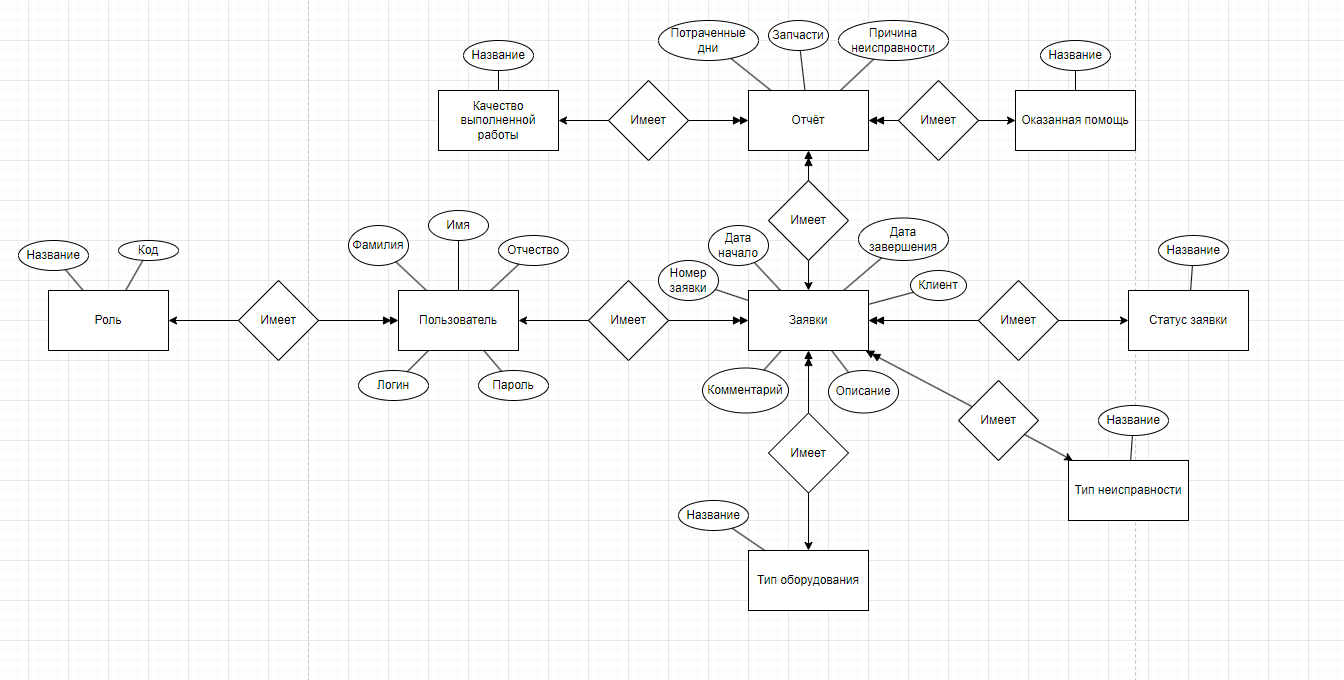
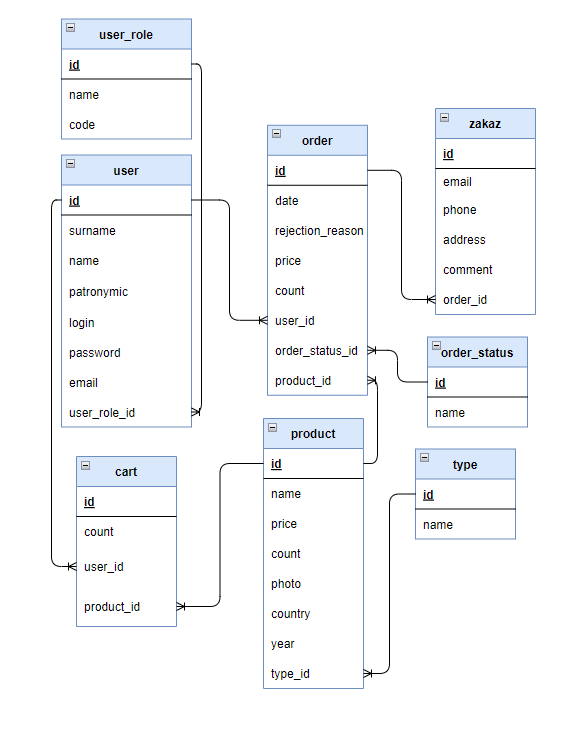
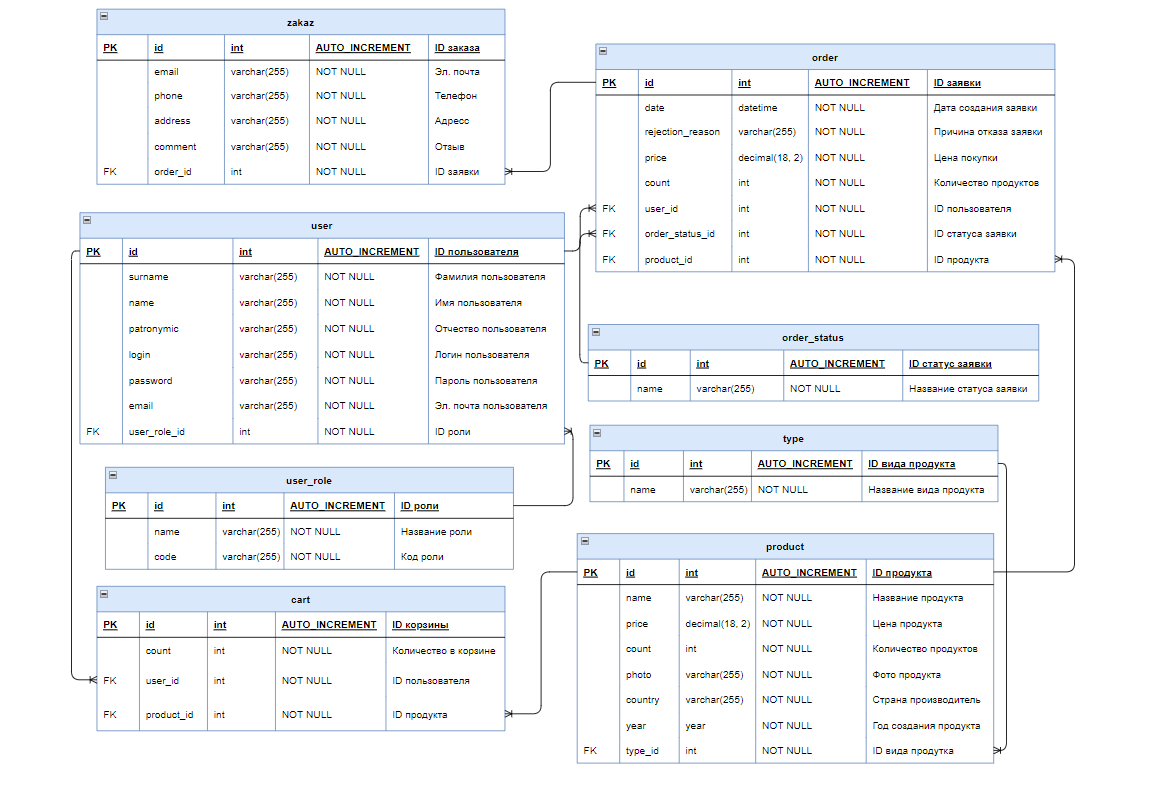
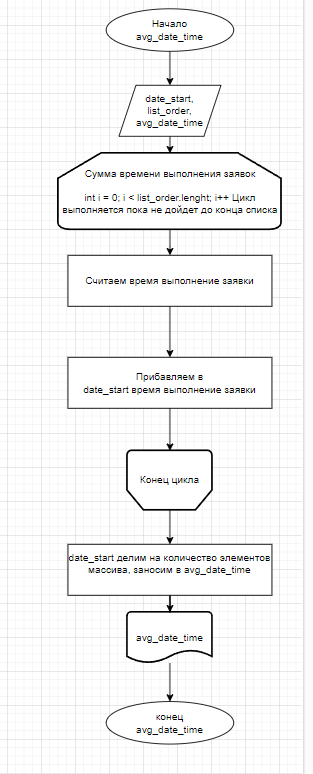
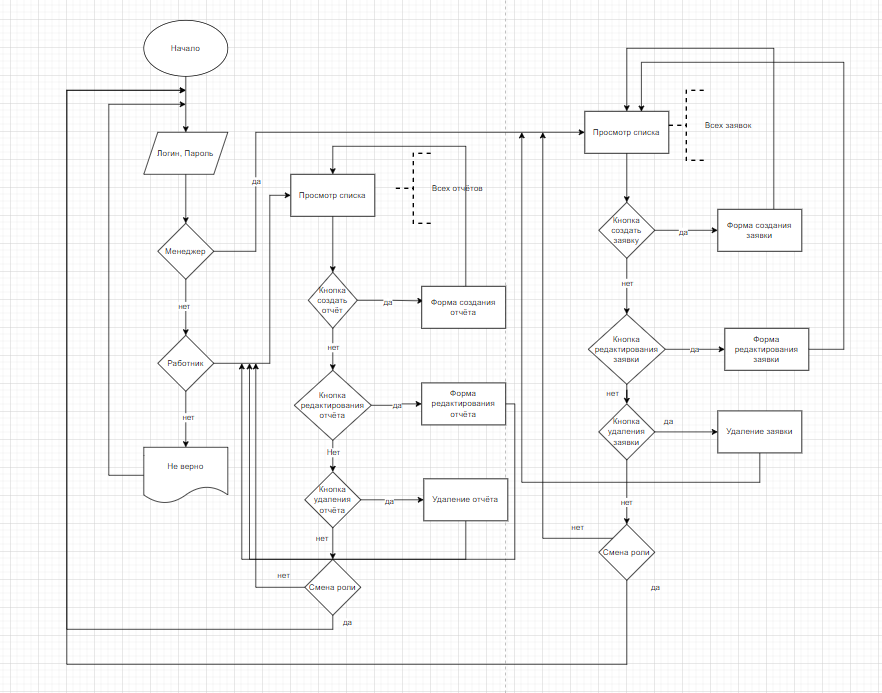
Диаграммы











# **ВНЕШНЯЯ СПЕЦИФИКАЦИЯ**

Таблица 1 – Внешняя спецификация метода нахождения среднего времени выполнения заявки

|  |  |  |
| --- | --- | --- |
| **Имя** | **Назначение** | **Тип/Диапазон** |
| **Входные величины** | | |
| date\_start | Дата начало заявки | Дата/время |
| list\_order | Список всех выполненных заявок | Строка |
| date\_end | Дата выполнения заявки | Дата/время |
| **Выходные величины** | | |
| avg\_date\_time | Среднее время выполнения заявки | Дата/время |

Код

RewriteEngine on  
RewriteRule (.\*)? /web/$1

Код файла web

<?php  
  
$params = require *\_\_DIR\_\_* . '/params.php';  
$db = require *\_\_DIR\_\_* . '/db.php';  
  
$config = [  
 'id' => 'basic',  
 'basePath' => dirname(*\_\_DIR\_\_*),  
 'bootstrap' => ['log'],  
 'aliases' => [  
 '@bower' => '@vendor/bower-asset',  
 '@npm' => '@vendor/npm-asset',  
 ],  
 'language' => 'ru-RU',  
 'components' => [  
 'request' => [  
 // !!! insert a secret key in the following (if it is empty) - this is required by cookie validation  
 'cookieValidationKey' => 'gdfgdfgdfg',  
 'baseUrl' => '',  
 ],  
 'cache' => [  
 'class' => 'yii\caching\FileCache',  
 ],  
 'user' => [  
 'identityClass' => 'app\models\User',  
 'enableAutoLogin' => true,  
 ],  
 'errorHandler' => [  
 'errorAction' => 'site/error',  
 ],  
 'mailer' => [  
 'class' => \yii\symfonymailer\Mailer::class,  
 'viewPath' => '@app/mail',  
 // send all mails to a file by default.  
 'useFileTransport' => true,  
 ],  
 'log' => [  
 'traceLevel' => *YII\_DEBUG* ? 3 : 0,  
 'targets' => [  
 [  
 'class' => 'yii\log\FileTarget',  
 'levels' => ['error', 'warning'],  
 ],  
 ],  
 ],  
 'db' => $db,  
  
 'urlManager' => [  
 'enablePrettyUrl' => true,  
 'showScriptName' => false,  
 'rules' => [  
 ],  
 ],  
  
 ],  
 'params' => $params,  
];  
  
if (*YII\_ENV\_DEV*) {  
 // configuration adjustments for 'dev' environment  
 $config['bootstrap'][] = 'debug';  
 $config['modules']['debug'] = [  
 'class' => 'yii\debug\Module',  
 // uncomment the following to add your IP if you are not connecting from localhost.  
 //'allowedIPs' => ['127.0.0.1', '::1'],  
 ];  
  
 $config['bootstrap'][] = 'gii';  
 $config['modules']['gii'] = [  
 'class' => 'yii\gii\Module',  
 // uncomment the following to add your IP if you are not connecting from localhost.  
 'allowedIPs' => ['\*', '::1'],  
 ];  
}  
  
return $config;

Пользователь

public static function findIdentity($id)  
{  
 return static::*findOne*($id);  
}  
  
*/\*\*  
 \* {****@inheritdoc****}  
 \*/*public static function findIdentityByAccessToken($token, $type = null)  
{  
 return null;  
}  
  
*/\*\*  
 \* Finds user by username  
 \*  
 \** ***@param*** *string $username  
 \** ***@return*** *static|null  
 \*/*public static function findByUsername($login)  
{  
 return User::*findOne*(['login'=>$login]);  
}  
  
*/\*\*  
 \* {****@inheritdoc****}  
 \*/*public function getId()  
{  
 return $this->id;  
}  
  
*/\*\*  
 \* {****@inheritdoc****}  
 \*/*public function getAuthKey()  
{  
 return null;  
}  
  
*/\*\*  
 \* {****@inheritdoc****}  
 \*/*public function validateAuthKey($authKey)  
{  
 return false;  
}  
  
*/\*\*  
 \* Validates password  
 \*  
 \** ***@param*** *string $password password to validate  
 \** ***@return*** *bool if password provided is valid for current user  
 \*/*public function validatePassword($password)  
{  
 return $this->password === $password;  
}  
  
public function isAdmin()  
{  
 return $this->userRole->code === "admin";  
}

Валидация пользователя

public function rules()  
{  
 return [  
 [['surname', 'name', 'login', 'password', 'email'], 'required'],  
 [['user\_role\_id'], 'integer'],  
 [['surname', 'name', 'patronymic', 'login', 'password', 'email'], 'string', 'max' => 255],  
 [['login'], 'unique'],  
 [['email'], 'unique'],  
 [['user\_role\_id'], 'exist', 'skipOnError' => true, 'targetClass' => UserRole::class, 'targetAttribute' => ['user\_role\_id' => 'id']],  
 [['surname', 'name', 'patronymic'], 'match', 'pattern' => '/^[а-яА-ЯёЁ -]+$/u', 'message' => 'Разрешены русские буквы, пробел и тире'],  
 [['login'], 'match', 'pattern' => '/^[a-zA-Z0-9-]+$/i', 'message' => 'Разрешены латинские буквы, цифры и тире'],  
 [['email'], 'email'],  
 [['password\_repeat'], 'compare', 'compareAttribute' => 'password', 'message' => 'Пароли не совпадают'],  
 [['rule'], 'compare', 'compareValue' => 1, 'message' => 'Согласитесь с правилами регистрации на сайте'],  
 ];  
}

Видимость заявок у пользователей

public function behaviors()  
{  
 return array\_merge(  
 parent::*behaviors*(),  
 [  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ]  
 );  
}  
  
*/\*\*  
 \* Lists all Order models.  
 \*  
 \** ***@return*** *string  
 \*/*public function actionIndex()  
{  
 if (\Yii::*$app*->user->identity->isAdmin()) {  
 $dataProvider = new ActiveDataProvider([  
 'query' => Order::*find*(),  
 ]);  
 } else {  
 $dataProvider = new ActiveDataProvider([  
 'query' => Order::*find*()->where(['user\_id' => \Yii::*$app*->user->getId()]),  
 ]);  
 }  
  
 return $this->render('index', [  
 'dataProvider' => $dataProvider,  
 ]);  
}

Видимость отчётов у пользователей

public function behaviors()  
{  
 return array\_merge(  
 parent::*behaviors*(),  
 [  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ]  
 );  
}  
  
*/\*\*  
 \* Lists all Report models.  
 \*  
 \** ***@return*** *string  
 \*/*public function actionIndex()  
{  
 if (\Yii::*$app*->user->identity->isAdmin()) {  
 $dataProvider = new ActiveDataProvider([  
 'query' => Report::*find*(),  
 ]);  
 } else {  
 $userOrders = Order::*find*()->where(['user\_id' => \Yii::*$app*->user->getId()])->all();  
  
 // Получаем ID заказов текущего пользователя  
 $userOrderIds = [];  
 foreach ($userOrders as $order) {  
 $userOrderIds[] = $order->id;  
 }  
  
 // Получаем отчеты, относящиеся к заказам текущего пользователя  
 $dataProvider = new ActiveDataProvider([  
 'query' => Report::*find*()->where(['order\_id' => $userOrderIds]),  
 ]);  
 }  
 return $this->render('index', [  
 'dataProvider' => $dataProvider,  
 ]);  
 }

Форма заявок

<?php $form = ActiveForm::*begin*(); ?>  
  
<?php if(!Yii::*$app*->user->isGuest && Yii::*$app*->user->identity->isAdmin()): ?>  
<?= $form->field($model, 'number')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'date\_start')->textInput(['type' => 'date']) ?>  
  
<?= $form->field($model, 'date\_end')->textInput(['type' => 'date']) ?>  
  
<?= $form->field($model, 'client')->textInput(['maxlength' => true]) ?>  
<?php endif; ?>  
  
<?php if(!$model -> isNewRecord): ?>  
<?= $form->field($model, 'status\_id')->dropDownList(\yii\helpers\ArrayHelper::*map*(\app\models\Status::*find*()->all(), 'id', 'name'),  
 ['prompt' => '', 'value' => '1', 'onchange' => 'selectStatus(this)']) ?>  
  
<?= $form->field($model, 'comment')->textarea(['rows' => 6]) ?>  
<?php endif; ?>  
  
<?php if(!Yii::*$app*->user->isGuest && Yii::*$app*->user->identity->isAdmin()): ?>  
<?= $form->field($model, 'description')->textarea(['rows' => 6]) ?>  
  
<?= $form->field($model, 'user\_id')->dropDownList(\yii\helpers\ArrayHelper::*map*(\app\models\User::*find*()->where(['user\_role\_id' => 2])->all(), 'id', 'name')) ?>  
  
<?= $form->field($model, 'type\_equipment\_id')->dropDownList(\yii\helpers\ArrayHelper::*map*(\app\models\TypeEquipment::*find*()->all(), 'id', 'name')) ?>  
  
<?= $form->field($model, 'type\_problem\_id')->dropDownList(\yii\helpers\ArrayHelper::*map*(\app\models\TypeProblem::*find*()->all(), 'id', 'name')) ?>  
<?php endif; ?>

Форма отчётов

<?php $form = ActiveForm::*begin*(); ?>  
  
<?= $form->field($model, 'order\_id')->dropDownList(\yii\helpers\ArrayHelper::*map*(\app\models\Order::*find*()->where(['user\_id' => Yii::*$app*->user->id])->all(), 'id', 'number')) ?>  
  
<?= $form->field($model, 'days')->textInput() ?>  
  
<?= $form->field($model, 'detail')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'price')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'reason')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'help\_id')->dropDownList(\yii\helpers\ArrayHelper::*map*(\app\models\Help::*find*()->all(), 'id', 'name')) ?>  
  
<?= $form->field($model, 'quality\_id')->dropDownList(\yii\helpers\ArrayHelper::*map*(\app\models\Quality::*find*()->all(), 'id', 'name')) ?>  
  
<div class="form-group">  
 <?= Html::*submitButton*('Сохранить', ['class' => 'btn btn-success']) ?>  
</div>  
  
<?php ActiveForm::*end*(); ?>

Атрибуты

[  
 'attribute' => 'order\_id',  
 'value' => function ($model) {  
 return $model->order->number;  
 }  
],

Скрытие кнопки

'visibleButtons' => [  
 'delete' => function ($model, $key, $index) {  
 return Yii::*$app*->user->identity->isAdmin();  
 }  
]

<p>  
 <?php if(Yii::*$app*->user->identity->isAdmin()): ?>  
 <?= Html::*a*('Создать заявку', ['create'], ['class' => 'btn btn-success']) ?>  
 <?php endif; ?>  
</p>

Статистика

<div class="site-about">  
 <h1><?= Html::*encode*($this->title) ?></h1>  
 <h4>Количество выполненных заявок: <?= \app\models\Order::*find*()->where(['status\_id' => 3])->count() ?> </h4>  
 <h4>Среднее время выполнения заявок: <?= intval(\app\models\Report::*find*()->average('days')) ?> дней</h4>  
 <h4>Количество внешней поломки: <?= \app\models\Order::*find*()->where(['type\_problem\_id' => 1])->count() ?> </h4>  
 <h4>Количество внутренней поломки: <?= \app\models\Order::*find*()->where(['type\_problem\_id' => 2])->count() ?> </h4>  
 <h4>Количество общей поломки: <?= \app\models\Order::*find*()->where(['type\_problem\_id' => 3])->count() ?> </h4>  
</div>

Оповещение

function selectStatus (select) {  
 if (select.value == 1) {  
 alert('В ожидании');  
 }  
 if (select.value == 2) {  
 alert('В работе');  
 }  
 if (select.value == 3) {  
 alert('Выполнено');  
 }  
}