МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВПО

«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информатика и программное обеспечение»

**Разработка системы управления проектами**

**К У Р С О В О Й П Р О Е К Т**

*по дисциплине* **«Разработка Web-узлов и приложений»**

всего листов \_\_\_

Выполнил: студ. гр. О-18-ПРИ-рпс-Б

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лядов В. С.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Руководитель: к.т.н., доц. Белов Е.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

**Брянск 2020**

Contents

[Введение 3](#_Toc88529705)

[1. Анализ предметной области 4](#_Toc88529706)

[1.1. Описание предметной области 4](#_Toc88529707)

[1.2 . Функциональные требования 4](#_Toc88529708)

[2. Технологический стек 5](#_Toc88529709)

[3. РАзработка базы данных 6](#_Toc88529710)

[4. Программная реализация системы и функциональное тестирование 11](#_Toc88529711)

[4.1. Авторизация 11](#_Toc88529712)

[4.2. Управление окладами 11](#_Toc88529713)

[4.3. Таблица занятости 12](#_Toc88529714)

[4.4. Управление проектами 13](#_Toc88529715)

[Заключение 15](#_Toc88529716)

[Список литературы 16](#_Toc88529717)

# Введение

В данной работе описан процесс разработки сайта для управления проектами. Описывается актуальность работы, выбор технологического стека, процесс разработки и тестирования сайта. Система предназначена для управления проектами в ходе разработки ПО, формирование таблицы занятости, ведение зарплатного учёта, производственного календаря.

# Анализ предметной области

## Описание предметной области

В ходе разработки крупного программного обеспечения возникает необходимость в контроле занятости всех сотрудников, которые задействованы в реализации продукта: разработчики, тестировщики, менеджеры, контент менеджеры. Важную роль играет ведения зарплатного учёта, системы повышений, а также распределение задач среди каждого сотрудника. В ходе разработки так же есть необходимость вести статистические отчёты и следить за маржинальностью продукта, контролировать учёт внешних расходов. Все перечисленные пункты становиться трудоёмко отслеживать даже на одном проекте, а когда вопрос становиться о ведении разработкой нескольких проектов и управление десятками исполнителей, то данный вопрос становиться ещё актуальнее.

## . Функциональные требования

Сайт должен поддерживать возможность авторизации пользователей. Реализацию разграничения ролей: менеджеры, разработчики, тестировщики, контент менеджеры, модераторы, администраторы. Каждая исполнительная роль (разработчик, тестировщик, контент менеджер) должен иметь возможность просмотреть личный профиль, иметь возможность его изменить. Так же он должен иметь возможность видеть свою занятость, в каких проектах он учувствует и какой вклад в неё, а также информацию о проектах. Каждому сотруднику этих ролей должен быть доступен раздел WIKI для просмотра информации о ходе ведения разработки. Менеджер и руководитель должны иметь возможность назначать занятость на проект, а также бронировать исполнителей на свои задачи. Создавать проекты, задачи, добавлять исполнителей. Администратор должен иметь возможность вести контроль за пользователями: создавать, удалять и редактировать. Управлять производственным календарём, окладами, бонусами а так же вести контроль за системой автоматического повышения сотрудников. Администратор так же имеет системную роль, может управлять типами проектов и их статусами, чистить кеш системы.

# Технологический стек

Для разработки серверной части было принято решение использовать язык программирования PHP версии 7.4, а так же высокоскоростной фреймворк Yii2.

Yii 2 представляет собой фреймворк, которые реализует паттеры MVC для организации структуры кода, Active Record для упрощения работы с БД. Серверная часть была архитектурно проектирована как REST API модуль, который состоит из ряда REST API методов, с которым взаимодействует Front-end. Все запросы распределяются по блокам управления в виде контроллеров, в них происходит обработка, валидация входных данные, а так же изменения данных в базе данных, посредством моделей, которые в свою очередь описывают каждую сущность. После чего формируется ответ и отправляется на клиентскую часть.

Организация ролей реализована с помощью паттерна RBAC.

Вся административная панель построена так же на фреймворке Yii2, который позволяет через третий компонент — представления, работать с формированием визуальных страниц и отображением на клиентской стороне. В качестве шаблона для представления использовался AdminLTE.

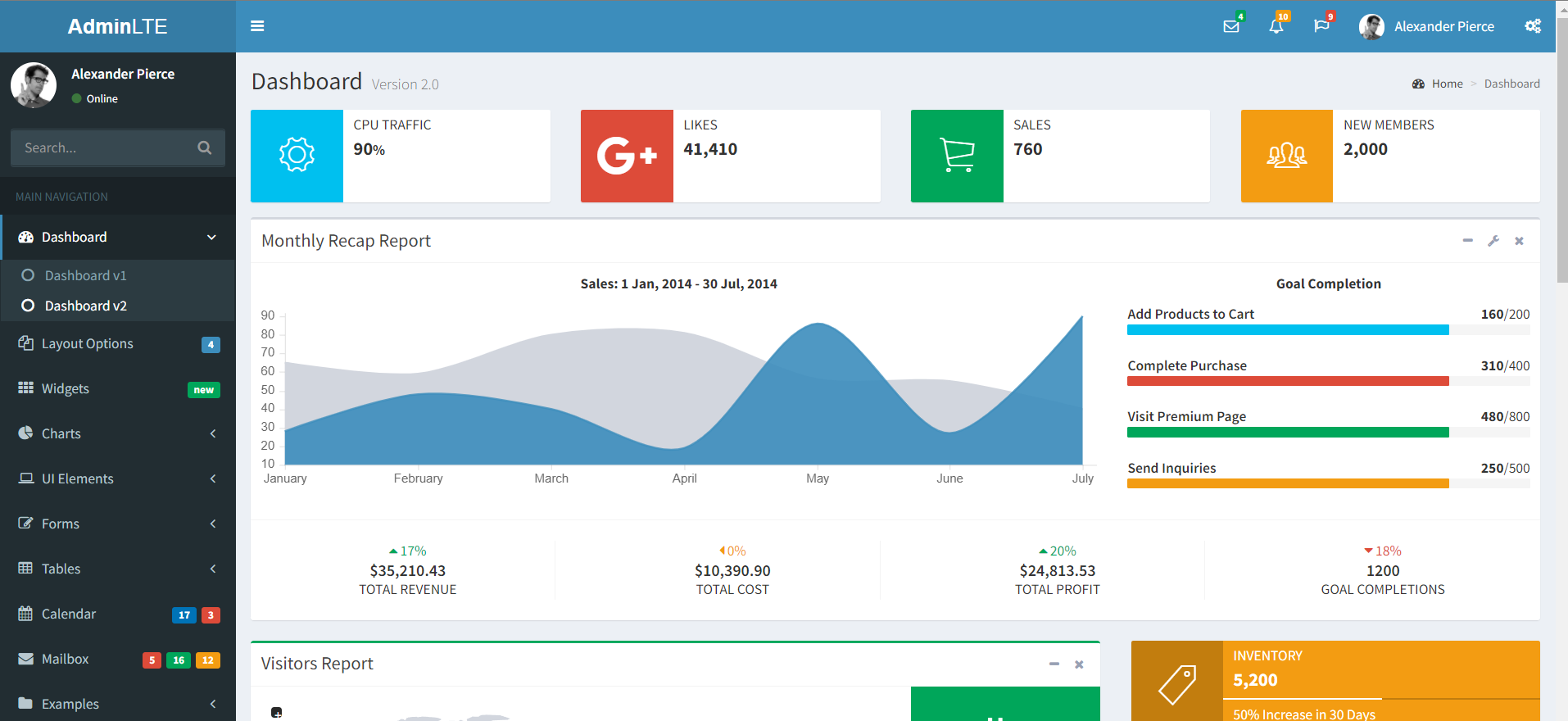


Рис. 1. Набор элементов AdminLTE

Клиентскую часть сайта было решено сделать на React.js в виде многостраничного, динамического портала.

# РАзработка базы данных

База данных приложения содержит 33 таблицы, вот некоторые из них:

* users (пользователи и в том числе администраторы)
* profiles (справочник с подробной информацией о каждом пользователе)
* project(проекты)
* grade (оклады пользователей)
* task (задачи в проектах)
* implementers (исполнители к задачам).

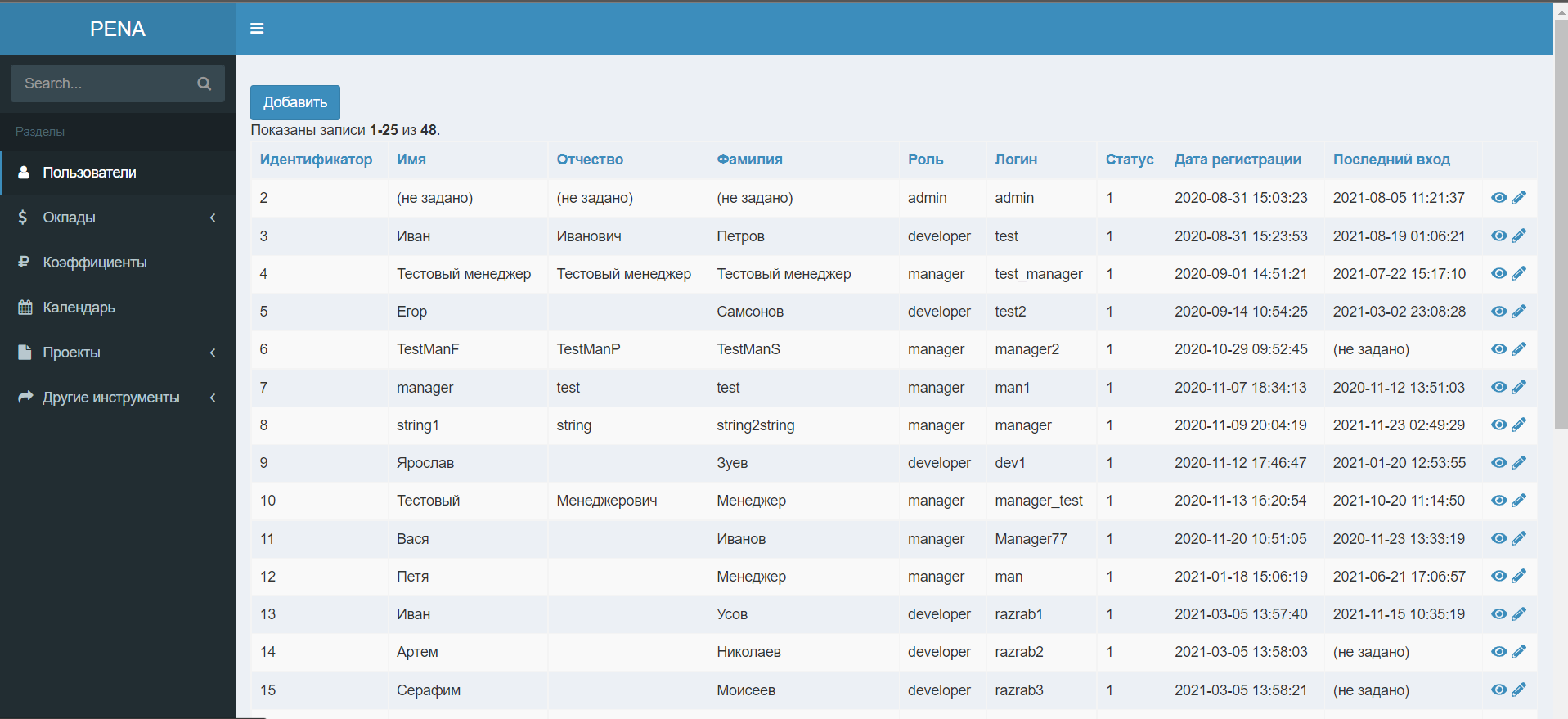


Рис. 2. Главная страница панели администратора

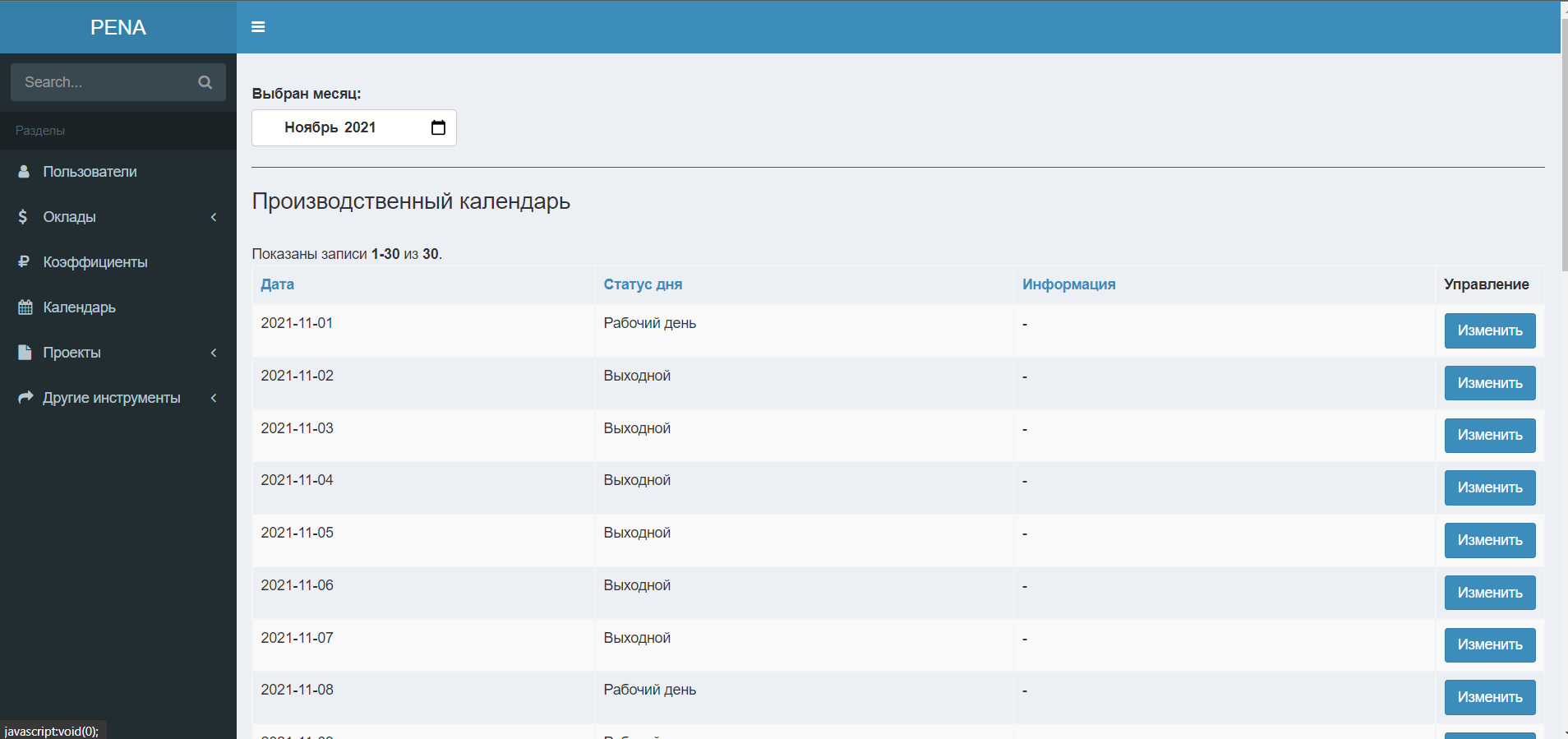


Рис. 3. Страница управления производственным календарём

Листинг 1. Контроллер управления пользователями из панели администратора

**<?php  
  
namespace** app\modules\admin\controllers;  
  
**use** app\models\Grade;  
**use** app\models\GradeLog;  
**use** app\models\Profile;  
**use** app\models\User;  
**use** app\modules\admin\models\EditUserForm;  
**use** app\modules\admin\models\UserForm;  
**use** Yii;  
**use** yii\data\ActiveDataProvider;  
**use** yii\db\Query;  
**use** yii\web\NotFoundHttpException;  
**use** yii\web\Response;  
**use** yii\widgets\ActiveForm;  
  
**class** UsersController **extends** AdminController  
{  
 **public function** actionIndex()  
 {  
 $dataProvider = **new** ActiveDataProvider(  
 [  
 **'query'** => User::*find*()->joinWith([**'profile'**]),  
 **'pagination'** => [  
 **'pageSize'** => 25  
 ]  
 ]  
 );  
  
 $dataProvider->**sort**->**attributes**[**'profile.firstName'**] = [  
 **'asc'** => [Profile::*tableName*() . **'.firstName'** => ***SORT\_ASC***],  
 **'desc'** => [Profile::*tableName*() . **'.firstName'** => ***SORT\_DESC***]  
 ];  
  
 $dataProvider->**sort**->**attributes**[**'profile.lastName'**] = [  
 **'asc'** => [Profile::*tableName*() . **'.lastName'** => ***SORT\_ASC***],  
 **'desc'** => [Profile::*tableName*() . **'.lastName'** => ***SORT\_DESC***]  
 ];  
  
 $dataProvider->**sort**->**attributes**[**'profile.patronymic'**] = [  
 **'asc'** => [Profile::*tableName*() . **'.patronymic'** => ***SORT\_ASC***],  
 **'desc'** => [Profile::*tableName*() . **'.patronymic'** => ***SORT\_DESC***]  
 ];  
  
 **return** $this->render(**'index'**, [**'provider'** => $dataProvider]);  
 }  
  
  
  
 **public function** actionView($id)  
 {  
 $user = User::*find*()  
 ->select([  
 **'id'**,  
 **'login'**,  
 **'status'**,  
 **'createDate'**,  
 **'lastVisitDate'**,  
 **'role'** ])  
 ->where([**'id'** => $id])  
 ->asArray()  
 ->one();  
  
 $profile = Profile::*findOne*($id);  
 **if** (!$user) {  
 **throw new** NotFoundHttpException(**'Пользователь не найден.'**);  
 }  
  
 $gradeHistory = **new** ActiveDataProvider(  
 [  
 **'query'** => (**new** Query())->from(**'grade\_log'**)  
 ->where([**'grade\_log.userId'** => $profile->**userId**])  
 ->orderBy([**'grade\_log.date'** => ***SORT\_DESC***])  
 ->innerJoin(**'grade'**, **'grade.gradeId = grade\_log.gradeId'**)  
 ]  
 );  
  
  
 **return** $this->render(**'view'**, [  
 **'user'** => $user,  
 **'profile'** => $profile,  
 **'gradeHistory'** => $gradeHistory  
 ]);  
 }  
  
  
  
 **public function** actionAdd()  
 {  
 $user = **new** UserForm();  
 $profile = **new** Profile();  
 $data = Yii::*$app*->**request**->post();  
 $grades = [];  
  
 **if** (Yii::*$app*->**request**->**isAjax** && $user->load($data) && $profile->load($data)) {  
 Yii::*$app*->**response**->**format** = Response::***FORMAT\_JSON***;  
 **return** ActiveForm::*validate*($user);  
 }  
  
 **if** ($user->load($data) && $profile->load($data)) {  
 **if** ($user->**errors** || $profile->**errors**) {  
 Yii::*$app*->**session**->setFlash(  
 **'error'**,  
 *array\_pop*($user->**errors**) .  
 *array\_pop*($profile->**errors**)  
 );  
 } **else** {  
 **if** ($user->save()) {  
 $role = Yii::*$app*->**authManager**->getRole($user->**role**);  
 Yii::*$app*->**authManager**->assign($role, $user->**id**);  
 $profile->**userId** = $user->**id**;  
 **if** ($profile->save()) {  
 Yii::*$app*->**session**->setFlash(**'success'**, **'Пользователь успешно создан.'**);  
 **return** $this->redirect([**'users/'**]);  
 }  
 }  
 }  
 }  
  
 **return** $this->render(  
 **'add'**,  
 [  
 **'model'** => $user,  
 **'profile'** => $profile,  
 **'roles'** => User::***ROLES\_DESCRIPTION***,  
 **'grades'** => $grades  
 ]  
 );  
 }  
  
 **public function** actionUpdate($id)  
 {  
 $model = EditUserForm::*findOne*([**'id'** => $id]);  
 **if** (!$model) {  
 **throw new** NotFoundHttpException(**"Пользователь не найден."**);  
 }  
  
 $auth = Yii::*$app*->**authManager**;  
 $postData = Yii::*$app*->**request**->post();  
  
  
  
 **if** ($model->load($postData) && $model->**profile**->load($postData)) {  
 **if** ($model->save() && $model->**profile**->save()) {  
 $currentMonth = *date*(**'Y-m'**);  
 $auth->revokeAll($model->**id**);  
 $role = $auth->getRole($model->**role**);  
 $auth->assign($role, $model->**id**);  
 GradeLog::*updateAll*([  
 **'gradeId'** => $model->**profile**->**gradeId** ], [  
 **'grade\_log.userId'** => $model->**id**,  
 **'grade\_log.date'** => $currentMonth  
 ]);  
  
 Yii::*$app*->**session**->setFlash(**'success'**, **'Запись обновлена.'**);  
 **return** $this->redirect([**'users/'**]);  
 }  
 $message = (!**empty**($model->**profile**->**errors**)) ?  
 *array\_pop*($model->**profile**->**errors**)  
 : *array\_pop*($model->**errors**);  
 Yii::*$app*->**session**->setFlash(**'error'**, $message);  
 }  
  
 $grades = Grade::*getGradeList*($model->**role**);  
  
 **return** $this->render(  
 **'update'**,  
 [  
 **'model'** => $model,  
 **'roles'** => User::***ROLES\_DESCRIPTION***,  
 **'grades'** => $grades  
 ]  
 );  
 }  
}

# Программная реализация системы и функциональное тестирование

## Авторизация

Портал состоит из двух модулей: API (модуль, обслуживающий ajax запросы) и административная. Каждая из них наделена своей логикой авторизации.

Для API авторизация представляет собой метод открыт по CORS заголовкам и доступен всем неавторизованным пользователям. Получивший запрос с логином и пароль контроллер обрабатывает, проверяет если такой пользователей в БД и если есть авторизует в модуле.

Для административной панели страница авторизации представляет собой представление с полями логин и пароль, после заполнения полей и отправки формы, в контроллере проверяется подлинность данных и в случае истинны фреймворк авторизует пользователя в модуле.

## Управление окладами

Отдельный блок в навигации отвечает за управлениями окладами сотрудников. В данном разделе можно создавать оклады для различных исполнительных ролей, добавлять бонусы к окладам. Рассчитывать эффективность и производить контроль над автоматическим повышением.

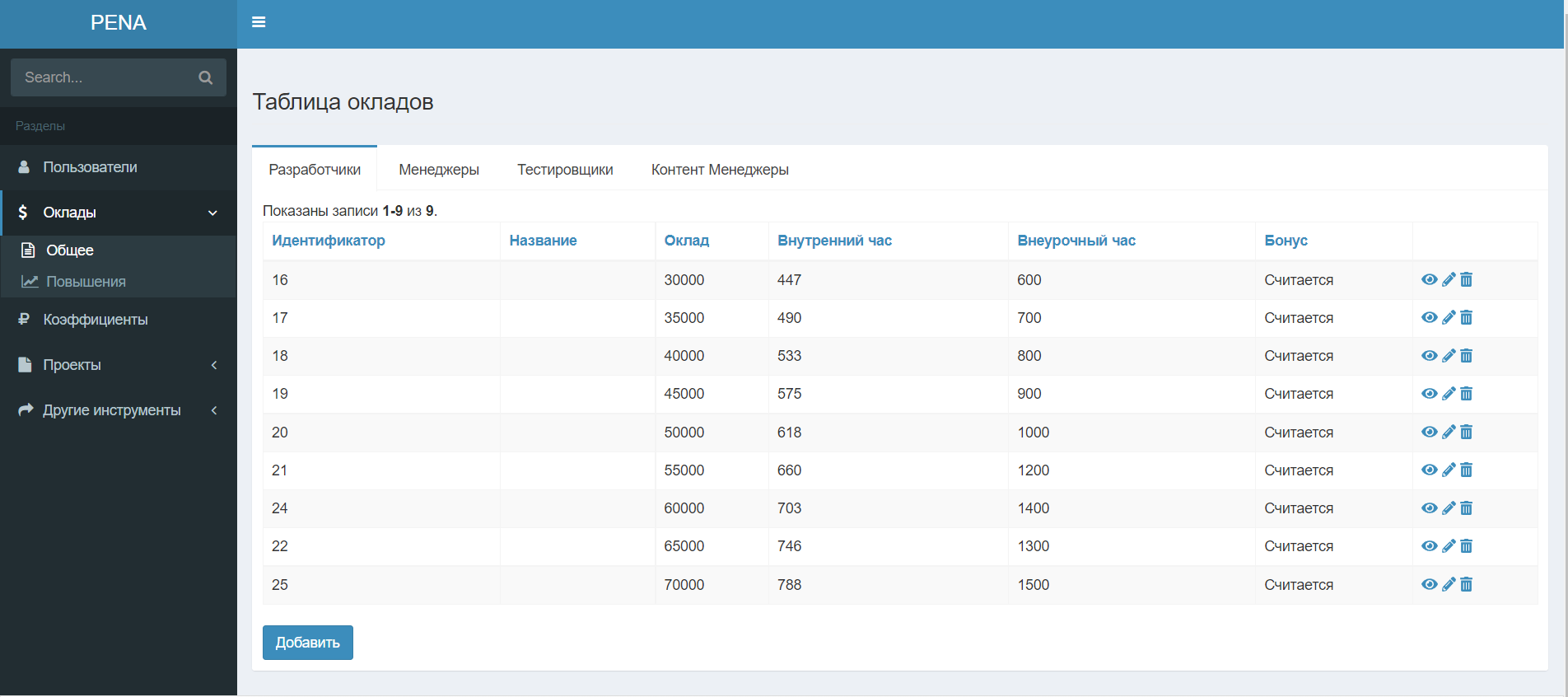


Рис. 4. Страница с окладами

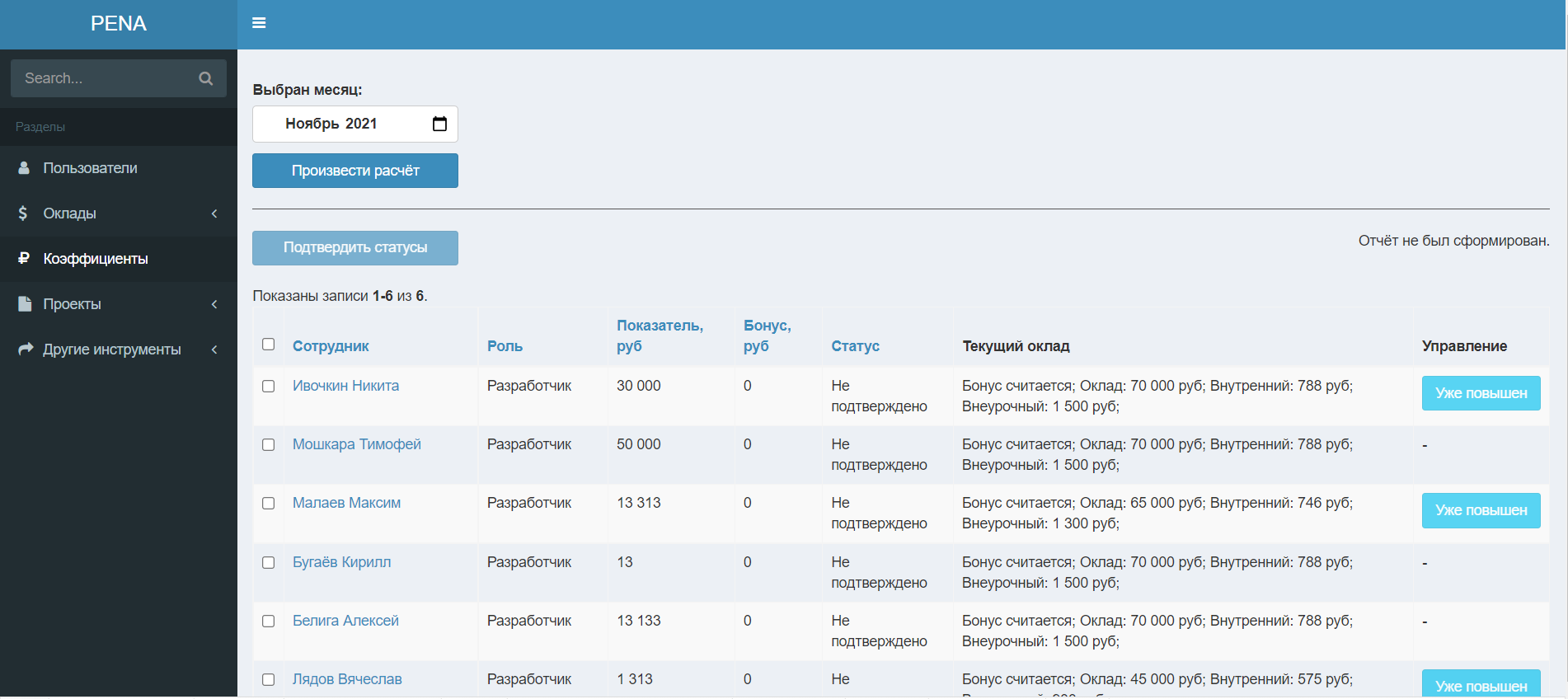


Рис. 5. Страница с расчётом эффективности за каждый месяц

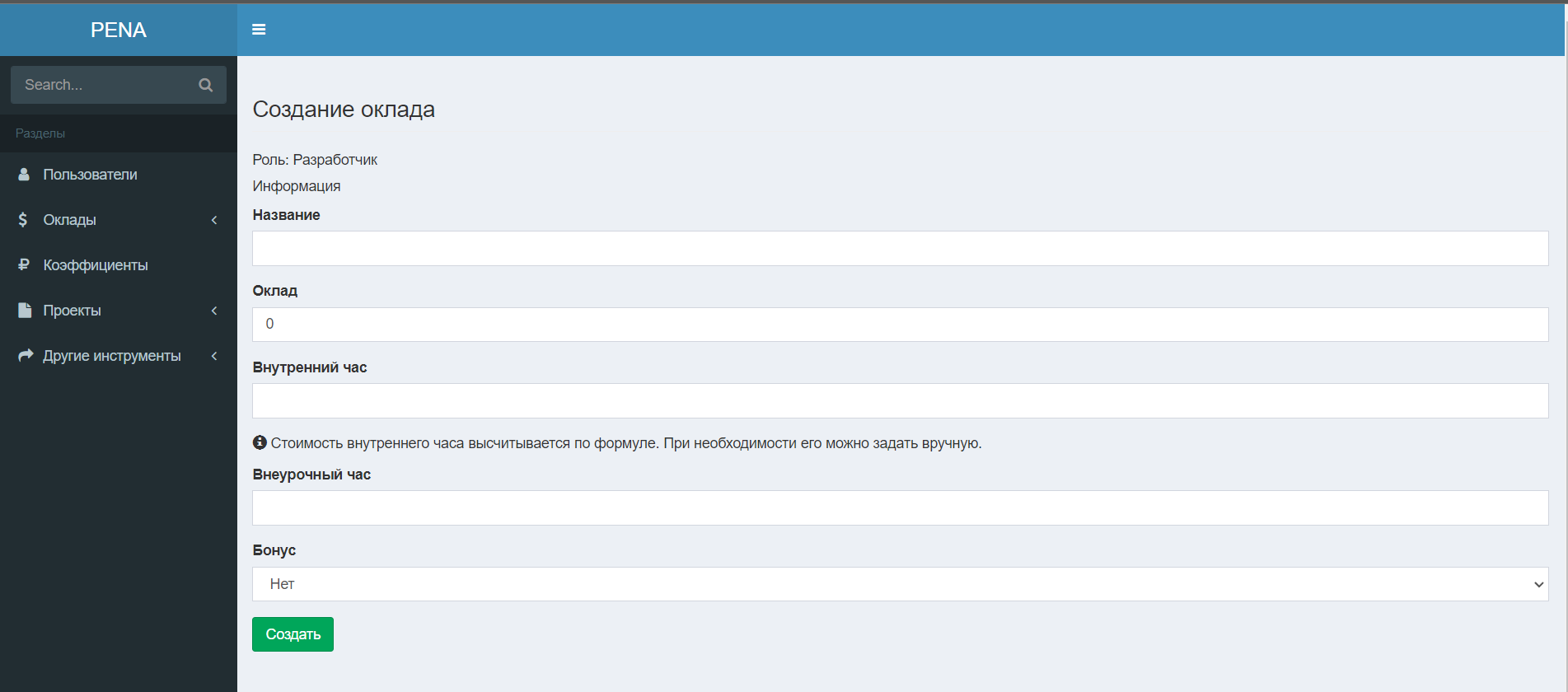


Рис. 6. Внешний вид страницы создания оклада

## Таблица занятости

Представляет собой страницу с управлением занятостью всех исполнителей разделённое по ролям. На странице есть возможность поставить занятость исполнителю, назначив на проект. Отслеживать отпуска и выходные дни.

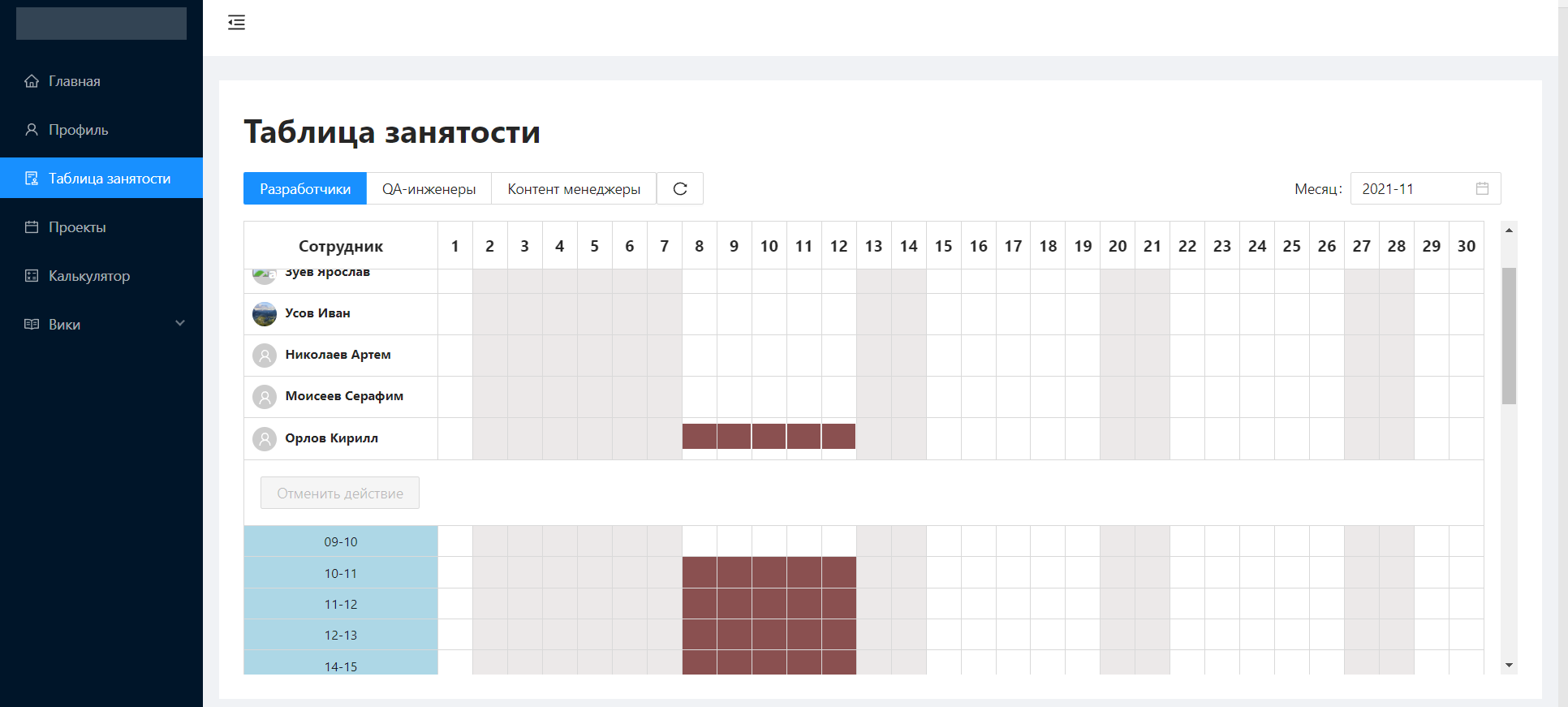


Рис. 7. Внешний вид таблицы занятости

## Управление проектами

Ниже представлен внешний вид раздела управления проектами. Можно просматривать список проектов, фильтровать, сортировать. Просматривать проект подробно, назначать задачи, исполнителей и показатели за каждый месяц.

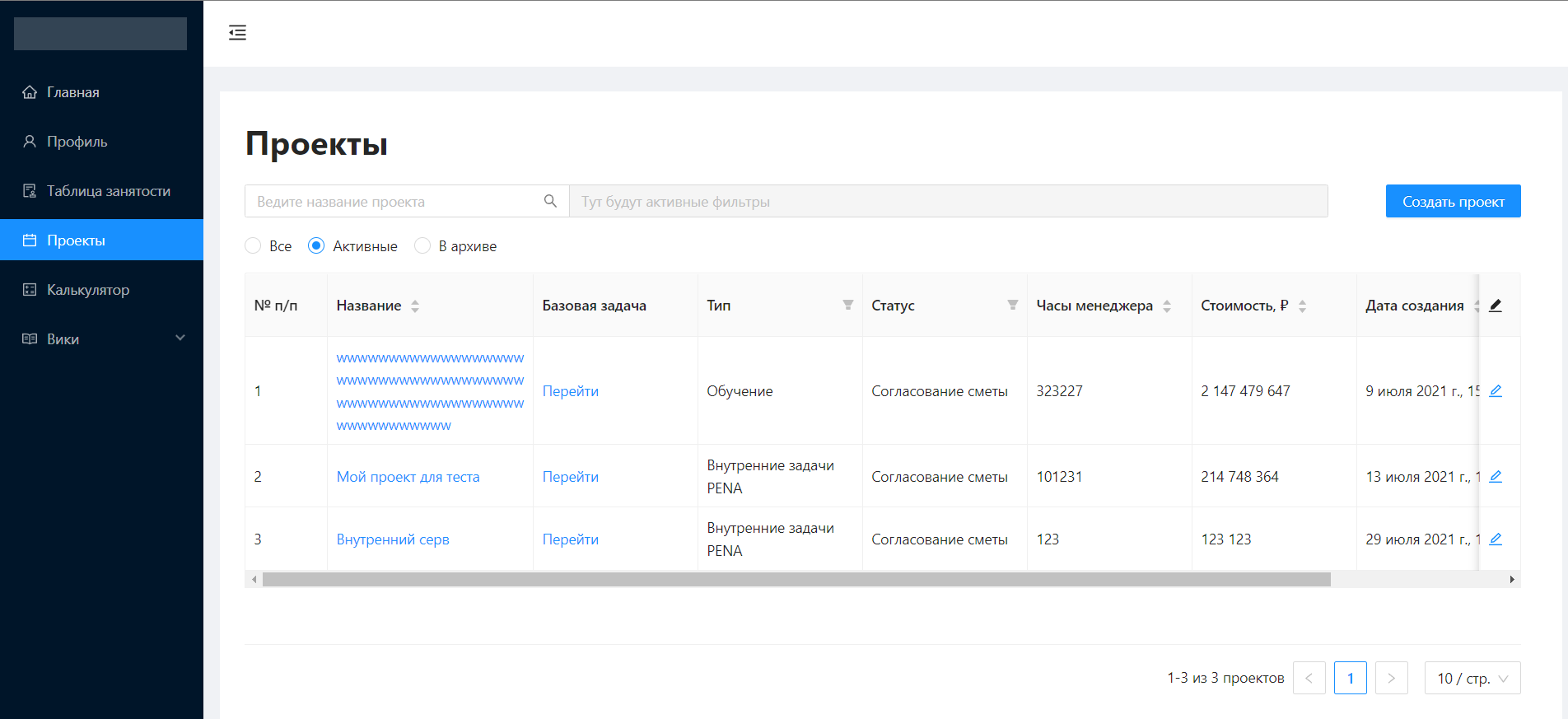


Рис. 7. Внешний вид списка проектов

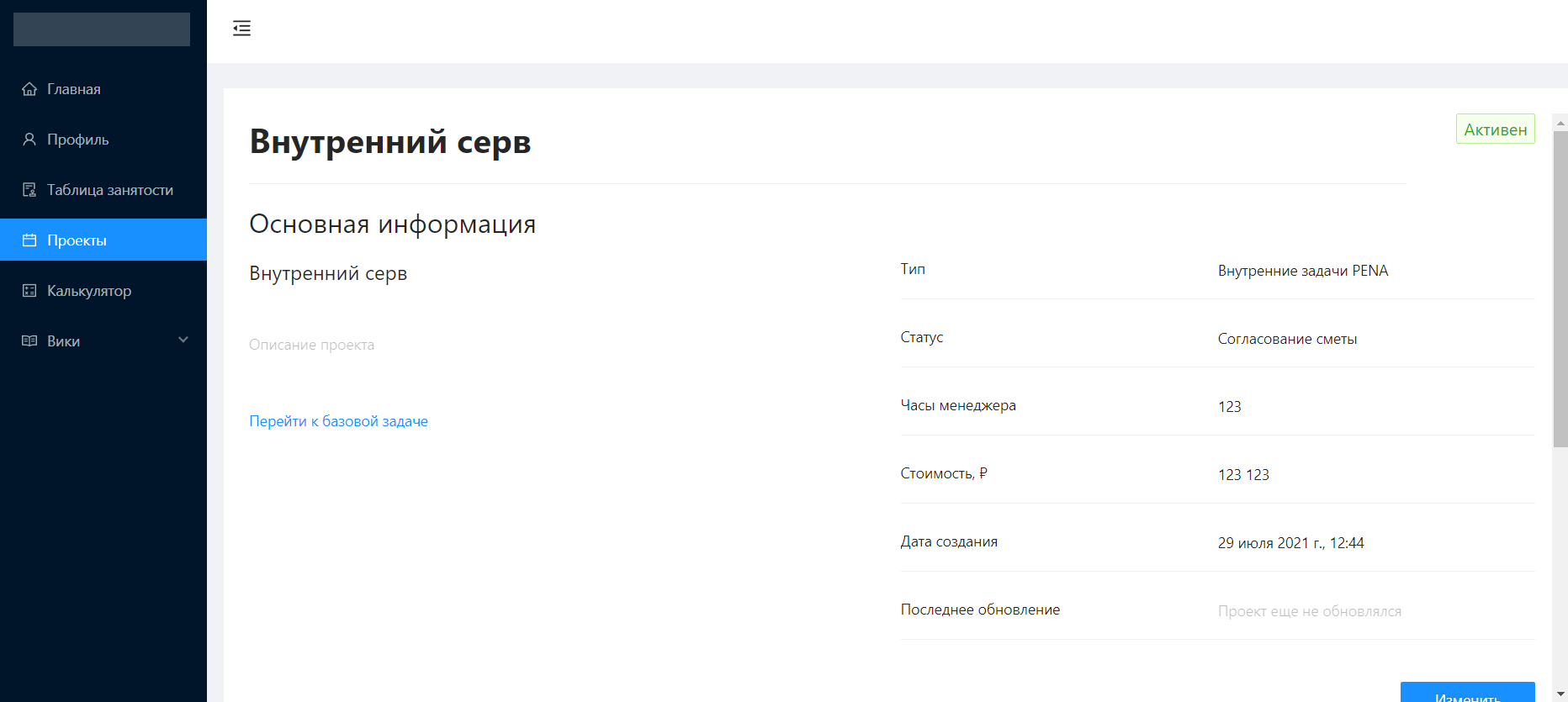


Рис. 8. Страница проекта

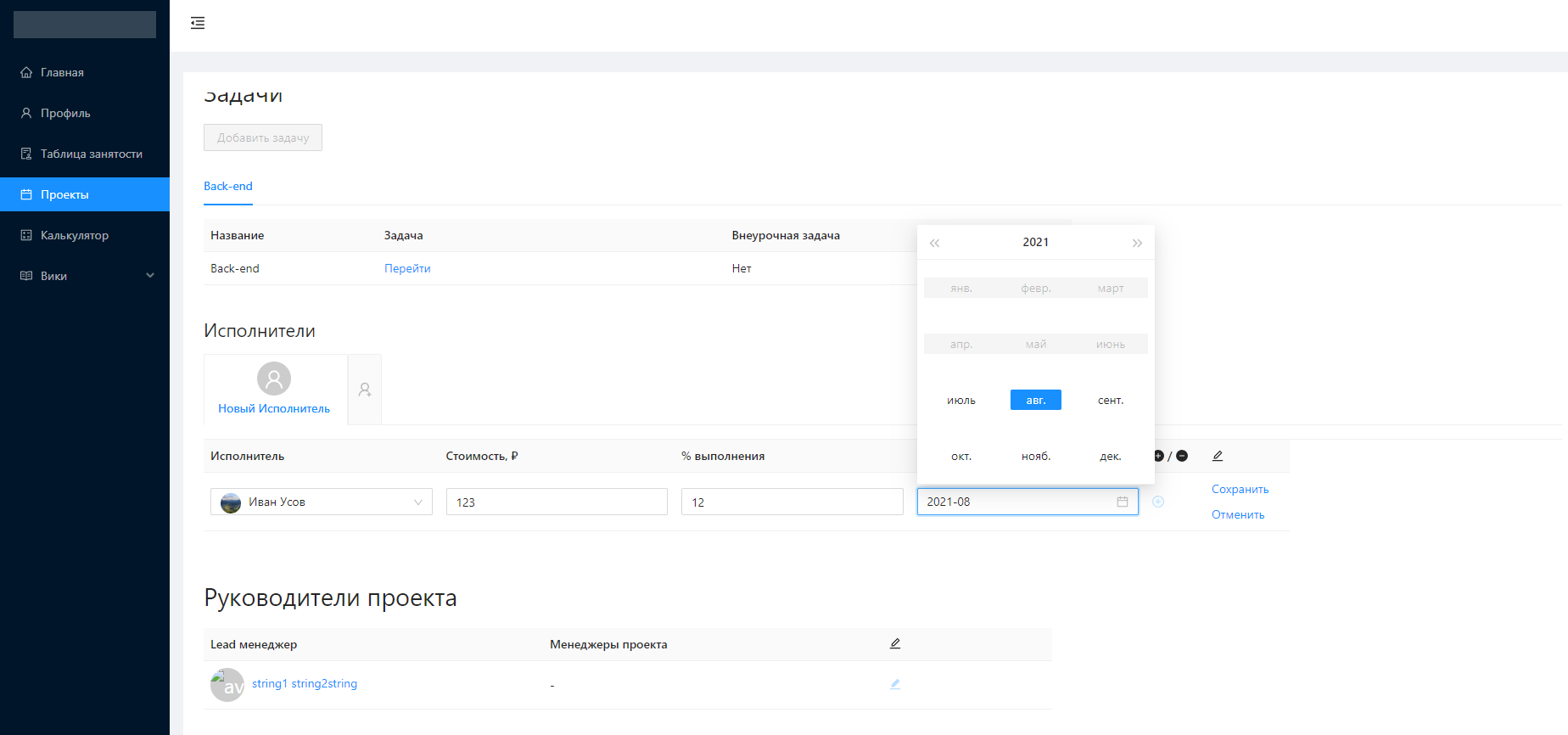


Рис. 9. Создание задачи и назначение исполнителя

# Заключение

Результатом выполнения курсового проекта стала клиент-серверная система управлением проектами. Реализованы административный и клиентский разделы, работа с проектами, исполнителями, задачами, их окладами и бонусами. Так же добавлена таблица занятости, реализующий собственный производственный календарь.

В процессе разработки я выучил язык программирования PHP, освоил фреймворк Yi2, познакомился с созданием REST API приложения, документированием API.

# Список литературы

1. Бардзелл Джеффри Macromedia Dreamweaver MX 2004 с ASP, ColdFusion и PHP. Из первых рук (+ CD-ROM); Эком - М., 2016. - 560 c.

2. Бенкен Елена PHP, MySQL, XML. Программирование для Интернета; БХВ-Петербург - М., 2017. - 336 c.

3. Гизберт Дамашке PHP и MySQL; НТ Пресс - М., 2016. - 569 c.

4. Дронов В. PHP, MySQL и Dreamweaver. Разработка интерактивных Web-сайтов; БХВ-Петербург - М., 2016. - 480 c.

5. Дунаев В.В. HTML, скрипты и стили; БХВ-Петербург - М., 2017. - 527 c.

6. Жадаев Александр PHP для начинающих; Питер - М., 2016. - 768 c.

7. Зандстра Мэтт PHP. Объекты, шаблоны и методики программирования; Вильямс - М., 2016. - 560 c.

8. Колисниченко Денис PHP и MySQL. Разработка Web-приложений; БХВ-Петербург - М., 2017. - 560 c.

9. Кузнецов М., Симдянов И., Голышев С. PHP 5. Практика создания Web-сайтов; БХВ-Петербург - М., 2017. - 960 c.

10. Кузнецов Максим , Симдянов Игорь Самоучитель PHP 5; БХВ-Петербург - М., 2017. - 560 c.