ПРИЛОЖЕНИЕ 3

ПРОГРАММНЫЙ МОДУЛЬ ДЛЯ ИНТЕГРАЦИИ СТОРОННИХ ПРИЛОЖЕНИЙ С СИСТЕМОЙ ОРОИКС

ТЕКСТ ПРОГРАММЫ

```php
<?php
namespace app\controllers;

use app\controllers\_base\OrioksController;
use app\kernel\Util;
use app\kernel\helper\OrderHelper;
use app\models\orioks\up\UpType;
use app\models\user\User;
use app\models\user\UserMobile;
use yii\helpers\ArrayHelper;
use app\models\orioks\Fac;
use app\models\orioks\group\Group;
use app\models\orioks\Semester;
use app\models\orioks\Student;
use app\models\orioks\up\UpSegment;
use app\models\orioks\Ball;
use app\models\orioks\dis\Dis;
use ReflectionMethod;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\web\TooManyRequestsHttpException;
use yii\db\ActiveQuery;
use app\models\news\Notification;
use app\models\orioks\dis\DisInfoStatus;
use app\models\test\ResetForm;
use yii\rest\ActiveController;

class ApiController extends Controller
{
    protected $student;

    /**
     * @inheritDoc
     */
    public function behaviors()
    {
        return array_merge(parent::behaviors(), [
            [
                'class' => \yii\filters\ContentNegotiator::className(),
                'formats' => [
                    'application/json' => \yii\web\Response::FORMAT_JSON,
                ]
            ]
        ]);
    }

```

```
    public function beforeAction($action)
    {
        \Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;
        if ($action->id != 'index') {
            $this->student = $this->findStudentByToken();
        }

        return true;
    }

    /**
     * @inheritdoc
     */
    public function afterAction($action, $result)
    {
        // Вызываем parent::parent::afterAction()
        $r = new ReflectionMethod(Controller::className(), 'afterAction');

        return $r->invoke($this, $action, $result);
    }

    public function actionIndex()
    {
        $user_agent = \Yii::$app->request->getHeaders()->get('User-Agent');
        $auth = \Yii::$app->request->getHeaders()->get('Authorization');
        $auth = explode(' ', $auth);
        $auth = base64_decode($auth[1]);
        $auth = explode(':', $auth);
        $user = User::findIdentity($auth[0]);
        if (!$user || !$user->validatePassword($auth[1])) {
            return json_encode(['error' => 'Неверный логин или пароль.'],
            ↪  JSON_UNESCAPED_UNICODE);
        }

        $student = $user->getCurrentStud();
        $token = \Yii::$app->getSecurity()->generateRandomString(32);

        $user_mobile = new UserMobile(['id_stud' => $student->id, 'token' => $token,
        ↪  'user_agent' => $user_agent]);
        $user_mobile->save();

        return json_encode(['token' => $token]);
    }

    public function actionStudent()
    {
```

```
        $student = $this->student;
        $op = $student->getCorrectUp()->getParentByType(UpType::OP);
        $actual_semester = Semester::getActual();

        return [
            'full_name' => $student->getFullName(),
            'group' => $student->group->getShortName(),
            'semester' => $actual_semester->id,
            'op' => $op->name,
            'np' => $op->parent->name,
        ];
    }

    public function actionDis()
    {
        $student = $this->student;

        $id_semester = \Yii::$app->request->get('semester');
        if (is_null($id_semester)) {
            $actual_semester = Semester::getActual();
        } else {
            $id_semester = (int)$id_semester;
            $actual_semester = Semester::findOne(['id' => $id_semester]);
            if (is_null($actual_semester)) throw new \Exception("Передан некорректный
             ↪  идентификатор семестра");
        }

        (new ActiveQuery(UpSegment::className()))->findWith(['semester'],
         ↪  $student->getCorrectUp()->upSegments);
        $actual_week = $actual_semester->getWeekByUp($student->getCorrectUp());
        list($dises, $offset_dises) = $this->getDises($student, $actual_semester,
         ↪  $actual_week, $actual_semester);
        $disciplines = [];
        foreach ($dises AS $k => $dis) {
            $disciplines[$k]['id'] = $dis['id'];
            $disciplines[$k]['name'] = $dis['name'];
            $disciplines[$k]['kaf'] = $dis['science']['kaf']['name'];
            $disciplines[$k]['ball'] = $dis['grade']['b'];
            $disciplines[$k]['maxBall'] = $dis['mvb'];
            $disciplines[$k]['formControl'] = $dis['formControl']['name'];
            $disciplines[$k]['dateExam'] = $dis['date_exam'];
            foreach ($dis['preps'] as $prep) {
                $disciplines[$k]['teachers'][] = $prep['name'];
            }
        }

```

```
134         return array_values($disciplines);
135     }
136 
137 
138     /**
139      * Возвращает итоговый массив со списком дисциплин студента
140      * @param Student $student
141      * @param Semester $semester
142      * @param int $actual_week
143      * @param Semester $actual_semester
144      * @return array
145      */
146     protected function getDises($student, $semester, $actual_week, $actual_semester)
147     {
148         //Получаем дисциплины студента
149         /** @var Dis[] $dises */
150         $dises = $student->getCorrectUp()->getDises()->andWhereSem($semester->id)-
            ↪  >with('disInfos')->all();
151         foreach ($dises as $k => $dis) {//отсеиваем невыбранные дисциплины
152             if (!$dis->isChosenByStud($student->id)) {
153                 unset($dises[$k]);
154             }
155         }
156         $student->getCorrectUp()->populateRelation('dises', $dises);
157         //-----------------------------
158 
159         (new ActiveQuery(Dis::className()))->findWith([ //подгружаем связные с
            ↪  дисциплинами данные
160             'disSegments' => function ($query) use ($semester) {
161                 /** @var ActiveQuery $query *//*@todo проверить многосеместровые
                    ↪  дисциплины*/
162                 $query->innerJoin(
163                     'up_segment',
164                     'up_segment.id=dis_segment.id_segment AND
                        ↪  up_segment.id_semester=:id_semester',
165                     [':id_semester' => $semester->id]
166                 );
167             },
168             'disSegments.upSegment',
169             'disSegments.allKms.type',
170             'disSegments.allKms.kmWeeks',
171             'disSegments.allKms.kmActives',
172             'disSegments.kms.kmWeeks',
173             'disSegments.kms.kmActives',
174             'disSegments.kms.balls',
175             'disSegments.allKms.irs.file',
```

```
            'disSegments.allKms.irs.type',
            'disSegments.allKms.balls' => function ($query) use ($student) {
                /** @var ActiveQuery $query */
                $query->andWhere(['id_stud' => $student->id]);
            },
            'science.kaf',
            'formControl',
            'linkTutorDis.disInfos'
        ], $student->getCorrectUp()->dises);

        /* ----- Подгатавливаем массивы представлению ----- */
        $dis_info = [];//Будет хранить информацию о успеваемости по дисциплине
        $kmGrWeek = [];//Будет хранить информацию о недели КМ у группы
        $now_km = [];//Будет хранить информацию о наличии на текущей неделе КМ
        $km_grade = [];//Будет хранить информацию об оценке за КМ
        $is_offset = [];//Будет хранить информацию является ли дисциплина
        ↪ перезачтенной
        $kmIrs = [];//Будет хранить список ИР КМ
        foreach ($student->getCorrectUp()->dises as &$dis) {//соберем дополнительную
        ↪ информацию, которую потом вставим в массив представлению
            $is_offset[$dis->id] = $dis->isOffsetByStud($student->id);
            //Собираем информацию об успеваемости по дисциплине
            $dis_info[$dis->id]['mvb'] = $dis->getCurrentKmSum($student->id_group,
            ↪ $semester, $actual_week); //Информация о текущем максимально
            ↪ возможносм балле студетна
            $dis_info[$dis->id]['grade'] =
            ↪ Util::solveGrade($dis->getCurrentSumToStudent($student, $semester,
            ↪ $actual_week, $dis_info[$dis->id]['mvb']),
            ↪ $dis_info[$dis->id]['mvb']);//Расчет оценки студента
            $dis_info[$dis->id]['grade']['f'] =
            ↪ $dis->getFullSumToStudent($student->id); //Информация об общей сумме
            ↪ баллов студетна
            $dis_info[$dis->id]['preps'] = $dis->getPrepsByStudent($student);
            $dis_info[$dis->id]['date_exam'] =
            ↪ $dis->getExamDateForGroup($student->id_group);

            //Начинаем собирать информацию о КМ
            OrderHelper::disKmByGroup($dis, $student->id_group, 'allKms');//фильтруем
            ↪ КМ по порядку недель
            foreach ($dis->disSegments[0]->allKms as $km) {
                $kmGrWeek[$km->id] = $km->getWeekForGroup($student->id_group);//тут
                ↪ берем неделю для группы и потом её всунем в массив

                //Считаем оценку
                if (!is_null($km->balls[0]->ball) ||
                ↪ $km->isActive($student->group->id)) {
```

```
209                 $km_grade[$km->id] = Util::solveGrade($km->balls[0]->ball,
                    ↪  $km->max_ball); //Если есть балл или КМ активно
210             } else {
211                 $km_grade[$km->id] = ['b' => '-', 'p' => '-', 'o' => 'n'];
                    ↪  //Иначе отстутствие балла - не вина студента
212             }
213             if ($km_grade[$km->id]['b'] == Ball::N) $km_grade[$km->id]['b'] = 'н';
214
215             if ($actual_semester->id == $semester->id) {//Если выбранный семетр -
                ↪  текущий, то ищем есть ли у дисциплины на текущей неделе КМ
216                 if ($kmGrWeek[$km->id] == $actual_week) {
217                     $now_km[$dis->id] = true;
218                 }
219             }
220             foreach ($km->irs as $_ir) {
221                 $ir = [
222                     'id_km' => $km['id'],
223                     'name' => $_ir->name,
224                     'link' => $_ir->file->getUrl(),
225                     'type' => $_ir->type->name,
226                     'label' => $_ir->getLabelColor(),
227                 ];
228                 $kmIrs[$km->id][] = $ir;
229             }
230         }
231     }
232
233     //Переводим все в итоговый массив и наполняет расчитанными в предыдущем цикле
        ↪  значениями
234     $a_dises = ArrayHelper::toArray($student->getCorrectUp()->dises, [], true,
        ↪  ['disSegments.allKms.type', 'disSegments.allKms.balls', 'science.kaf',
        ↪  'formControl']);
235     $_a_offset_disses = [];
236     foreach ($a_dises as $k => &$dis) {
237         if ($is_offset[$dis['id']]) {
238             $dis['grade'] = DisInfoStatus::getOffsetBall($is_offset[$dis['id']]);
239             $_a_offset_disses[] = $dis;
240             unset($a_dises[$k]);
241         } else {
242             $dis['preps'] = $dis_info[$dis['id']]['preps'];
243             $dis['mvb'] = $dis_info[$dis['id']]['mvb'];
244             $dis['grade'] = $dis_info[$dis['id']]['grade'];
245             $dis['now_km'] = $now_km[$dis['id']];
246             $dis['date_exam'] = $dis_info[$dis['id']]['date_exam'];
247
248             foreach ($dis['disSegments'][0]['allKms'] as &$km) {
```

```
                    $km['week'] = $kmGrWeek[$km['id']]; //Забираем недели групп
                    $km['grade'] = $km_grade[$km['id']]; //Забираем оценку
                    $km['irs'] = $kmIrs[$km['id']]; //Забираем ИР
                }
            }
        }

        return [$a_dises, $_a_offset_disses];
    }

    protected function findStudentByToken()
    {
        $auth = \Yii::$app->request->getHeaders()->get('Authorization');
        //$auth = 'Bearer UgAYzK8CWLqWDX2a2ZS1IRA0-jB6rgse';
        $auth = explode(' ', $auth);
        if ($auth[0] === 'Bearer' && !empty($auth[1])) {
            $model = UserMobile::find()->where(['token' => $auth[1]])->one();
            if ($model == null) {
                return false;
            }
            $model->last_used = date('Y-m-d H:i:s');
            $model->save();
            $student = Student::find()->where(['id' => $model->id_stud, 'main' =>
             ↪  1])->one();
            return $student;
        }

        return false;
    }
}
```

Листинг 3.1 — Контроллер, отвечающий за API(controllers/ApiController.php)

```
<?php

namespace app\models\orioks;

use app\models\orioks\km\Km;
use yii\helpers\VarDumper;
use yii\log\Logger;

/**
 * This is the model class for table "ball".
```

```
 *
 * @property integer $id
 * @property double $ball
 * @property integer $id_km
 * @property integer $id_stud
 *
 * @property Km $km
 * @property Student $student
 * @property BallChange[] $ballChanges
 */
class Ball extends \app\models\OrioksActiveRecord
{
    /**
     * Студент не посетил занятие
     */
    const N = -1;

    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return 'ball';
    }

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            [['ball'], 'number'],
            [['id_km', 'id_stud'], 'required'],
            [['id_km','id_stud'], 'integer'],
        ];
    }

    /**
     * Созранит дополнительно изменения балла в лог
     * @inheritdoc
     */
    public function afterSave($insert, $changedAttributes) {
        if(array_key_exists('ball',$changedAttributes)){
            $ballChange = new BallChange();
            $ballChange->id_ball=$this->id;
            $ballChange->old_ball=$changedAttributes['ball'];
```

```
            $ballChange->new_ball=$this->ball;
            $ballChange->save();
        }
        parent::afterSave($insert, $changedAttributes);
    }

    /**
     * @inheritdoc
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'ball' => 'Ball',
            'id_km' => 'Id Km',
            'id_stud' => 'Студент',
        ];
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getKm()
    {
        return $this->hasOne(Km::className(), ['id' => 'id_km']);
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getStudent()
    {
        return $this->hasOne(Student::className(), ['id' => 'id_stud']);
    }
    /**
     * @return \yii\db\ActiveQuery
     */
    public function getBallChanges()
    {
        return $this->hasMany(BallChange::className(), ['id_ball' => 'id']);
    }

    /**
     * Получить корректный балл студента.
     * В Случае если балл <0 возвращает 0
     * @return double
```

```
     */
    public function getCorrectBall() {
        if($this->ball<0) return 0;
        return $this->ball;
    }

    /**
     * Заменить значение балла на балл из лога определённую дату.
     * @param $date
     */
    public function loadBallByDate($date){
        /** @var BallChange $lastChange */
        $lastChange = $this->getBallChanges()->andWhere(['<=','datetime',$date."
        ↪  23:59:00"])->orderBy('datetime desc')->limit(1)->one();
        $this->ball = $lastChange ? $lastChange->new_ball : $this->ball;
    }
}
```

Листинг 3.2 — Модель сущности балл (models/Ball.php)

```
<?php

namespace app\models\orioks;

use app\models\orioks\dis\DisInfo;
use app\models\orioks\group\Group;
use app\models\orioks\port\PortAuthor;
use app\models\orioks\practice\LinkStudent;
use app\models\orioks\up\Up;
use app\models\questionnaire\Questionnaire;
use Yii;

/**
 * This is the model class for table "student".
 *
 * @property integer $id
 * @property string $login
 * @property string $numst
 * @property integer $id_group
 * @property integer $id_up
 * @property integer $main
 * @property boolean $active
 * @property boolean $academ
```

```
 * @property boolean $invalid
 * @property boolean $needToExpulsion
 *
 * @property Group $group
 * @property Up $up
 * @property DisInfo[] $disInfos
 * @property TimeUnvisit[] $timeUnvisits
 * @property StudMessage[] $studMessages
 * @property Sanction[] $sanctions
 * @property Ball $balls
 * @property DebtBall[] $debtBalls
 * @property PortAuthor[] $portAuthors
 * @property Questionnaire $questionnaire
 * @property LinkStudent $linkStudentContract
 */
class Student extends \app\models\OrioksActiveRecord
{
    use \app\components\name_manager\NameMethodTrait;
    /**
     * @var Up УП студента
     */
    protected $_correctUp;
    /**
     * @var boolean состаяние - должник или нет
     */
    protected $_isDebt;
    /**
     * @var boolean состаяние - должник и имеет долг без даты
     */
    protected $_hasDebtWithoutDate;
    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return 'student';
    }

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            [['login', 'active'], 'required'],
            [['numst'], 'string'],
```

```
            [['id_group', 'id_up', 'main',
             ↪  'needToExpulsion','academ','active','invalid'], 'integer'],
        ];
    }

    /**
     * @inheritdoc
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'login' => 'Login',
            'numst' => 'numst',
            'id_group' => 'Id Group',
            'id_up' => 'Id Up',
            'main' => 'Main',
            'active' => 'Active',
            'academ' => 'Academ',
            'needToExpulsion' => 'needToExpulsion'
        ];
    }
    /**
     * @inheritdoc
     */
    public function fields() {
        $field =  parent::fields();
        $field['fullName']=function (){
            return $this->getFullName();
        };
        $field['shortName']=function (){
            return $this->getShortName();
        };
        $field['last_name']=function (){
            return Yii::$app->get('name_provider')->name($this->login)->last_name;
        };
        return $field;
    }

    /**
     * @inheritdoc
     */
    public function afterSave($insert, $changedAttributes) {
        if(isset($changedAttributes['active']) && $changedAttributes['active']==1 &&
         ↪  $this->active==0){
            //Если сюда зашли, значит активность была изменена с "активен" на "не
             ↪  активен". Значит надо закрыть все долги студента
```

```
            foreach ($this->disInfos as $disInfo){
                if($disInfo->debt==1 && $disInfo->debt_closed==0){
                    $disInfo->debt_closed=1;
                    $disInfo->save();
                }
            }
        }
        if(array_key_exists('id_up',$changedAttributes) &&
        ↪  is_null($changedAttributes['id_up']) && !is_null($this->id_up)){
            //Если сюда зашли, значит студента перевели на ИУП. Значит надо закрыть
            ↪  все долги студента в РУПе

            foreach ($this->group->up->dises as $dis){
                if(!is_null($disInfo=$dis->getDisInfoByStudent($this->id))){
                    if($disInfo->debt==1){
                        $disInfo->debt_closed=1;
                        $disInfo->save();
                    }
                }
            }
        }
        parent::afterSave($insert, $changedAttributes); // TODO: Change the
        ↪  autogenerated stub
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getGroup(){
        return $this->hasOne(Group::className(),['id'=>'id_group']);
    }
    /**
     * @return \yii\db\ActiveQuery
     */
    public function getUp(){
        return $this->hasOne(Up::className(),['id'=>'id_up'])->inverseOf('student');
    }
    /**
     * @return \yii\db\ActiveQuery
     */
    public function getDisInfos()
    {
        return $this->hasMany(DisInfo::className(), ['id_stud' => 'id']);
    }
    /**
     * @return \yii\db\ActiveQuery
```

```
     */
    public function getTimeUnvisits()
    {
        return $this->hasMany(TimeUnvisit::className(), ['login' => 'login']);
    }
    /**
     * @return \yii\db\ActiveQuery
     */
    public function getStudMessages()
    {
        return $this->hasMany(StudMessage::className(), ['id_stud' => 'id']);
    }
    /**
     * @return \yii\db\ActiveQuery
     */
    public function getSanctions()
    {
        return $this->hasMany(Sanction::className(), ['id_stud' => 'id']);
    }
    /**
     * @return \yii\db\ActiveQuery
     */
    public function getBalls()
    {
        return $this->hasMany(Ball::className(), ['id_stud' => 'id']);
    }
    /**
     * @return \yii\db\ActiveQuery
     */
    public function getDebtBalls()
    {
        return $this->hasMany(DebtBall::className(), ['id_stud' => 'id']);
    }
    /**
     * @return \yii\db\ActiveQuery
     */
    public function getPortAuthors()
    {
        return $this->hasMany(PortAuthor::className(), ['id_stud' => 'id']);
    }
    /**
     * @return \yii\db\ActiveQuery
     */
    public function getQuestionnaire()
    {
        return $this->hasOne(Questionnaire::className(), ['id_stud' => 'id']);
```

```
203     }
204     /**
205      * @return \yii\db\ActiveQuery
206      */
207     public function getLinkStudentContract()
208     {
209         return $this->hasOne(LinkStudent::className(), ['id_stud' => 'id']);
210     }
211 
212     /**
213      * Получить УП студента, в зависимости от id_up
214      * @return Up
215      * @throws \Exception
216      */
217     public function getCorrectUp(){
218         if(is_null($this->_correctUp)){
219             if($this->isIndividual()){
220                 $this->_correctUp=Up::findOne($this->id_up);
221             }else{
222                 $this->_correctUp=Up::findOne($this->group->id_up);
223             }
224             if(is_null($this->_correctUp)) throw new \Exception("Не найден учебный
                ↪  план!");
225         }
226         return $this->_correctUp;
227     }
228 
229     /**
230      * Является студент индивидуальщзиком
231      * @return bool
232      */
233     public function isIndividual(){
234         return !is_null($this->id_up);
235     }
236 
237     /**
238      * Является ли студент должником
239      */
240     public function isDebt(){
241         if(is_null($this->_isDebt)){
242             foreach ($this->disInfos as $disInfo){
243                 if ($disInfo->debt && !$disInfo->debt_closed){
244                     $this->_isDebt=true;
245                     goto ret;
246                 }
247             }
```

```
248             $this->_isDebt=false;
249         }
250         ret:
251         return $this->_isDebt;
252     }
253 
254     public function hasDebtWithoutDate() {
255         if(is_null($this->_hasDebtWithoutDate)){
256             if(!$this->isDebt()){
257                 $this->_hasDebtWithoutDate=false;
258                 goto ret;
259             }
260 
261             foreach ($this->disInfos as $disInfo){
262                 if ($disInfo->debt && !$disInfo->debt_closed &&
                    ↪  !$disInfo->debt_control_date){
263                     $this->_hasDebtWithoutDate=true;
264                     goto ret;
265                 }
266             }
267             $this->_hasDebtWithoutDate=false;
268         }
269         ret:
270         return $this->_hasDebtWithoutDate;
271     }
272 
273     /**
274      * Получить название группы
275      * @return string
276      */
277     public function getFullGroupName(){
278         return $this->group->getFullName($this);
279     }
280 
281     /**
282      * Получить логический идентификатор записи студента.
283      * Это строка вида <логин>-<логический идентификатор группы> в нижнем регистре
284      * @return string
285      */
286     public function getLogicalId(){
287         return $this->login.'-'.$this->group->getLogicalId();
288     }
289 }
```

Листинг 3.3 — Модель сущности студент (models/Student.php)