/\* add sorting icons to gridview sort links \*/  
a.asc:after, a.desc:after {  
 position: relative;  
 top: 1px;  
 display: inline-block;  
 font-family: 'Glyphicons Halflings';  
 font-style: normal;  
 font-weight: normal;  
 line-height: 1;  
 padding-left: 5px;  
}  
  
a.asc:after {  
 content: /\*"\e113"\*/ "\e151";  
}  
  
a.desc:after {  
 content: /\*"\e114"\*/ "\e152";  
}  
  
.sort-numerical a.asc:after {  
 content: "\e153";  
}  
  
.sort-numerical a.desc:after {  
 content: "\e154";  
}  
  
.sort-ordinal a.asc:after {  
 content: "\e155";  
}  
  
.sort-ordinal a.desc:after {  
 content: "\e156";  
}  
  
.grid-view th {  
 white-space: nowrap;  
}  
  
.hint-block {  
 display: block;  
 margin-top: 5px;  
 color: #999;  
}  
  
.error-summary {  
 color: #a94442;  
 background: #fdf7f7;  
 border-left: 3px solid #eed3d7;  
 padding: 10px 20px;  
 margin: 0 0 15px 0;  
}  
  
/\* align the logout "link" (button in form) of the navbar \*/  
.nav li > form > button.logout {  
 padding: 15px;  
 border: none;  
}  
  
@media (max-width: 767px) {  
 .nav li > form > button.logout {  
 display: block;  
 text-align: left;  
 width: 100%;  
 padding: 10px 15px;  
 }  
}  
  
.nav > li > form > button.logout:focus,  
.nav > li > form > button.logout:hover {  
 text-decoration: none;  
}  
  
.nav > li > form > button.logout:focus {  
 outline: none;  
}  
  
.dialog-messages {  
 padding: 10px;  
 max-height: 71vh;  
 min-height: 71vh;  
 transition: all 0.5s ease;  
 -webkit-box-shadow: inset 0 1px 1px rgba(0, 0, 0, .05);  
 box-shadow: inset 0 1px 1px rgba(0, 0, 0, .05);  
}  
  
.message {  
 margin-bottom: 10px;  
 z-index: -1;  
}  
  
.toggled {  
 display: block **!important**;  
}  
  
#page-content-wrapper,  
#sidebar-wrapper {  
 padding: 0;  
}  
  
.profile-view .pagination {  
 margin: 0;  
 padding: 0;  
}  
  
.profile-view .list-view .container {  
 padding: 0;  
}  
  
#user-avatar,  
.follower-avatar {  
 width: 100%;  
 object-fit: cover;  
}  
  
.follower-avatar {  
 height: 200px **!important**;  
}  
  
#user-avatar {  
 height: 293px;  
}  
  
.dialog-messages::-webkit-scrollbar,  
.user-conversations::-webkit-scrollbar {  
 width: 0.4em;  
 height: 0.4em;  
}  
  
.dialog-messages::-webkit-scrollbar-thumb,  
.user-conversations::-webkit-scrollbar-thumb {  
 background-color: rgba(0, 0, 0, 0.2);  
}  
  
/\*left column in page display conversations\*/  
.user-conversations {  
 max-height: 73vh;  
 min-height: 73vh;  
}  
  
.dialog-messages,  
.user-conversations {  
 overflow-y: auto;  
}  
  
.conversation-display .list-group-item {  
 height: 60px;  
 margin-bottom: 0;  
 border: none;  
 border-bottom: 1px solid #f7f7f7;  
 border-radius: unset **!important**;  
}  
  
.header {  
 padding: 5px 15px;  
 background-color: #eee;  
 height: 60px;  
}  
  
.header .buttons {  
 padding-top: 8px;  
}  
  
.side {  
 margin: 0;  
 padding: 0;  
 height: 100%;  
}  
  
.reply {  
 height: 100%;  
 padding: 10px 15px;  
 background-color: #eee;  
}  
  
.reply .side {  
 height: 35px;  
}  
  
.reply button {  
 width: 100%;  
}  
  
.reply textarea {  
 width: 99%;  
 height: 35px;  
}  
  
.conversation-users,  
.conversations {  
 border: 1px solid rgba(0, 0, 0, .08);  
}  
  
.conversation {  
 border-right: 1px solid rgba(0, 0, 0, .08);  
}  
  
.conversation-users .header,  
.conversations .header {  
 font-size: 22px;  
 padding-top: 17px;  
}  
  
.conversation-users .list-group {  
 margin-bottom: 15px;  
 min-height: 75vh;  
}  
  
#new-conversation {  
 margin-bottom: 0;  
}  
  
#new-conversation .btn {  
 -webkit-border-radius: unset;  
 -moz-border-radius: unset;  
 border-radius: unset;  
}

main.js

$.fn.scrollBottom = **function**(scroll){  
 **if**(**typeof** scroll === 'number'){  
 window.scrollTo(0,$(document).height() - $(window).height() - scroll);  
 **return** $(document).height() - $(window).height() - scroll;  
 } **else** {  
 **return** $(document).height() - $(window).height() - $(window).scrollTop();  
 }  
};  
  
**function** updateChat() {  
 **let** lastId = $('.dialog-messages div[data-key]:last').data('key');  
 $.ajax({  
 url: window.actionUrl.addUrlParam('lastId', lastId)  
 }).done(**function** (isChanged) {  
 **if** (isChanged) {  
 $.pjax.reload({container: '#messages', async:**false**});  
 $.pjax.reload({container: '#conversations', async:**false**});  
 }  
 });  
}  
  
  
jQuery(**function** () {  
 setInterval(updateChat, 3000);  
  
 **var** messages = $('#messages');  
  
 **function** scrollDown() {  
 **var** list = messages.parents('.dialog-messages');  
 list.scrollTop(list[0].scrollHeight);  
 }  
 scrollDown();  
  
 messages.on('pjax:end', scrollDown);  
 $('#new\_message').on('pjax:end', **function** () {  
 $.pjax.reload({container: '#messages', async:**false**});  
 $.pjax.reload({container: '#conversations', async:**false**});  
 });  
  
  
 $('#send\_btn').on('click', **function**(){  
 **var** btn = $(**this**);  
 **var** i = btn.find('i');  
 i.addClass('glyphicon-time').removeClass('glyphicon-send');  
 setTimeout(**function** () {  
 btn.prop('disabled', **true**);  
 });  
 })  
});

Результат

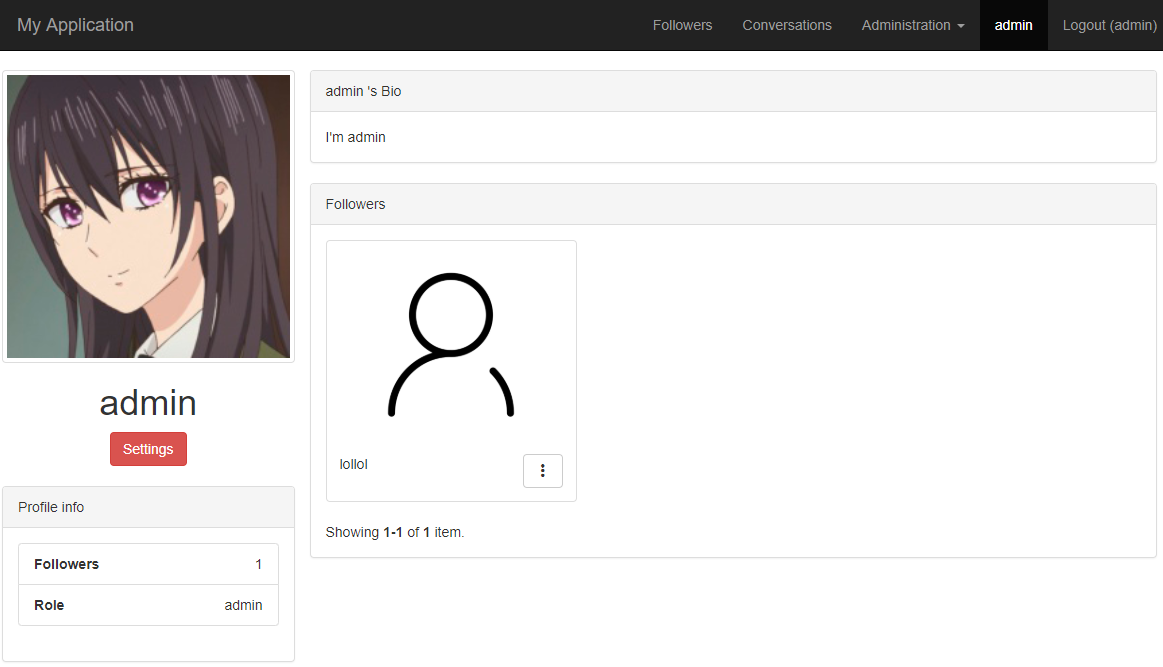


Рисунок 4 - Профіль користувача



Рисунок 5 - Доступні сторінки користувачу із роллю "user"



Рисунок 6 - Доступні сторінки користувачу із роллю "admin" чи "moderator"

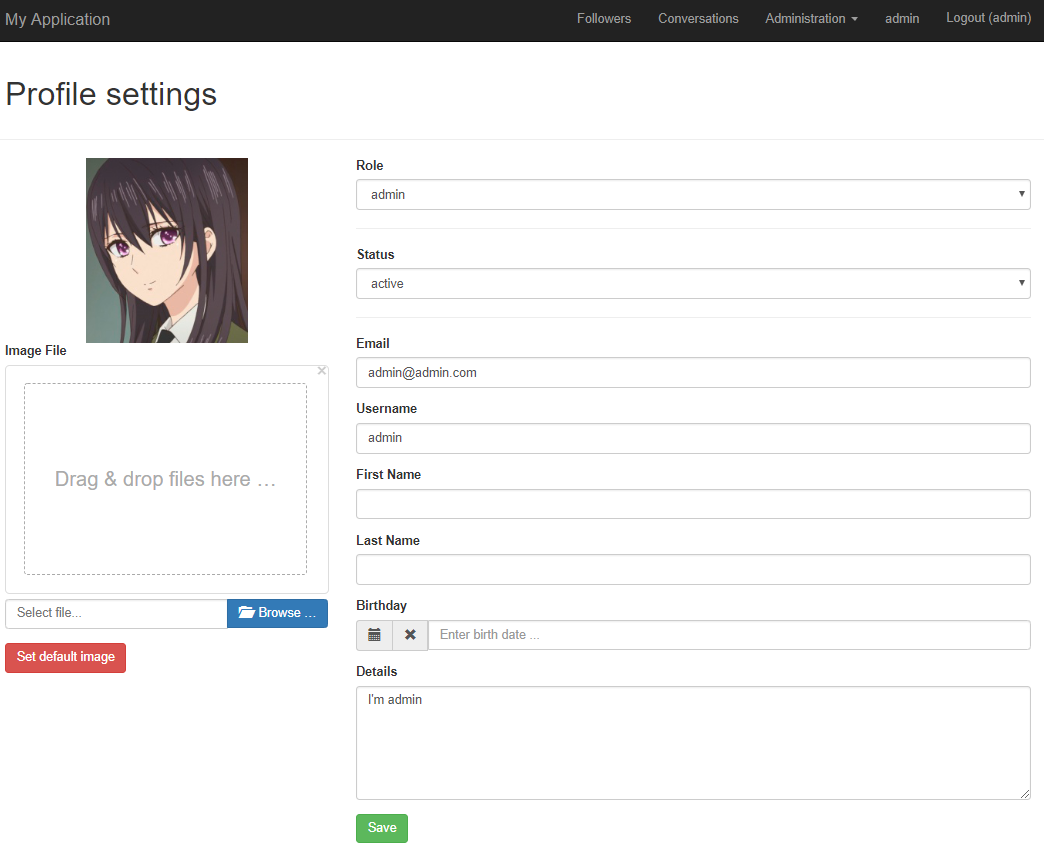


Рисунок 7 - Редагування користувача

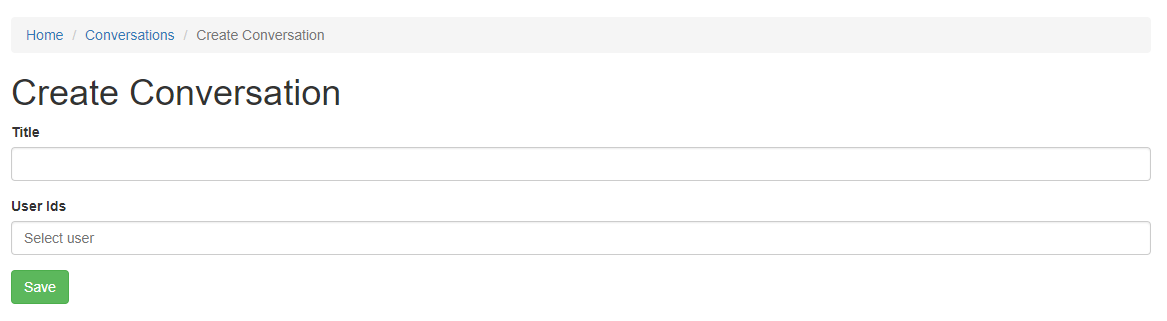


Рисунок 8 - Створення бесіди

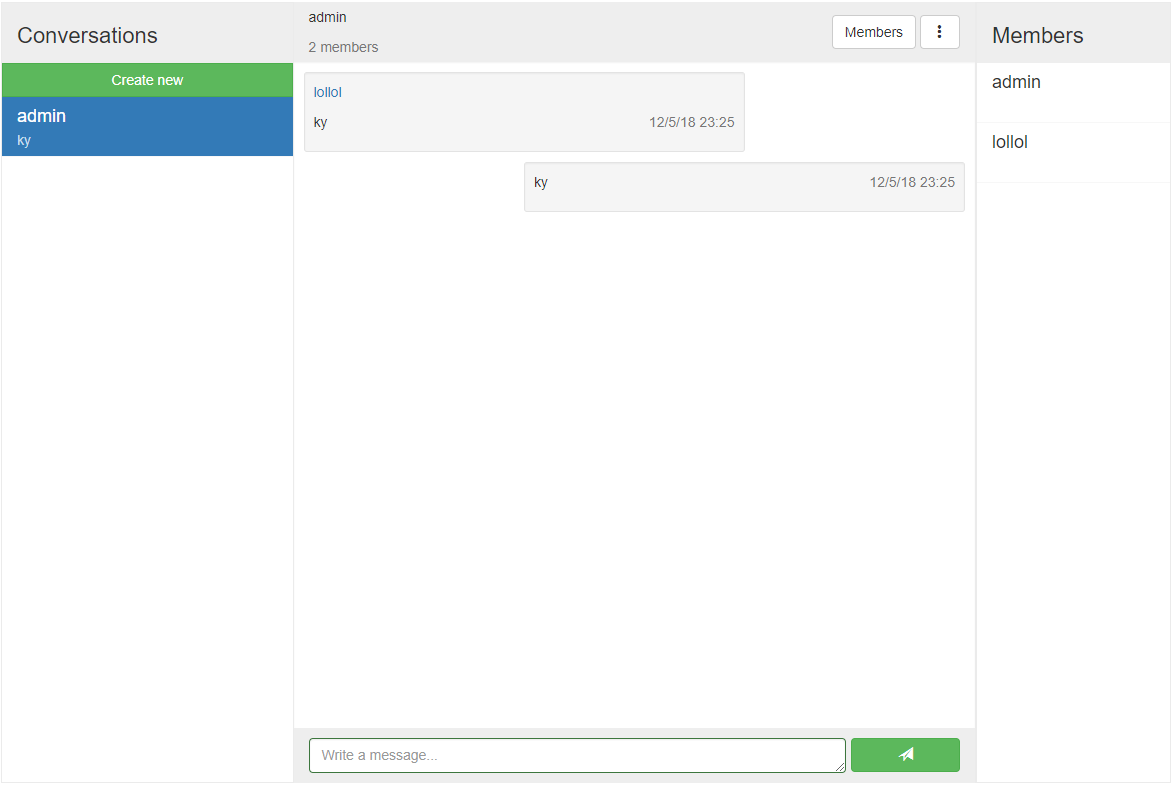


Рисунок 9 - Вигляд чату

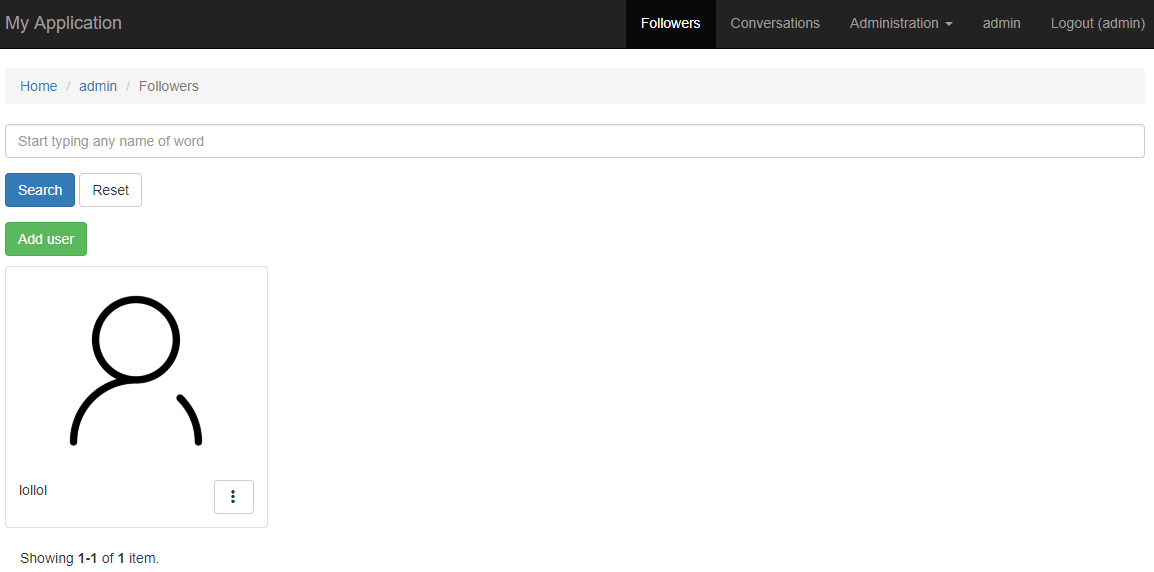


Рисунок 10 - Друзі користувача

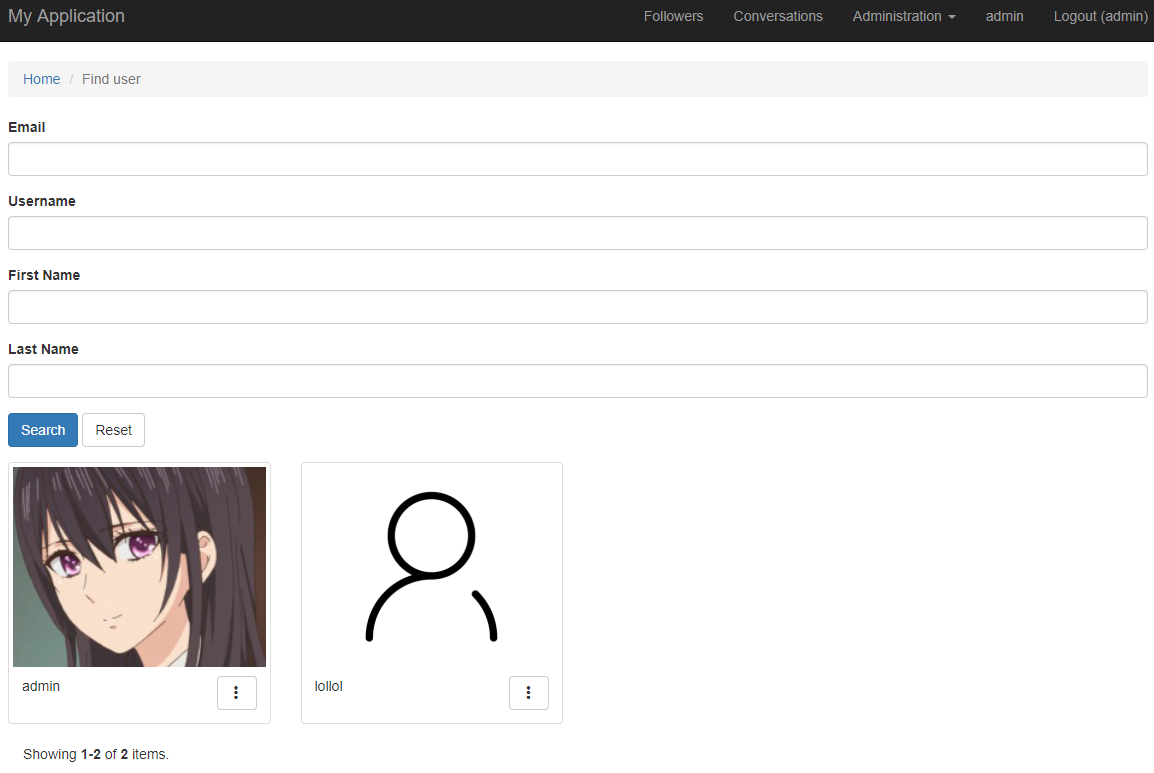


Рисунок 11 - Пошук користувача

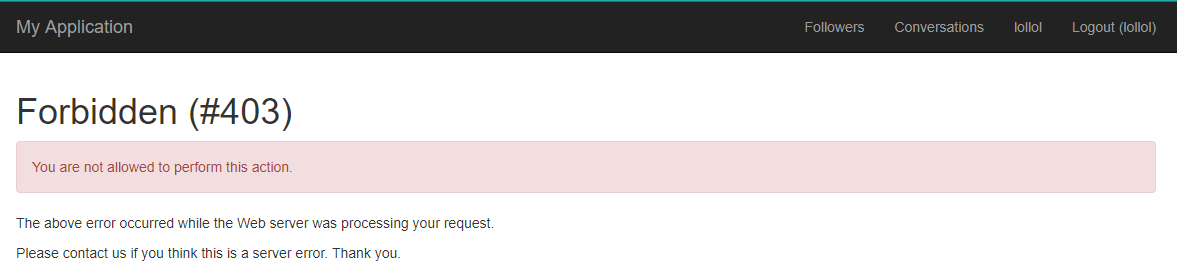


Рисунок 12 - Повідомлення про недостатність прав

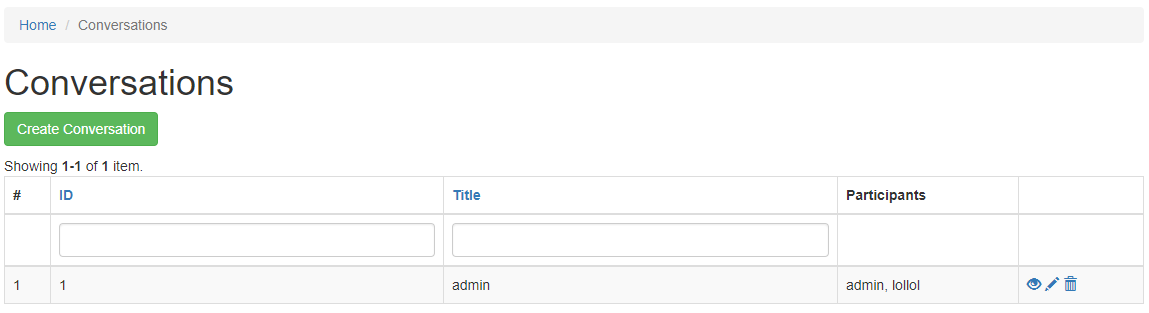


Рисунок 13 - CRUD бесід

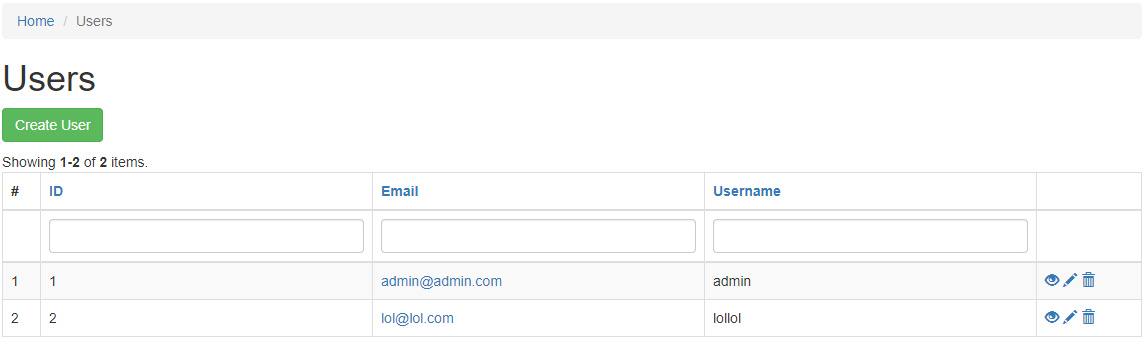


Рисунок 14 – CRUD користувачів

Використані джерела:

* <https://www.yiiframework.com/doc/guide/2.0/en> | Документація Yii2
* <https://github.com/kartik-v/yii2-widget-datepicker> | Віджит для вибору дати
* <https://github.com/kartik-v/yii2-widget-fileinput> | Віджет для завантаження файлів

Висновок.

На даній лабораторній роботі я спробував створити власну соціальну мережу на основі Yii2. Результат <https://github.com/Weismann9/yii2-social-network>