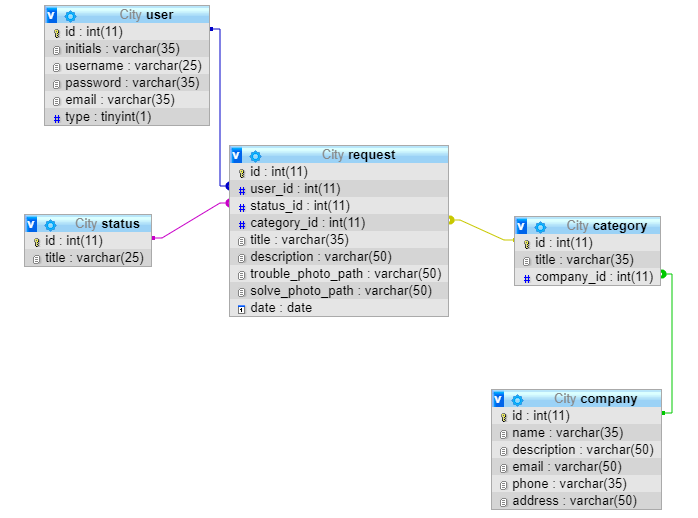
1. Создаем бдшку:
   1. user
      * 1. -id
        2. -initials
        3. -username
        4. -password
        5. -email
        6. -type
   2. request
      * 1. -id
        2. -user\_id
        3. -status\_id
        4. -category\_id
        5. -title
        6. -description
        7. -trouble\_photo\_path
        8. -solve\_photo\_path
        9. -date
   3. status
      * 1. -id
        2. -title
   4. category
      * 1. -id
        2. -title
        3. -company\_id
   5. company
      * 1. -id
        2. -name
        3. -description
        4. -email
        5. -phone
        6. -address
2. Создаем связи в бдшке:
   1. Заходим в таблицу request, вкладка «Структура», выбираем user\_id и category\_id и нажимаем на кнопку с молнией «Индекс»;
   2. Жмем на бдшку и открываем «Дизайнер», жмем кнопку «Создать ссылку», соединяем requests ОТ user, category;
   3. Выбираем CASCADE;
   4. 
3. Заходим в таблицу users, вкладка «Вставить» и делаем юзера admin с паролем adminWSR;
4. Распаковываем фреймворк в папку;
5. В config/web.php меняем 'cookieValidationKey’ на рандомный;
6. В config/db.php меняем данные для подключения к БД;
7. Делаем ЧПУ:
   1. В config/web.php убираем комменты urlManager’a;
   2. Там же, под ключом, ставим ‘baseUrl’=>’’;
   3. Создаем web/.htaccess и вставляем:
   4. RewriteEngine On RewriteBase /  
        
      RewriteCond %{REQUEST\_FILENAME} !-f  
      RewriteCond %{REQUEST\_FILENAME} !-d  
        
      RewriteRule . index.php
   5. Создаем в корне сайте .htaccess и вставляем:
   6. Options +FollowSymLinks  
      IndexIgnore \*/\*  
      RewriteEngine on  
        
      RewriteCond %{REQUEST\_URI} !^/(web)  
      RewriteRule ^assets/(.\*)$ /web/assets/$1 [L]  
      RewriteRule ^css/(.\*)$ web/css/$1 [L]  
      RewriteRule ^js/(.\*)$ web/js/$1 [L]  
      RewriteRule ^images/(.\*)$ web/images/$1 [L]  
      RewriteRule (.\*) /web/$1  
        
      RewriteCond %{REQUEST\_FILENAME} !-f  
      RewriteCond %{REQUEST\_FILENAME} !-d  
      RewriteRule . /web/index.php
8. Устанавливаем шаблон:
   1. Копируем views/layouts/main.php и меняем название (e.g. basic.php);
   2. Убираем лишний html-код и вставляем из предоставленного шаблона хедер, футер и пр.(не забыть вставить <?= $content ?>);
   3. Вставляем в config/web.php в блок $config = [ после ‘bootstrap’ - ‘layout’ = > ‘basic’ ];
   4. Подключаем стили в assets/AppAsset.php в public $css = [‘css/style.css’];
   5. Подключаем скрипты в assets/AppAsset.php в public $js = [‘js/windows.js’];
   6. /////Комментим в public $depends = [//’yii\bootstrap\BootstrapAsset’];
   7. Копируем в web/css/ стили из шаблона и в web/js/ скрипты;
9. В файл views/site/index.php вставляем код основной страницы;
10. Создаем модуль админки (и юзера, также):
    1. Копируем файл models/User.php и называем UserDefault.php;
    2. Проходим по пути /web/index.php?r=gii, заходим в «Module Generator»;
    3. Пишем в «Module Class» - app\modules\admin\Module и в «Module ID» - admin;
    4. Копируем появившейся код в config/web.php после «UrlManager» за квадратными скобками;
    5. Заходим в «Model Generator», ставим «\*» и создаем модели;
    6. Заходим в «CRUD Generator» и пишем: в «Model Class» - app\models\Category, в «Search Model Class» - app\models\CategorySearch, в «Contoller Class» - app\modules\admin\controllers\CategoryController, в «View Path» - @app/modules/admin/views/Category;
    7. Генерим все и делаем тоже самое с Company и Request(только не для админа, а для юзера);
11. Делаем авторизацию:
    1. Прописываем в модели models/User.php в классе implements IdentityInterface и жмем Alt+Enter чтобы реализовать методы;
    2. Прописываем в findIdentity($id) - return User::findOne($id);
    3. В getId() - return $this->id;
    4. Добавляем функции - public function findByUsername($username) {  
        return User::find()->where(['username'=>$username])->one();  
       }
    5. public function validatePassword($password) {  
        return ($this->password == $password) ? true : false;  
       }
    6. public function create() {  
        return $this->save(false);  
       }
    7. Меняем в views/layouts/basic.php шапку - проставляем ссылки (/web/index.php?r=site/login), проверяем на гостя, выводим инициалы - Yii::$app->user->identity->initials, у логаута ставим тип данных data-method="post") проверяем на гостя, вставляем инициалы;
12. Делаем регистрацию:
    1. Копируем views/site/login.php и называем register.php;
    2. Меняем поля под поля из БДшки;
    3. Создаем models/RegisterForm.php и пишем там - namespace app\models;  
       use yii\base\model;  
       class RegisterForm extends Model {  
        public $initials;  
        public $username;  
        public $password;  
        public $email;  
        public function rules() {  
        return [  
       [['initials','username','password','passwordCopy','email','rememberMe'], 'required', 'message'=>'Поле обязательно'],  
       [['initials','username'], 'string'],  
       ['initials', 'match', 'pattern'=>'/^[а-яА-ЯёЁ\s]+$/u', 'message'=>'Только кириллица и пробелы'],  
       [['username'], 'unique', 'targetClass' => 'app\models\User','targetAttribute'=>'username','message'=>'Логин уже занят'],  
       [['username'], 'match', 'pattern' => '/^[a-zA-Z]+$/', 'message' => 'Только латиница'],  
       [['email'], 'email','message'=>'Некорректный тип почты'],  
       [['email'], 'unique', 'targetClass' => 'app\models\User','targetAttribute'=>'email','message'=>'Почта уже занята'],  
       [['password'], 'match', 'pattern' => '/^(?=.\*[a-z])(?=.\*[A-Z])[a-zA-Z]{6,35}$/', 'message' => 'Только латиница, как минимум один символ в верхнем регистре, больше 5 символов'],  
       ['passwordCopy', 'compare', 'compareAttribute' => 'password','message'=>'Пароли не совпадают'],  
       ['rememberMe','compare','compareValue'=>true,'message'=>'Без согласия регистрация невозможна']
    4. ];  
        }  
        public function register() {  
        if ($this->validate()) {  
        $user = new User();  
        $user->attributes = $this->attributes;  
        return $user->create();  
        }  
        }  
       }
    5. В controllers/SiteController.php прописываем новый action под логином - public function actionRegister()  
       {  
        $model = new RegisterForm();  
        if (Yii::$app->request->isPost) {  
        $model->load(Yii::$app->request->post());  
        if ($model->register()) {  
        return $this->redirect(['site/login']);  
        }  
        }  
        return $this->render('register',['model'=>$model]);  
       }
13. Закрываем админку и юзерку:
    1. Заходим в файл modules/admin/Module.php и пишем там -
14. Кастомизируем гриды:
    1. Чтобы сделать красивый вывод табличек на мобилах, надо обернуть их в <div class="table-responsive">
    2. Кастомная колонка для удаления записи: [  
        'label' => 'Удалить?',  
        'format' => 'raw',  
        'value' => function($data){  
        if ($data->status\_id == 1) {  
        return Html::a(  
        'Удалить',  
        'delete?id='.$data->id,  
        [  
        'target' => '\_blank',  
        'data-method' => 'post',  
        'data-confirm'=>'Are you sure you want to delete this item?'  
        ]  
        );  
        } else {  
        return Html::encode('');  
        }  
        }  
       ],;
    3. Чтобы изменить кол-во выводимых записей (н.р. на 25) надо в searchController прописать - $dataProvider->pagination->pageSize = 25;
    4. Чтобы связать грид с другой таблицей и вставить вместо id-шника название компании н-р:
       * 1. Проверяем функцию public function getCompany()  
            {  
             return $this->hasOne(Company::className(), ['id' => 'company\_id']);  
            } в модели models/Category.php и меняем название на getIdCompany;
         2. В modules/admin/views/Category/index.php меняем колонку company\_id на [‘attribute’ => ‘company\_id’, ‘value’ =>’idCompany.name’,’filter’=>$arrCompany,];
         3. В modules/admin/controllers/CategoryController.php в actionIndex() и actionUpdate() и т.д. пишем - $model\_company = Company::find()->orderBy('name ASC')->all();  
            foreach ($model\_company as $value) {  
             $arrCompany[$value->id] = $value->name;  
            } и в ретёрне ставим 'arrCompany' => $arrCompany,;
    5. Чтобы выводить картинки с постом надо:
       * 1. В modules/user/views/Request/index.php вместо trouble\_photo\_path прописать - [  
             'attribute' => 'trouble\_photo\_path',  
             'label' => 'Фото проблемы',  
             'format' => 'html',  
             'value' => function ($data) {  
             return Html::img('/web/photos/t'.$data->id.'.png', ['style' => 'width:100px;']);  
             },  
            ],;
    6. Чтобы выводить заявки только этого юзера надо:
       * 1. В модели models/RequestSearch.php после объявления переменной $query написать следующее - $query->where('user\_id='.\Yii::$app->user->identity->id);
    7. Чтобы сделать дропдаун в создании категории, надо:
       * 1. В modules/admin/controllers/CategoryController.php в actionCreate() и actionUpdate() вставить то же, что и в actionIndex(пункт сверху);
         2. В modules/admin/views/Category/\_form.php меняем текстинпут на dropDownList($arrCompany,[‘prompt’=>’’];)
         3. В modules/admin/views/Category/create.php и update.php в рендере после модели ставим ‘arrCompany’ => $arrCompany;
    8. Чтобы выводить связанные данные в view элемента, то надо в виджете DetailView прописать вместо idCompany - idCompany.name;
    9. Настраиваем создание заявки:
       * 1. Делаем загрузку изображения:
         2. В форме modules/user/views/Request/\_form.php меняем тип поля загрузки изображения на fileInput;
         3. В модели models/Request.php вставляем функцию public function uploadFile(UploadedFile $file,$id) {  
             $file->saveAs('photos/t'.$id.'.png');  
            }
         4. В modules/user/controllers/RequestController.php в actionCreate() добавляем в условии if ($model->load(Yii::$app->request->post()) && $model->save()) {  
             $file = UploadedFile::getInstance($model, 'trouble\_photo\_path');  
             $model->uploadFile($file,$model->id);
         5. В форме modules/user/views/Request/\_form.php проверяем юзера на админа, если не админ, то выключаем поля поле статуса заявки, поле фотографии решения проблемы и т.д.;
         6. Заполняем стандартными данными перед принтом полей если идет создание заявки: <?php  
            if ($isCreate) {  
             $now = new DateTime();  
             $result = $now->format('Y-m-d H:i:s');  
              
             $model->user\_id = Yii::$app->user->identity->id;  
             $model->status\_id = 1;  
             if ($isCreate)  
             $model->solve\_photo\_path = '-';  
             $model->date = $result;  
            }  
            ?>
         7. В файле modules/user/views/Request/create.php после рендера и передачи массивов arrStatus e.t.c передаем переменную ‘isCreate’=>true;
         8. В модели models/Request.php добавляем проверку на админа в правилах валидации $arr = [  
             [['user\_id', 'status\_id', 'category\_id', 'title', 'description', 'date'], 'required'],  
             [['user\_id', 'status\_id', 'category\_id'], 'integer'],  
             [['date'], 'safe'],  
             [['title'], 'string', 'max' => 35],  
             [['description', 'trouble\_photo\_path', 'solve\_photo\_path'], 'string', 'max' => 50],  
             [['user\_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::className(), 'targetAttribute' => ['user\_id' => 'id']],  
             [['category\_id'], 'exist', 'skipOnError' => true, 'targetClass' => Category::className(), 'targetAttribute' => ['category\_id' => 'id']],  
             [['status\_id'], 'exist', 'skipOnError' => true, 'targetClass' => Status::className(), 'targetAttribute' => ['status\_id' => 'id']],  
            ];  
            if (Yii::$app->user->identity->type == 1) {  
             $res = [[['solve\_photo\_path'], 'file', 'extensions' => 'png, jpg', 'maxFiles' => 1,'maxSize' => 2\*1024\*1024,'minSize'=>10],];  
            } else {  
             $res = [  
             [['trouble\_photo\_path'], 'file', 'extensions' => 'png, jpg', 'maxFiles' => 1,'maxSize' => 2\*1024\*1024,'minSize'=>10],  
             ['user\_id','compare','compareValue'=> Yii::$app->user->identity->id,'message'=>'Вы пытаетесь добавить заявку не от своего имени!'],  
             ['status\_id','compare','compareValue'=> 1,'message'=>'Вы пытаетесь изменить статус заявки! (Вам нельзя)'],];  
            }  
            return array\_merge($arr,$res);
15. Делаем вывод решенных заявок на главной странице:
    1. В модели models/Request.php прописываем функцию public function getCompletedRequest() {  
        $completed = Yii::$app->db->createCommand("SELECT id,title,description FROM request WHERE status\_id = 3")->queryAll();  
        return $completed;  
       }
    2. В controllers/SiteController.php в actionIndex() пишем $model = new Request();  
       $completed = $model->getCompletedRequest();  
         
       return $this->render('index',  
        [  
        'completed'=>$completed  
        ]);
    3. В views/index.php пишем цикл внутри row после «Последние решенные проблемы» <?php foreach ($completed as $value) {  
        echo'<div class="col-sm-6 col-md-3">  
        <div class="thumbnail">  
        <img src=web/photos/t'.$value['id'].'.png alt="Яма на дороге">  
        <img src=web/photos/s'.$value['id'].'.png alt="Яма на дороге">  
        <p>'.$value['title'].'</p>  
        </div>  
       </div>';  
       } ?>