Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

Югорский государственный университет

Институт (НОЦ) технических систем и информационных технологий

КУРСОВОЙ ПРОЕКТ

по дисциплине: «Современные технологии разработки веб-приложений»

(09.03.01 Прикладная математика и информатика)

на тему: Разработка веб-сервиса по планированию экономической деятельности в Югорском государственном университете

Группа: 1661м

Студент: Якимчук Александр Васильевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Руководитель: Гусаков Артем Владимирович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Отметка о защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

Ханты-Мансийск 2017 г.

Содержание

[Введение 3](#_Toc498249221)

[1. Методологии разработки 6](#_Toc498249222)

[2. Средства разработки 12](#_Toc498249223)

[2.1 Технологии разработки серверной части системы 12](#_Toc498249224)

[2.2 Технологии разработки клиентской части системы 16](#_Toc498249225)

[3. Описание системы 19](#_Toc498249226)

[3.1 Реализация модуля авторизации и ролевой политики 22](#_Toc498249227)

[3.2 Реализация модуля администрирования 25](#_Toc498249228)

[3.3 Реализация модуля «управленческий учет» 27](#_Toc498249229)

[3.4 Описание базы данных 30](#_Toc498249230)

[3.5 Описание интерфейса системы 30](#_Toc498249231)

[4. Тестирование 31](#_Toc498249232)

[4.1. Нагрузочное тестирование 31](#_Toc498249233)

[4.2 Функциональное тестирование 33](#_Toc498249234)

[Заключение 34](#_Toc498249235)

[Список использованных источников 35](#_Toc498249236)

[Приложение А 37](#_Toc498249237)

[Приложение Б 41](#_Toc498249238)

[Приложение В 42](#_Toc498249239)

# Введение

В настоящее время высшие учебные заведения становятся более независимы: ежегодно все меньшую долю в финансировании ВУЗа составляют бюджетные средства, одновременно растет объем внебюджетных образовательных услуг. В этих условиях возрастает роль управленческого учета, как в подразделениях вуза, так и в вузе в целом, а также повышаются требования к автоматизированным информационным системам управления финансово-экономической деятельностью образовательного учреждения [1].

Внедрение автоматизированных технологий обработки экономической информации позволяет повысить эффективность управления организацией за счет обеспечения руководителей и специалистов максимально полной, оперативной и достоверной информацией на основе единого источника данных. Уникальность структуры каждого отдельно взятого высшего учебного заведения не позволяют создать унифицированный инструмент. Это обусловлено, прежде всего, интенсивным развитием современных информационных технологий и отсутствием общепринятого подхода её создания. Таким образом, среди множества возможных вариантов реализации экономической системы необходимо выбрать наилучшее решение, учитывая при этом ограничения, накладываемые на экономические и технологические возможности образовательного учреждения [2].

В данном курсовом проекте рассматривается Югорский государственный университет как объект для разработки системы по ведению экономической деятельности.

В Югорском государственном университете внедрены такие системы, как:

* Корпоративная информационная система класса ERP Галактика 9.1. «Управление учебным процессом».
* Учет труда и заработной платы – АМБа;
* 1C Бухгалтерия;
* Многочисленные разработанные внутренние сервисы (Информационно-справочная система, Сервисы научной библиотеки и т.д.).

При этом отсутствует унифицированная система ведения экономической деятельности, которая бы интегрировала в себя необходимые данные с описанных сервисов и позволяла бы вести планирование, управление и анализ бюджета образовательного учреждения.

С каждым днем всё большее количество систем разрабатываются в виде веб-сервисов по той причине, что они имеют ряд преимуществ перед остальными. Во-первых, доступ к веб-сервису не требует установки на компьютер, достаточно лишь наличие подключения в интернет. Во-вторых, работать можно под любой операционной системой (Windows, MacOS, Linux и др.). Необходим только установленный браузер последних версий. В-третьих, веб-системы фактически не требовательны к ресурсам пользователей и не предъявляют никаких требований к аппаратной платформе пользователя.

Также веб-системы позволяют пользователям быть по-настоящему мобильными: они позволяют работать, сохранять результаты своей работы и, в случае необходимости, иметь к ним доступ с любого компьютера, где имеется доступ в Интернет.

В связи с этим можно сказать, что разрабатываемая система должна быть создана именно в виде веб-сервиса, что позволит вовлечь максимальное количество работников университета, так как каждое рабочее место в университете оборудовано скоростным интернетом и браузером.

Целью данного курсового проекта является разработка автоматизированной веб-системы, позволяющей вести планирование экономической деятельности в Югорском государственном университете с максимальной вовлеченностью всех специалистов и руководителей образовательного учреждения.

Для достижения данной цели необходимо решить следующие задачи:

* Определение методологии разработки;
* Определение паттерна для разработки системы;
* Выбор средств для разработки системы;
* Реализация модуля авторизации на основе службы для системного управления Active Directory;
* Реализация ролевой политики системы;
* Реализация модуля по управлению сервисами;
* Реализация модуля по планированию доходов образовательного учреждения;
* Реализация модуля по планированию расходов образовательного учреждения;
* Реализация модуля по установлению лимитов образовательного учреждения;
* Проведение тестирования с целью проверки соответствию требованиям.

# Методологии разработки

Методология разработки программного продукта – это процесс построения и развития системы. Выбор модели жизненного цикла оказывает влияние на качество разрабатываемой системы в целом [3]. В связи с этим необходимо проанализировать известные жизненные циклы и выбрать наиболее подходящий для разрабатываемой системы.

В силу отсутствия возможности внедрения гибких методологий разработки и специфики условий формирования требований на разработку систем в Югорском государственном университете решено использовать итеративную методологию разработки программного продукта.

В данной методологии разработка осуществляется как пошаговый процесс, последовательно приближая систему к её конечному виду. Данная методология уменьшает затраты за счёт поэтапного наращивания функциональных возможностей, при этом увеличивая производительность и функционал системы с каждой итерацией.



Рисунок 1.1 – Итеративная стратегия жизненного цикла программного продукта

Функционал системы будет изменяться и дополняться на каждой итерации, и выбранная методология для разработки системы по планированию экономической деятельности позволит уменьшить время разработки системы в целом, а также увеличить качество реализуемой системы. Процесс создания системы состоит из разработки отдельных независимых модулей, к которым по отдельности также применяется итеративная модель разработки.

Рассмотрим применение итеративной методологии разработки на основе реализации одного из модулей, а именно модуль «Планирование доходов».

Первым этапом необходимо совместно с начальником экономического управления (выступает в роли заказчика модуля системы) собрать требования на разработку системы по планированию доходов на учебный год.

После проведения этапа сбора требований – провести их анализ и составить функциональные (определяют функциональность системы) требования к разрабатываемому модулю.

Функциональные требования:

* Пользователь должен иметь возможность создания нового заявления на доход;
* При заполнении нового дохода система должна предоставлять пользователю возможность выбора центра финансовой ответственности, если роль пользователя позволяет выполнить данную возможность;
* При заполнении нового дохода Пользователь должен иметь возможность указания основных аналитик (КФО, направление деятельности, вид деятельности и ОКВЭД);
* При заполнении нового дохода система должна предоставлять пользователю возможность заполнения дополнительных аналитик только после заполнения основных аналитик;
* При заполнении нового дохода пользователь должен иметь возможность заполнения дополнительных аналитик;
* При заполнении нового дохода пользователь должен иметь возможность заполнения спецификации;
* При заполнении нового дохода пользователь должен иметь возможность указать наименование заявления и период;
* После полного заполнения нового дохода пользователь должен иметь возможность сохранить заявление;
* При сохранении нового дохода, в случае ошибок, система должна указать пользователю, почему невозможно сохранение, при этом заполненная информация должна сохраниться;
* Система должна обеспечивать корректное сохранение заполненного заявления в базу данных;
* Пользователь должен иметь возможность просмотра заполненных заявлений;
* Пользователь должен иметь возможность вывода на печать заполненных заявлений;
* Пользователь, имеющий роль администратора, должен иметь возможность «утверждения» и «разутверждения» отдельно взятых заявлений;
* Пользователь должен иметь возможность редактирования неутвержденных заявлений, при этом, если заявление утверждено, пользователь не должен иметь возможность редактирования заполненного заявления;
* Пользователь должен иметь возможность просмотра свода заполненных заявлений по статьям;
* При просмотре свода заполненных заявлений по статьям система должна предоставлять возможность изменения фильтров, а именно указание статуса заявлений и изменение структурных подразделений;
* Система должна предоставлять пользователям инструкцию по заполнению нового заявления на доход.

После описания и согласования требований было принято, что в первой итерации необходимо реализовать возможность создания нового заявления на доход и возможность просмотра заполненных заявлений. Тестирование разработанного функционала не выявило ошибок. По окончании первой итерации реализованный функционал модуля продемонстрирован и выпущен в опытную эксплуатацию, а также внесены некоторые дополнения к следующей итерации:

* Пользователь должен иметь возможность указать, является ли данное заявление плановым или нет;
* Пользователь должен иметь возможность указания года планирования при создании нового заявления на доход;
* При просмотре заполненных заявлений у пользователя должна быть возможность фильтрации заполненных заявлений по годам планирования.

После внесения изменений была проведена вторая итерация, в ходе которой реализованы дополнения, указанные после первой итерации, а также реализован свод по статьям дохода и написана инструкция по заполнению.

В дальнейшем было также проведено тестирование функционала: ошибок в работе модуля не было обнаружено. В конечном итоге модуль продемонстрирован заказчику и выпущен в опытную эксплуатацию.

Таким образом, в ходе разработки данного модуля проведено две итерации, каждая из которых длилась 2-3 недели, при этом в ходе каждой были проведены стандартные этапы разработки.

Помимо выбора методологии разработки программного продукта немаловажным для качества разрабатываемой системы является выбор шаблона проектирования (паттерн). Под шаблоном проектирования чаще всего понимают проверенные и готовые к использованию решения часто возникающих в повседневном программировании задач. Шаблоны позволяют сделать код более понятным и легко используемым.

В данном курсовом проекте выбор шаблона проектирования обусловлен выбором технологии разработки, в качестве которой используется язык программирования PHP с фреймворком Phalcon (информация о выборе данных технологий разработки представлена во второй главе), поэтому в качестве архитектурного шаблона разрабатываемой системы будет применяться архитектурный шаблон Model-View-Controller (MVC) [4].

Шаблон MVC описывает простой способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса. В результате чего, система легче реализуется, масштабируется, и сопровождается.

Рассмотрим концептуальную схему шаблона MVC (представлена на рисунке 1.2)

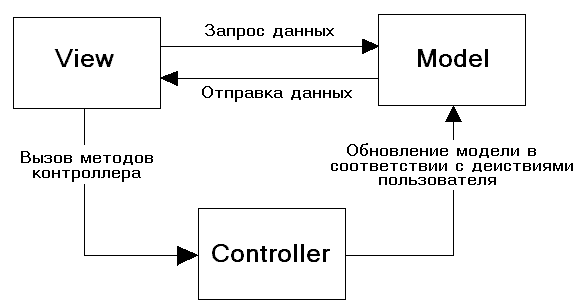


Рисунок 1.2 – Схема шаблона MVC

Модель (Model) содержит бизнес-логику приложения и включает методы выборки (это могут быть методы ORM), обработки (правила валидации и пр.) и предоставления конкретных данных.

Представление (View) отвечает за пользовательский интерфейс приложения. Чаще всего это HTML файлы исключительно для вывода данных. Этот слой отвечает за вывод данных в веб-браузер или другой инструмент, который обращается к Вашему приложению.

Контроллеры (Controller) обеспечивают обмен данными между моделью и представлением. Контроллеры ответственны за обработку входящих запросов от веб-браузера, запрашивание данных у модели и передачу этих данных в представление для вывода.

В разрабатываемой системе каждый отдельный модуль разрабатывается по шаблону проектирования MVC.

# Средства разработки

В настоящее время существует большое количество языков программирования для разработки систем, поэтому выбор языка обусловлен поставленными задачами. Система разрабатывается в виде веб-системы, в связи с этим необходимо определить технологии разработки для клиентской и для серверной части, которые позволят успешно реализовать поставленные задачи с наименьшими трудозатратами.

## 2.1 Технологии разработки серверной части системы

Серверная часть системы отвечает за бизнес-логику, взаимодействие с базами данных и производительность. Для разработки серверной части системы необходимо выбрать серверный язык программирования и базу данных.

Рассмотрим несколько наиболее популярных в настоящее время серверных языков программирования (ЯП): Ruby, Python, PHP, C#, Java, NodeJS. Проведём сравнительный анализ с целью определения наиболее эффективного языка для решения поставленных задач. Сравнительный анализ по ряду критериев представлен в таблице 2.1.

Таблица 2.1 – Сравнительный анализ серверных языков программирования

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Критерий** | **Ruby** | **Python** | **PHP** | **C#** | **Java** | **NodeJS** |
| **Стоимость** | Бесплатно | Бесплатно | Бесплатно | Бесплатно | Бесплатно | Бесплатно |
| **Отказоустойчивый ввод/вывод** | + | + | + | + | + | + |
| **Кроссплатформенность** | + | + | + | + | + | + |
| **Документация и литература** | 7 | 8 | 9 | 9 | 9 | 10 |
| **Безопасность** | 8 | 8 | 7 | 9 | 9 | 7 |
| **Популярность** | 7 | 10 | 10 | 10 | 10 | 10 |
| **Объектно-ориентированные возможности** | 10 | 9 | 8 | 10 | 10 | 6 |
| **Возможности функциональной разработки** | 4 | 7 | 7 | 5 | 7 | 9 |
| **Типизация** | Динамическая | Динамическая | Динамическая | Статическая | Статическая | Динамическая |
| **Стандартизация** | ISO/IEC 30170 | PEP | RFC | ISO | JLS | ECMA |
| **Скорость разработки** | 8 | 9 | 9 | 8 | 8 | 9 |
| **Скорость работы** | 8 | 8 | 7 | 8 | 8 | 8 |
| **Наличие современных фреймворков** | 8 | 10 | 10 | 10 | 10 | 10 |
| **Легкость в освоении** | 8 | 9 | 9 | 8 | 8 | 9 |
| **Официальный сайт** | <https://www.ruby-lang.org/ru/> | <https://www.python.org/> | <https://www.php.net/> | <https://www.visualstudio.com/ru/> | <https://www.java.com/ru/> | <https://nodejs.org/en/> |

Сравнительный анализ языков программирования проведен на основе конкретных критериев, которые были проанализированы с помощью официальных сайтов, форумов и блогов разработчиков [5],[6]. Сравнение языков показало, что каждый язык подходит для конкретной области и задачи разработки.

Основополагающим аспектом выбора языка программирования для реализации системы стали такие аспекты, как скорость разработки, легкость в освоении и наличие современных фреймворков. Таким образом, для поставленных задач лучше всего подходит язык программирования PHP.

Следующий этап работы заключается в выборе фреймворка (каркас построения систем) для языка программирования PHP, который позволит определить структуру системы и облегчить написание кода [7]. В качестве рассматриваемых фреймворков выбраны следующие наиболее популярные фреймворки: Laravel, Symfony, Yii2, Phalcon, CodeIgniter. Сравнительный анализ представлен в таблице 2.2.

Таблица 2.2. – Сравнительный анализ PHP фреймворков.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Критерий** | **Laravel** | **Symfony** | **Yii2** | **Phalcon** | **CodeIgniter** |
| Последняя версия | 5.5.19 | 3.3.9 | 2.0.12 | 3.2.2 | 3.1.4 |
| Дата последнего обновления | 18.10.2017 | 11.09.2017 | 06.06.2017 | 10.09.2017 | 20.03.2017 |
| Минимальная версия PHP | 5.6.3 | 5.5.9 | 5.4.0 | 5.6.1 | 5.5.2 |
| Поддержка и сообщество | 9 | 7 | 8 | 6 | 5 |
| Поддержка MVC | + | + | + | + | + |
| Наличие ORM | + | + | + | + | + |
| Возможность проведения Unit тестирования | + | + | + | + | + |
| Безопасность | 9 | 9 | 7 | 9 | 6 |
| Наличие шаблонизатора | + | + | + | + | - |
| Возможность кэширования  данных | + | + | + | + | - |
| Поддержка Ajax | + | + | + | + | + |
| REST | + | + | + | + | + |
| Автозагрузчик | + | + | + | + | - |
| Роутинг | + | + | + | + | + |
| Кэширование | + | + | - | + | - |
| Производительность | 8 | 8 | 7 | 10 | 5 |

В результате проведенного сравнительного анализа решено использовать фреймворк Phalcon, по той причине, что он имеет наилучшую производительность, т.к. реализован в виде расширения на Zephir/C и загружается вместе с PHP один раз при запуске демона веб-сервера. Благодаря низко-уровненной архитектуре и оптимизации Phalcon обеспечивает наименьшую нагрузку при работе MVC-приложения [8]. Использование данного фреймворка позволит также уменьшить нагрузку на память и процессор сервера.

В качестве системы управления базами данных был выбран Oracle, так как, во-первых, он является основной базой данных в информационной среде университета, а во-вторых, переход на другую БД может вызвать дополнительные трудности при разработке системы.

К преимуществам данной СУБД можно отнести производительность, быстроту, защищенность и самоуправление (механизмы Oracle позволяют самостоятельно перераспределять нагрузку на систему, оптимизировать и корректировать SQL-запросы).

## 2.2 Технологии разработки клиентской части системы

Для разработки клиентской части веб-системы в настоящее время применяется стандартный стек технологий HTML, CSS и JavaScript.

HTML (Hyper Text Markup Language) – это язык гипертекстовой разметки страницы, стандарт разметки страниц для Интернета. Считывая данный язык, браузеры интерпретируют его и отображают уже как сформированный документ [9].

CSS (Cascading Style Sheets) – это каскадные таблицы стилей, которые используются для описания внешнего вида документа, написанного языком разметки [10].

Применение определенного CSS фреймворка позволит создать единый дизайн системы, увеличить скорость разработки, за счет того, что для большей части элементов не надо описывать каждый элемент самостоятельно, а можно применять уже готовые стили. Ещё одним преимуществом является кроссбраузерность, которую способны предоставить CSS фреймворки.

В качестве CSS фреймворка был выбран Foundation 6 по следующим причинам:

* Имеет строгий стиль;
* Обладает хорошей документацией, благодаря чему легко описывать любой элемент системы;
* Возможно использовать стили только для определенных элементов, что позволяет уменьшить размер конечного исходного файла системы;
* Помимо современных браузеров имеется поддержка Internet Explorer 8 версии и выше;
* Имеются готовые шаблоны, которые позволяют быстро реализовать основной интерфейс системы.

Данные критерии стали ключевыми в пользу выбора данного CSS фреймворка.

JavaScript – это прототипно-ориентированный, мультипарадигменный язык с динамической типизацией, который поддерживает объектно-ориентированный, императивный и декларативный (например, функциональное программирование) стили программирования. JavaScript запускается на стороне клиента и может использоваться для создания/программирования того, как веб-страницы будут вести себя при наступлении каких-либо событий [11].

Стандартом языка JavaScript является ECMAScript. По состоянию на 2012, все современные браузеры полностью поддерживают ECMAScript 5.1.

Использование сторонних фреймворков позволяет существенно увеличить скорость разработки клиентской части, поскольку в них уже реализован базовый функционал. Однако это негативно сказывается на размере приложения, потому что большая часть функционала и реализованной логики, поставляемые по умолчанию с этими инструментами, могут вообще не использоваться в приложении.

Таким образом, принято решение использовать минимальный набор библиотек для языка JavaScript, которые позволят увеличить скорость разработки системы и будут использованы в каждом из реализуемых модулей. Выбраны следующие библиотеки:

* JQuery: помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими [12].
* MomentJS: позволяет удобно работать с временем [13];
* MaskJS: позволяет накладывать на поля ввода определенные «маски» для ввода [14];
* Foundation-datepicker: позволяет добавить календарь, который может быть использован для выбора и отображения дат [15].

# Описание системы

Как упоминалось ранее в работе, система разрабатывается как автоматизированная веб-система, позволяющая вести планирование экономической деятельности.

В первую очередь, необходимо реализовать каркас серверной части системы, используя выбранный фреймворк Phalcon. Для увеличения скорости разработки и качества системы в документации имеется пример построения файловых структур MVC. Наиболее подходящей файловой структурой является мульти-модульная MVC структура, которая позволяет разбить все части системы на отдельные модули [16]. В каждом из модулей применяется своя MVC структура, при этом представлено общее представление для каждой структуры, а также глобальные зависимости, подключенные в каждом модуле.

Первый файл, который необходимо реализовать — это скрипт инициализации системы. Данный файл реализует инициализацию компонентов и служит основой приложения, предоставляя контроль над всем его поведением. Листинг данного файла представлен на рисунке 3.1



Рисунок 3.1 – листинг реализации файла начальной загрузки

Рассмотрим подробнее основные методы, используемые в данном файле:

* Loader – используется для загрузки пространства имен и классов, которые будут доступны для каждого модуля;
* Autoloader – отвечает за инициализацию сторонних компонентов, которые были загружены с помощью пакетного менеджера Composer;
* Modules – используется для инициализации пространства имен и расположения реализуемых модулей.
* Services – отвечает за подключения глобальных зависимостей системы.

Целесообразно рассмотреть настройки данного файла подробнее.

В файле Services описывается концепция внедрения зависимостей (Dependency injection) [17]. В файле происходит регистрация большинства стандартных компонентов фреймворка. Таким образом, нет необходимости регистрировать один за другим компонент, а при необходимости их можно заменить собственной реализацией. В данном файле происходит подключение таких компонентов, как:

* Router – данный компонент позволяет определить маршруты, которые сопоставляются с контроллерами или обработчиками, которые должны получить запрос. Маршрутизатор анализирует URI для определения этой информации;
* View – инициализация мульти модульного представления, которое будет подключено во всех модулях;
* Dispatcher – компонент, ответственный за создание экземпляров контроллеров и выполнение необходимых действий над ними в системе, например, проверка прав на использование контроллера;
* DB – позволяет использовать подключение к настроенной схеме базы данных;
* Session – компонент для регистрации возможности работы с сессиями в системе;
* Breadcrumbs – настройка компонентов навигационной цепочки.

Таким образом, создано ядро для работы с системой. В дальнейшем данные файлы будут дополняться, если необходимо будет использовать дополнительные возможности.

После начальной настройки серверной стороны создадим файл-представление (View), который подключается во все модули. Общий файл-представление создаётся для того, чтобы задать общий стиль системы и подключить необходимые общие JavaScript и CSS файлы.

Phalcon предоставляет свой шаблонизатор для представлений – Volt. Данный шаблонизатор является быстрым, по той причине, что он реализован на C для PHP. Volt тесно связан с компонентами Phalcon. Все представления Volt после их реализации и первого запуска компилируются в чистый PHP.

В общем файле-представлении настроим все необходимые метатеги, реализуем область для шапки и подвала, а также для основного содержания системы, далее подключим общие библиотеки для работы с CSS и JavaScript, а также реализуем возможность подключения дополнительных файлов для CSS и JavaScript прямо из контроллера. Данная возможность будет полезна, когда каждому модулю необходимо будет реализовать свои стили и поведение. Листинг реализации данного файла представлен в приложении А на рисунке А.1.

Все JavaScript, CSS, изображения и шрифты, как и в большинстве веб-систем будут находиться в директории public с соответствующим наименованием директорий.

После настройки общего каркаса системы необходимо реализовать ряд модулей и компонентов, которые будут соответствовать поставленным задачам. Разработка системы будет разделена на следующие этапы (каждый из которых будет описан более подробно):

* Реализация модуля авторизации и ролевой политики;
* Реализация модуля администрирования;
* Реализация модуля управленческий учет, включающего в себя планирование доходов и расходов, фактическое исполнение.

## Реализация модуля авторизации и ролевой политики

В Югорском государственном университете в качестве централизованной системы групповых политик используется служба каталогов Windows Active Directory [18]. Каждый сотрудник университета имеет корпоративный логин и пароль, поэтому необходимо реализовать авторизацию с помощью LDAP (облегченный протокол доступа к каталогам). Реализация авторизации в виде use case диаграммы представлена на рисунке 3.1.

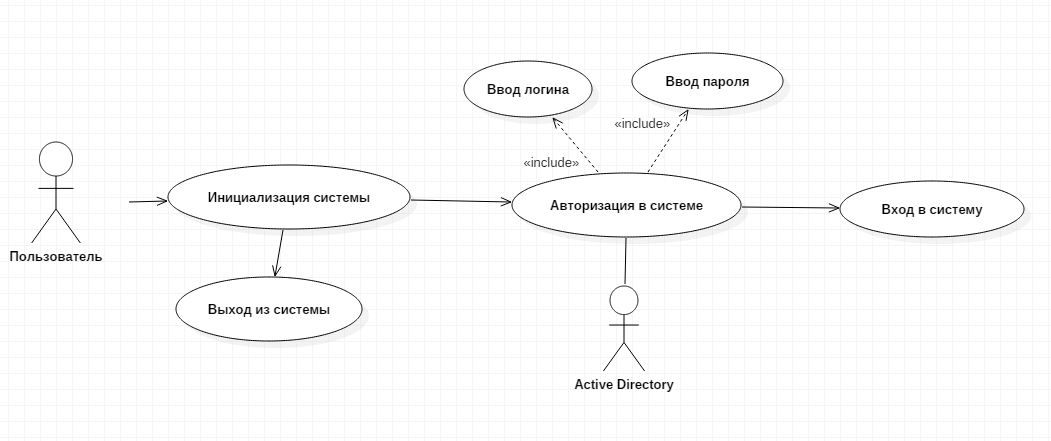


Рисунок 3.1 – Use Case Авторизация пользователя

Реализация модуля авторизации была вынесена в отдельный независимый класс Authorization и была подключена как глобальная зависимость в Services, отвечающая за внедрение зависимостей.

В первую очередь, в данном файле реализованы параметры подключения к LDAP серверу, полученные у системного администратора университета. Следующим этапом был реализован метод, позволяющий очистить логин пользователя от лишней информации, чтобы была возможность приводить все логины к едином виду (Например, логин A\_Yakimchuk@ugrasu.ru, привести к виду A\_Yakimchuk). Далее реализованы методы для проверки соединения с LDAP сервером и проверки, что данный пользователь существует в каталоге. В положительном случае данный метод «возвращал» Ф.И.О. пользователя и отдел, в котором он работает. Листинг реализации класса авторизации представлен в приложении А на рисунке А.2.

Основная часть информации о работниках университета хранится в ERP «Галактика» в модуле «Кадровый учёт». Соответственно, для получения полной информации о сотрудниках и отделах университета после авторизации нужно реализовать взаимодействие с данным модулем.

Взаимодействие реализовано посредством базы данных. ERP «Галактика» в качестве хранилища данных использует систему управления базами данных Oracle, которая находится на сервере университета. Данные «Галактика» хранятся в схеме «Galaserv».

Для реализуемой системы на сервере университета базы данных Oracle была создана схема «Finance». В данной схеме реализована вся логика хранения данных.

Из схемы «Galaserv» выданы права на использование таблиц-синонимов (доступные только для чтения) в схему «Finance». Таким образом, получены все данные, которые имеются в ERP «Галактика» по сотрудникам, отделам и др. Реализация взаимодействия реализуемой системы с «Галактика» в виде диаграммы представлена на рисунке 3.2.

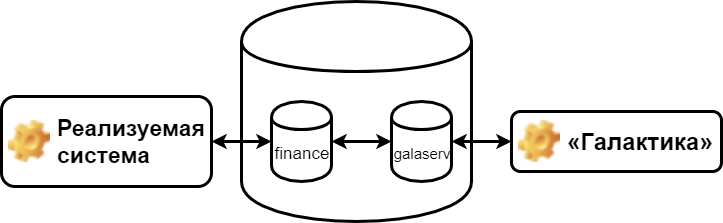


Рисунок 3.2 – Схема взаимодействия реализуемой системы с «Галактика»

После реализации взаимодействия с ERP «Галактика» в базе данных в рабочей схеме finance реализовано представление «v\_persons» на основе полученных таблиц, которое позволяло получить уникальный номер сотрудника, его отдел, должность и другую информацию, касающуюся его деятельности в университете. Таким образом, зная Ф.И.О. и отдел сотрудника после авторизации и подставляя эти данные в представление, получаем все необходимые данные по сотруднику.

Югорский государственный университет имеет сложную организационную структуру, поэтому вместе с авторизацией необходимо также реализовать механизм ролевой политики в системе, который позволял бы разграничить пользователей по ролям согласно их занимаемой должности. Для этого выделены следующие роли:

* Супервизор – пользователь с данной ролью обладает правами администратора ко всем реализуемым модулям;
* Администратор – пользователь с данной ролью обладает правами администратора к определенному модулю;
* Ректор – пользователь с данной ролью обладает правами, которые позволяют получить полный доступ ко всем отчётам в реализуемой системе, а также возможность редактирования данных подчиненных подразделений;
* Проректор – пользователь с данной ролью обладает правами, которые позволяют получить доступ только к данным всех подчиненных подразделений;
* Директор – пользователь с данной ролью обладает правами, которые позволяют получить доступ только к данным подчиненного подразделения;
* Заместитель – пользователь с данной ролью обладает такими же правами, как и директор;
* Сотрудник – пользователь обладает минимальным набором прав: может вести планирование и редактирование только тех заявлений, которые сделал сам.

Стоит также отметить, что каждый пользователь может иметь несколько ролей в системе, т.к. каждый сотрудник может занимать несколько должностей, например, сотрудник может работать преподавателем и начальником отдела, следовательно, пользователь должен иметь возможность переключения между ролями.

Для реализации такой ролевой политики реализовано сопоставление ролей в базе данных в уже описанном представлении v\_persons. Таким образом, каждый пользователь получил роль, которая соответствует его должности. Листинг реализации представления v\_persons представлен в приложении А на рисунке А.3.

Также необходимо реализовать список управления доступом (ACL) с помощью имеющихся в фреймворке Phalcon средств. В Phalcon имеется класс ACL, который предоставляет простое и легкое управление списками контроля доступа, а также разрешениями, назначаемыми этим спискам. Списки управления доступом позволяют приложению управлять доступом к различным своим частям и запрошенным объектам.

Для каждой роли в отдельности описываются контроллер и метод, к которому необходимо дать доступ. Данный способ затратный по времени при разработке, но при этом более безопасен.

Список управления доступом реализован как промежуточное звено (middleware) таким образом, что для каждой роли в системе при переходе между модулями системы проверяется, имеется ли доступ к данному контроллеру и методу у данной роли. Если нет, то в доступе отказано, либо же пользователь получает доступ.

## 3.2 Реализация модуля администрирования

В реализуемой системе модуль администрирования позволяет выполнить следующий функционал в виде подмодулей:

* Управление сервисами системы;
* Управление ролевой политикой системы;
* Контроль делегирования полномочий;
* Подмена пользователя.

Рассмотрим реализацию и описание каждого из подмодулей. Доступом к данному модулю обладают только администраторы системы.

Управление сервисами системы позволяет гибко настраивать модули системы. В данном модуле не реализуется создание файлов MVC структуры, а только идет привязка определенного модуля в системе к определенному контроллеру. Для каждого модуля имеется возможность добавлять новые сервисы в систему для контроля доступа к ним, назначать роли, которые будут иметь доступ к модулю, назначать администраторов, а также зменять наименование и картинку для отображений в интерфейсе системы.

Управление ролевой политикой системы позволяет добавлять или удалять роли пользователей, таким образом, чтобы любому пользователю можно было назначить любой отдел и роль. Данный функционал реализован для того, чтобы можно было гибко менять роли пользователей по причине частой необходимости в такой возможности.

Контроль делегирования полномочий позволяет отслеживать ролевую политику каждого подразделения в университете, что позволяет увидеть, какой пользователь, за какое подразделение отвечает, и кто в этом подразделении является проректором.

Данный подмодуль позволяет пользователям с ролью «Супервизор» входить в систему под любым из работников университета. Данный функционал является очень полезным, когда у пользователей возникают вопросы, касающиеся системы.

Общая диаграмма use case для модуля администрирования представлена на рисунке 3.2.1.

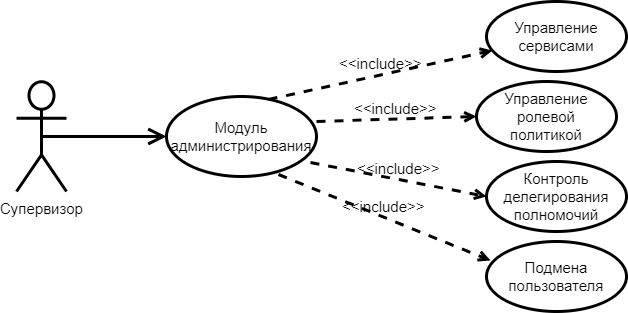


Рисунок 3.2.1 – Диаграмма Use Case для модуля администрирования

## 3.3 Реализация модуля «управленческий учет»

Модуль «управленческий учет» является основным в описываемой системе, т.к. разработка системы ведется именно для планирования экономической деятельности университета.

Разработку модуля «управленческий учет» можно разбить на части, представленные на диаграмме рисунка 3.3.1.

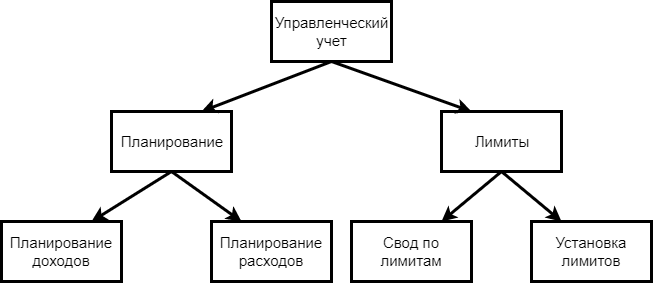


Рисунок 3.3.1 – Диаграмма модуля управленческий учет

Рассмотрим разработку подмодуля планирования с реализацией планирования доходов. Данный модуль включает в себя части, представленные на рисунке 3.3.2.

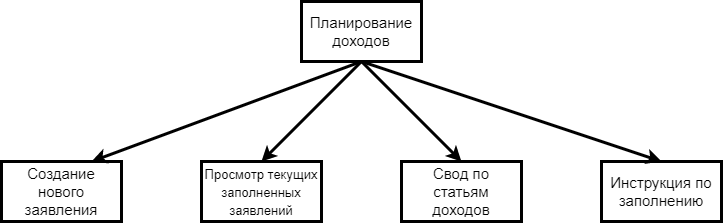


Рисунок 3.3.2 – Диаграмма модуля планирование доходов

Более подробно рассмотрим разработку каждого из модулей планирования доходов.

Для реализации данных модулей создадим единую модель (Income.php) и контроллер (IncomeController.php), которые будут использоваться реализуемыми модулями как общие. С помощью уже описанного модуля «управление сервисами» создадим иерархию модулей, описанную на рисунке 3.3.1 и 3.3.2. В качестве пользователей модулей добавим все роли, т.к. каждый сотрудник может работать с управленческим учетом, но при этом будут наложены ограничения согласно ролевой политике.

Перейдем непосредственно к созданию модуля «Создание нового заявления». В первую очередь, необходимо реализовать метод в контроллере (create\_statementAction), отвечающий за создание нового заявления на доход, в котором будут подгружаться необходимые начальные данные из модели, а также подключаться CSS и JavaScript файлы.

Далее создадим представление, которое будет находиться в папке income (соответствует наименованию контроллера) с файлом create\_statement, который соответствует методу контроллера, описанного выше. Такая концепция реализована во многих современных фреймворках по следующему алгоритму: как только определённый контроллер завершает свою работу, Phalcon автоматически передаёт управление компоненту представлений. Данный компонент ищет в папке представлений название, которое совпадает с последним исполнившимся контроллером, а затем – файл, имя которого соответствует последнему выполненному действию (action) [19].

Для хранения информации в базе данных создадим таблицу с наименованием «uu\_request», в которой будут храниться все основные данные по заполняемому заявлению, а также таблицу «uu\_request\_specification», в которой будут храниться данные, касающиеся спецификации заявления на доход.

После создания таблиц, представления и метода реализуем основную логику работы модуля. При входе в данный модуль реализована возможность заполнения основных сведений и аналитик по доходу. После этого с помощью JavaScript и технологии AJAX в зависимости от выбранных аналитик загружается вторая часть формы, в которой реализована возможность заполнения дополнительных аналитик, спецификаций для данного заявления, а также дата планирования и наименования заявления. При сохранении заявления данные посылаются POST запросом на ещё один реализованный в контроллере метод «save\_analiticsAction», который принимает их и вызывает объект класса «Income» c методом «saveRequestAnalytics», в который эти данные передаются и в случае успешной валидации сохраняются. Таким образом, реализуется создание нового заявления на доход.

Перейдем к разработке модуля просмотра заполненных заявлений на доход. Данный модуль также реализуется в контроллере IncomeController и модели Income.

В настоящем модуле необходимо реализовать возможность просмотра заполненных заявлений по ролям, таким образом, чтобы работник видел только заявления, заполненные непосредственно им самим, начальник работника – заявления всех своих подчиненных, проректор – все заявления подчиненных ему подразделений, а ректор и администратор – все заполненные заявления. При создании нового заявления данные пользователя также сохраняются.

Так же, как и для остальных методов создан метод в контроллере, отвечающий за вывод заполненных заявлений и представление с одноименным наименованием «history\_current\_statement». Вывод всех заполненных заявлений реализован с помощью представления базы данных. Листинг реализованного представления представлен в приложении А на рисунке А.4.

В модуле «просмотра заполненных заявлений» реализована возможность краткого просмотра заявлений и возможность детализации, а также реализован функционал редактирования, удаления и печати заявления. Для администратора модуля «управленческий учет» реализован функционал утверждения заявления, после которого запрещается удаление и редактирование заявлений.

Рассмотрим реализацию модуля «свод по статьям дохода». В данном модуле реализована аналогичная ролевая политика, как и в ранее описанном модуле «просмотр заполненных заявлений». Данный модуль позволяет просматривать заполненные заявления на доход, но в разрезе четырех основных кодов финансового обеспечения (КФО) и статей доходов. Для данного модуля реализована возможность детализации каждого разреза для просмотра помесячного план-графика и центра финансового обеспечения, запланировавшего данный доход.

В модуле «инструкция по заполнению» реализован функционал скачивания файла, в котором описывается процесс работы с модулем.

Модули «планирование расходов» и «лимиты» реализованы аналогично модулю «планирование доходов».

## 3.4 Описание базы данных

Для реализуемой системы создана схема Finance, в которой реализуются необходимые таблицы, представления, последовательности, триггеры и пр.

Для хранения информации созданы таблицы, соответствующие логики каждого модуля. Для вывода информации в основном используются представления, так как они упрощают процесс получения данных из сложной выборки.

Схема базы данных реализованной системы представлена в приложении Б.

## 3.5 Описание интерфейса системы

Пользовательский интерфейс имеет большое значение для любой программной системы и является неотъемлемой ее составляющей, ориентированной, прежде всего, на конечного пользователя. Именно через интерфейс пользователь судит о прикладной программе в целом.

Интерфейс реализован в едином стиле благодаря выбранному CSS фреймворку. Интерфейс модулей разработанной системы представлен в приложении В.

# Тестирование

Процесс тестирования и отладки программного обеспечения является неотъемлемой частью процесса создания программного обеспечения в целом и проводится с целью улучшения качества разрабатываемого продукта. Во время разработки программного обеспечения при написании кода могут быть допущены определенные ошибки, в дальнейшем влияющие на логику самой программы [20]. Для выявления ошибок и отладки программы проведено нагрузочное и функциональное тестирование.

## 4.1. Нагрузочное тестирование

В широком понимании нагрузочное тестирование – это автоматизированное тестирование, которое проводится с помощью специального инструментального средства, имитирующего работу определенного количества пользователей системы за единицу времени. Итоговые показатели проведённого тестирования зависят от следующих факторов:

1. Качество разработанной системы;

2. Производительность сервера и настройка сервера, на котором работает система;

3. Количество пользователей, использующих систему за определенный промежуток времени.

Нагрузочное тестирование проводится с помощью инструмента Apache JMeter, который позволяет создавать бесконечное число потоков и производить эмуляцию действий пользователя в системе.

Нагрузочное тестирование проводится на отдельные скрипты с имитацией 100 потоков, последовательно обращающихся к системе и осуществляющих взаимодействие с базой данных, а именно тестирование на чтение и запись базы данных. Тем самым, это позволит проверить систему на стабильность и производительность.

Apache JMeter генерирует входные данные, которыми являются GET или POST-запросы, с помощью которых происходит взаимодействие с базой данных.

Выходными данными является отработанный ответ от сервера с полученными данными и время действия. Если от сервера приходит ошибка или пустой ответ, то система на каком-то этапе не смогла обработать необходимый запрос.

Результаты нагрузочного тестирования представлены на рисунке 4.1 в виде графика

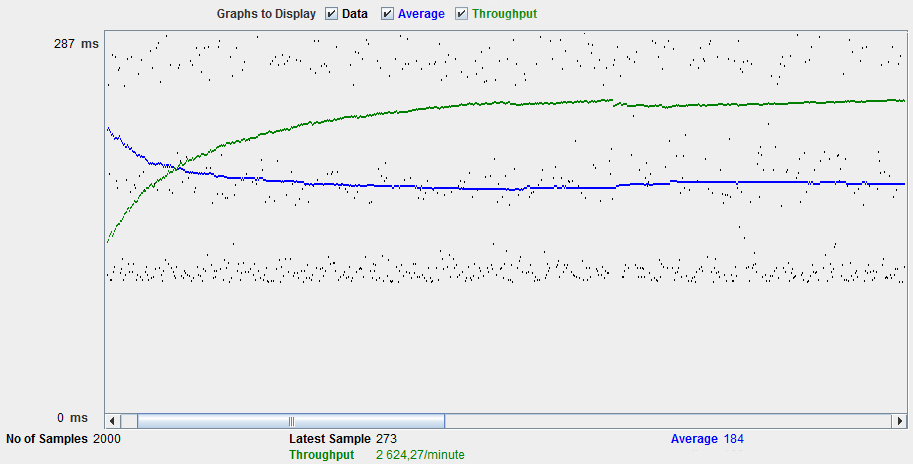


Рисунок 4.1 – График результата проведённого нагрузочного тестирования

Все значения на графике представлены в миллисекундах. Зелёная линия (Throughput) на графике означает пропускную способность выполняемых запросов. Синяя линяя (Average) – усредненное время отклика, график изменения нагрузки для каждого создаваемого потока. Черные точки на графике (Data) – время отклика каждой отдельной единицы данных, т.е. каждого проверенного URL.

Нагрузочное тестирование показало, что среднее время отклика системы не превышает 300 миллисекунд, что является хорошим результатом.

## 4.2 Функциональное тестирование

Функциональное тестирование проводилось с помощью инструмента Selenium [21]. Selenium – это инструмент для автоматизации действия веб-браузера. Данный вид тестирования проводится в основном на этапе разработки системы, что позволяет проверить функционал системы при изменении кода.

Для написания тестов на Selenium необходимо лишь установить его как плагин и далее записать скрипт, который будет повторять записанные действия. Пример такого скрипта для создания нового заявления на расход представлен на рисунке 4.2.1.

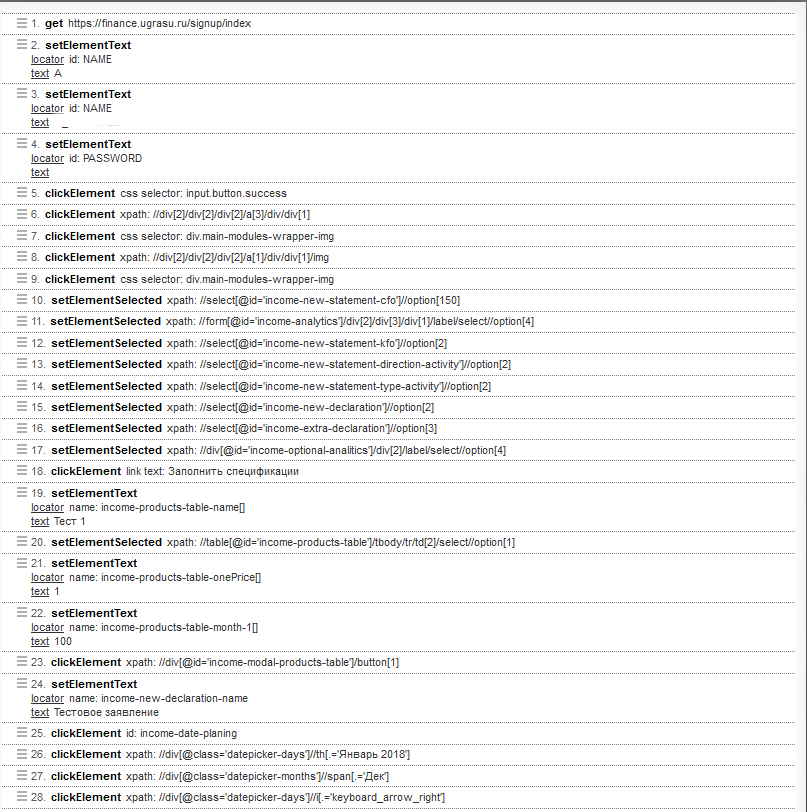


Рисунок 4.2.1 – Тестовый скрипт для заполнения нового дохода

# Заключение

В ходе выполнения курсового проекта получены следующие результаты:

1. Проведено определение паттерна и методологии для разработки системы;
2. Проведен выбор средств для реализации системы;
3. Реализован модуль авторизации и ролевая политика в системе;
4. Реализован модуль администрирования системы;
5. Реализован модуль управленческого учета, позволяющий вести планирование доходов и расходов образовательного учреждения;
6. Проведено нагрузочное тестирование;

Таким образом, разработана, протестирована и внедрена система по планированию экономической деятельности в Югорском государственном университете.

Данный проект позволяет работникам университета вести планирование экономической деятельности в веб-системе.

# Список использованных источников

1. Беляев Д.А. Применение информационных систем управления как необходимый элемент стратегического управления экономическими процессами в вузе // Университетское управление. – 2004. – №5-6 (33). – С. 179 – 183.
2. Григорьева М.В. Информационное обеспечение автоматизированной системы управленческого учета подразделения вуза // Доклады ТУСУРа. – 2004. – №2. – С. 19 – 25.
3. Орлов С.А., Цилькер Б.Я. Технологии разработки программного обеспечения: Учеб пособие. — 4-е изд. – СПб.: Питер, 2012. — 608 с.
4. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Гамма Э., Хелм Р., Джонсон Р. и др. – СПб.: Питер, 2016. – 366 с.
5. Comparison of programming languages / Wikipedia, The Free Encyclopedia. – Access mode: [https://en.wikipedia.org/w/index.php?title=Comparison\_of\_programming\_languages&oldid=803183776 16.09.2017].
6. Деревянченко М. Самые востребованные языки программирования. Языки программирования для начинающих // FB.ru – 2016. Режим доступа: [http://fb.ru/article/262355/samyie-vostrebovannyie-yazyiki-programmirovaniya-yazyiki-programmirovaniya-dlya-nachinayuschih 21.09.2017].
7. PHP frameworks comparison / Social Compare Beta. – Access mode: [http://socialcompare.com/it/comparison/php-frameworks-comparison 22.09.2017].
8. Полнофункциональный PHP-фреймворк, поставляемый как С-расширение / Phalcon.com. – Access mode: [https://phalconphp.com/ru/ 01.10.2017].
9. HTML developer guide / MDN web docs moz://a. – Access mode: [https://developer.mozilla.org/az/docs/Web/Guide/HTML 03.10.2017].
10. CSS / MDN web docs moz://a. – Access mode: [https://developer.mozilla.org/ru/docs/Web/CSS 03.10.2017].
11. JavaScript / MDN web docs moz://a. – Access mode: [https://developer.mozilla.org/ru/docs/Web/JavaScript 06.10.2017].
12. JQuery / jQuery.com. – Access mode: [https://jquery.com/ 06.10.17].
13. Moment.js 2.19.1 / momentjs.com. – Access mode: [http://momentjs.com/ 12.10.2017].
14. MaskJS / Github.com. – Access mode: [https://github.com/atmajs/MaskJS 12.10.2017].
15. Foundation-datepicker.js / Foundation-datepicker.js Examples & Docs. – Access mode: [http://foundation-datepicker.peterbeno.com/example.html 13.10.2017].
16. Phalcon MVC Examples / Github.com. – Access mode: [https://github.com/phalcon/mvc 16.10.2017].
17. Внедрение зависимости / Википедия, свободная энциклопедия. – Режим доступа: [https://ru.wikipedia.org/wiki/Внедрение\_зависимости 17.10.2017].
18. Active Directory / Википедия, свободная энциклопедия. – Режим доступа: [https://ru.wikipedia.org/wiki/Active\_Directory 22.10.2017].
19. Views / Phalcon.com. – Access mode: [https://docs.phalconphp.com/uk/3.2/views 25.10.2017].
20. Средства автоматизированного тестирования / Alpana Quality Services. – Режим доступа: [http://aplana.ru/infocenter/publications/2009/1952-sredstva-avtomatizirovannogo-testirovaniya 30.10.2017].
21. SeleniumHQ / SeleniumHQ Browser Automation. – Access mode: [http://docs.seleniumhq.org/ 30.10.2017].

# Приложение А

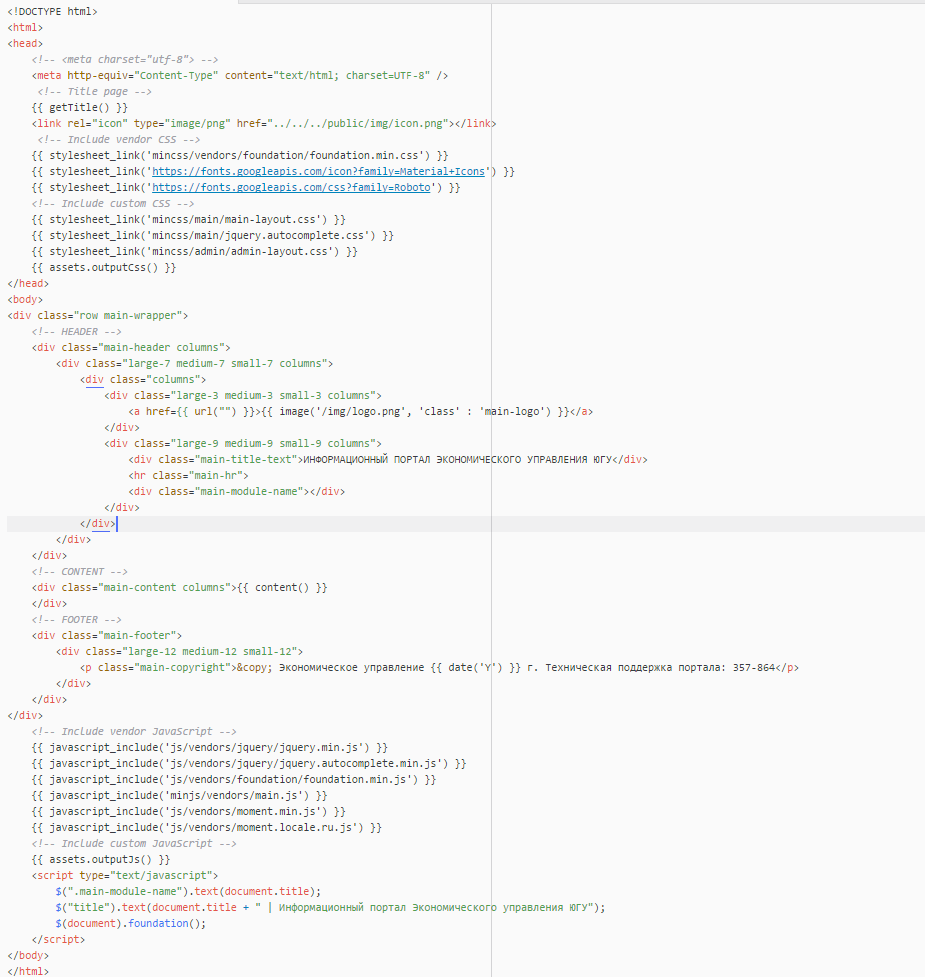


Рисунок А.1 – Реализация общего файла-представления



Рисунок А.2 – Реализация класса авторизации

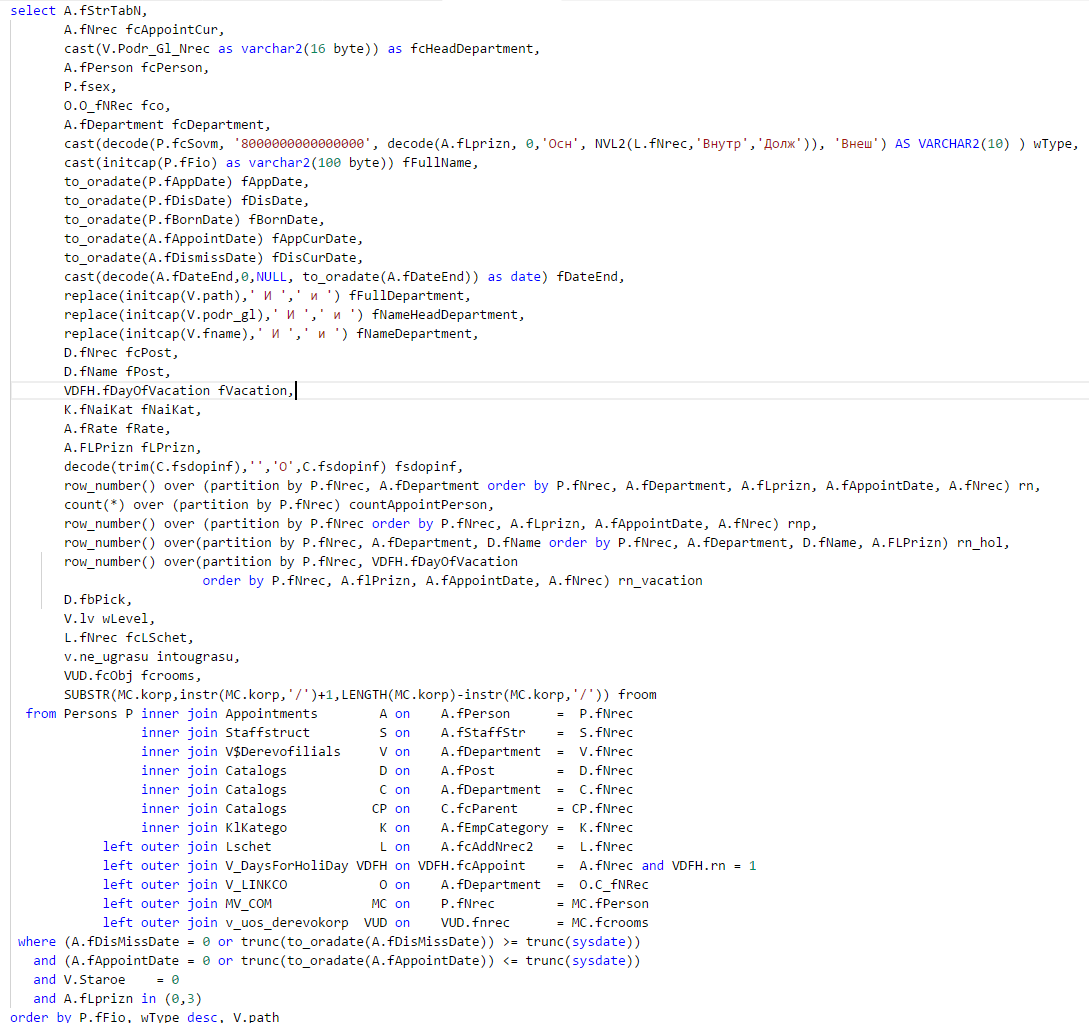


Рисунок А.3 – Реализация вида для получения информации о пользователях

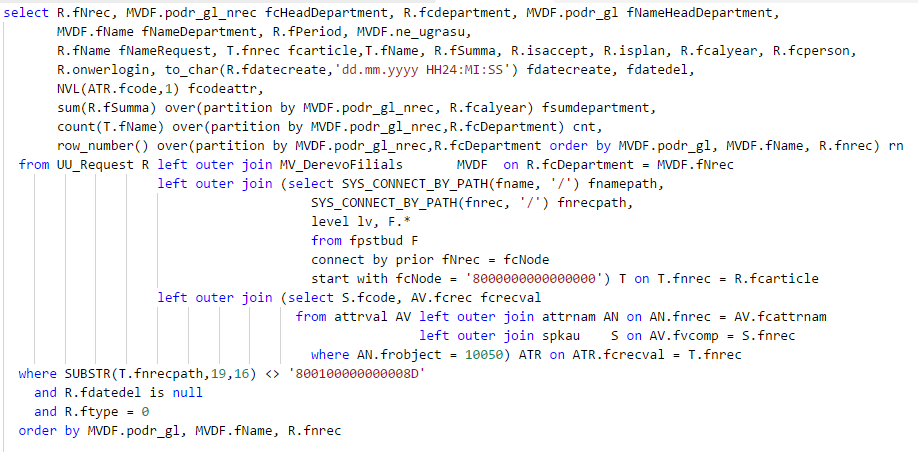


Рисунок А.4 – Реализация вида для вывода информации о заполненных заявлениях

# Приложение Б

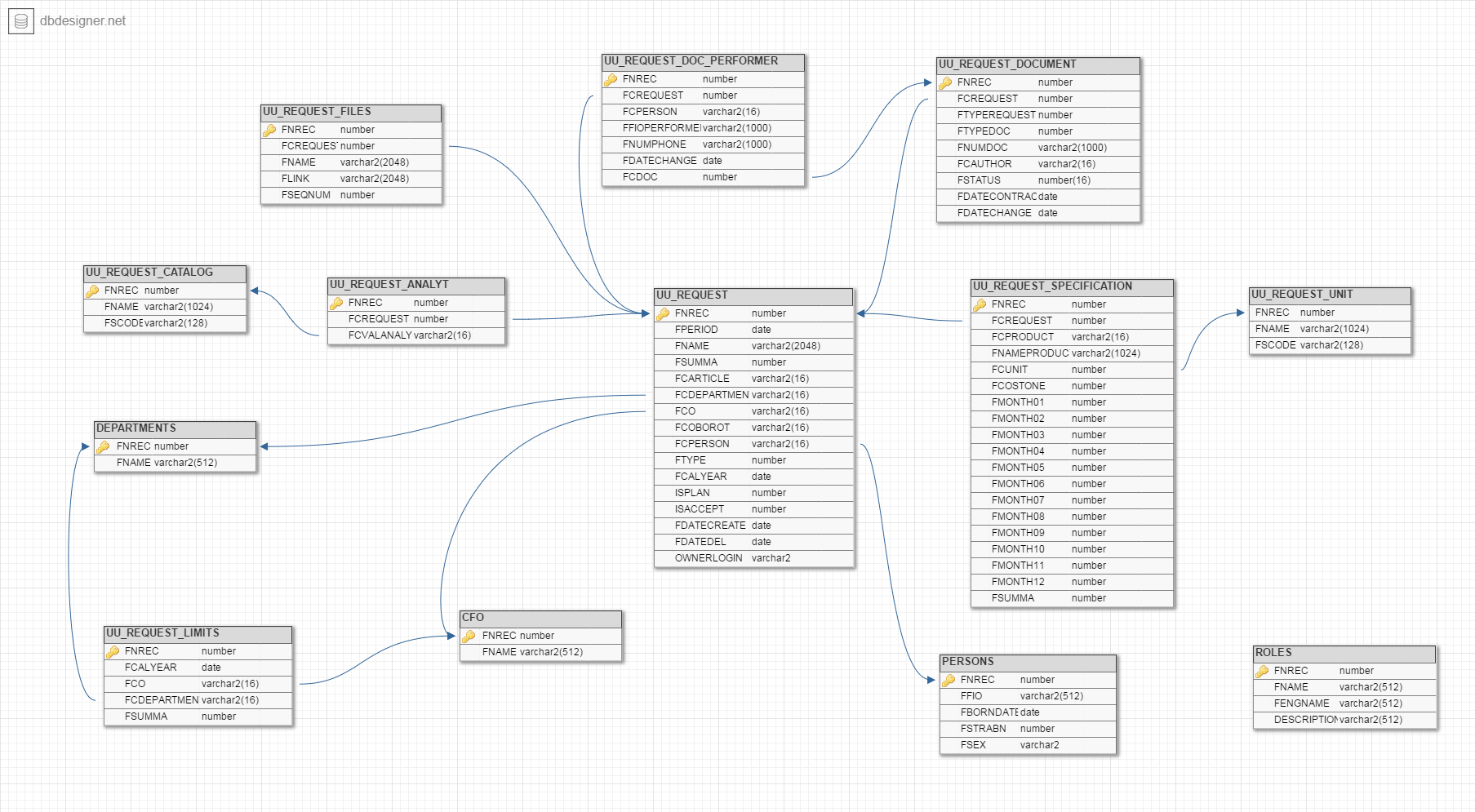


Рисунок Б.1 – Схема базы данных системы

# Приложение В

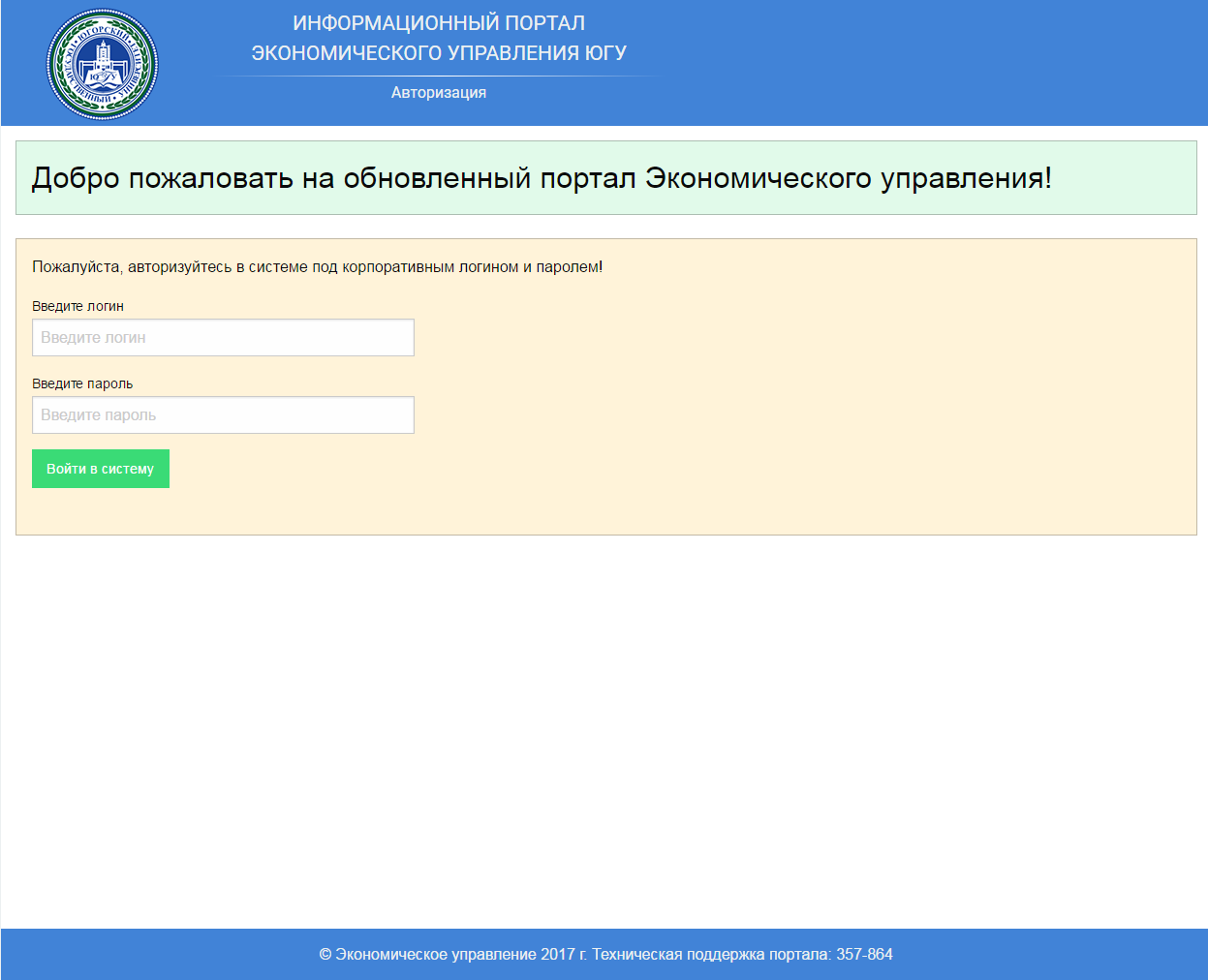


Рисунок В.1 – Интерфейс модуля авторизации

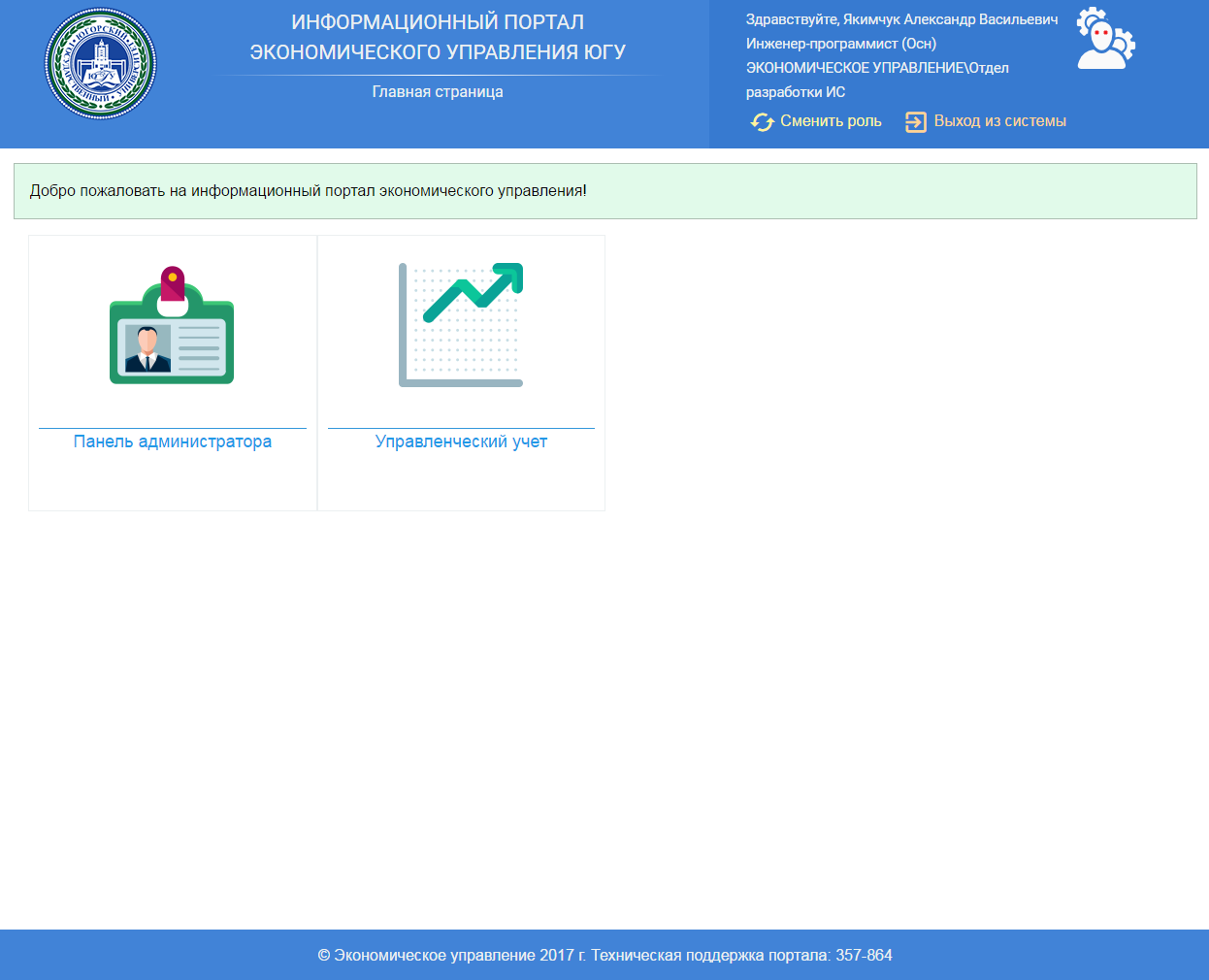


Рисунок В.2 – Интерфейс главной страницы

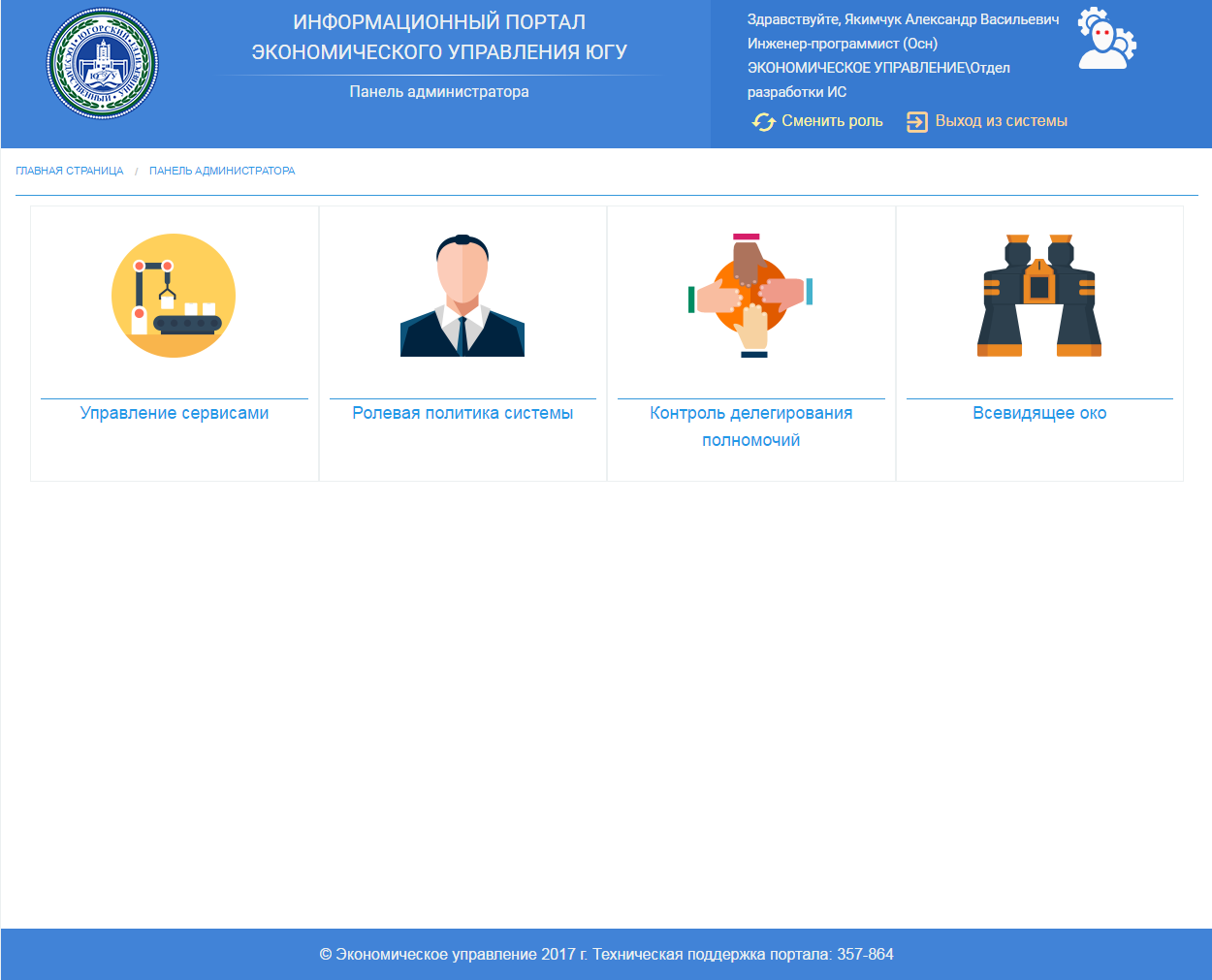


Рисунок В.3 –Интерфейс панели администратора

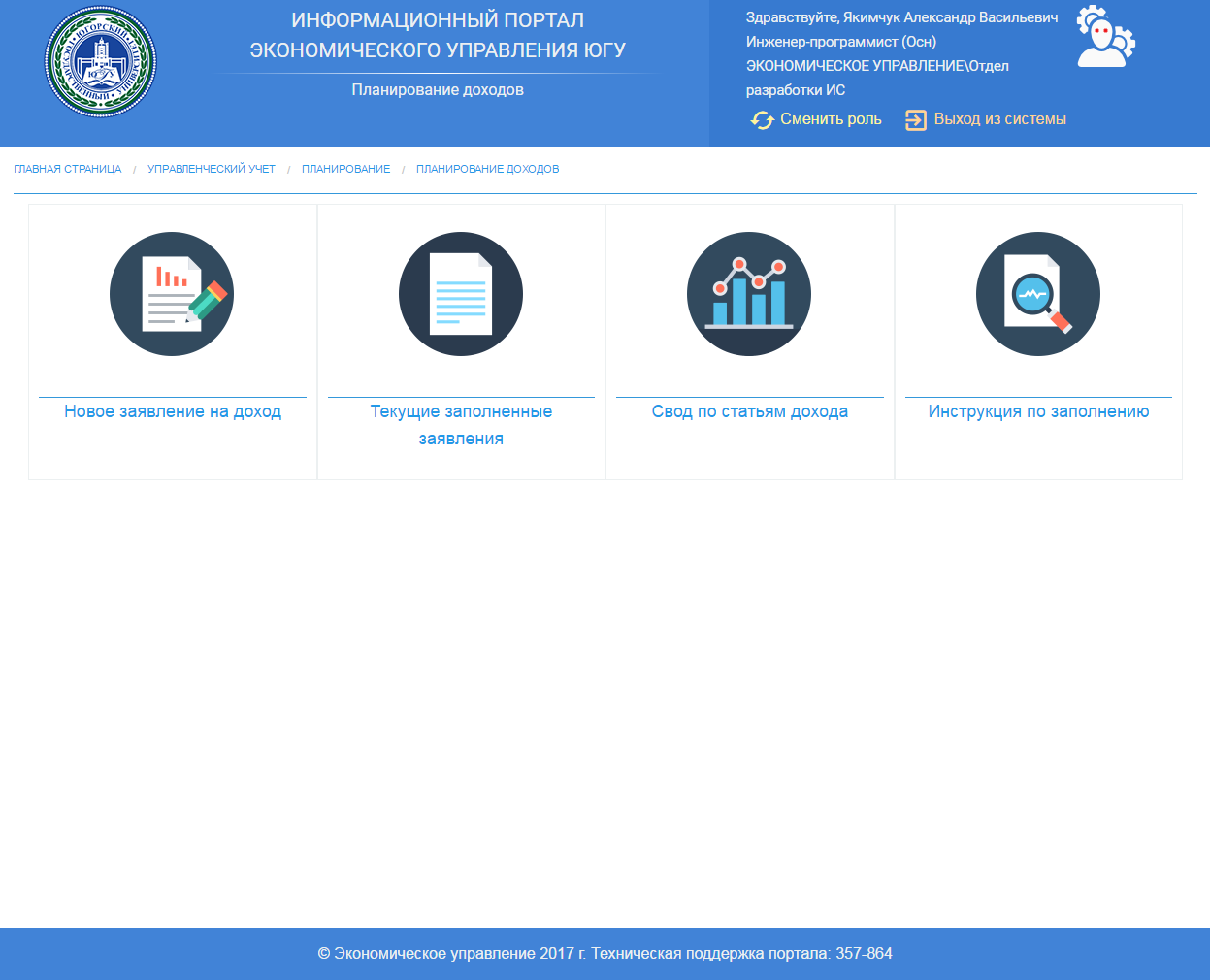


Рисунок В.4 – Интерфейс модуля планирование доходов

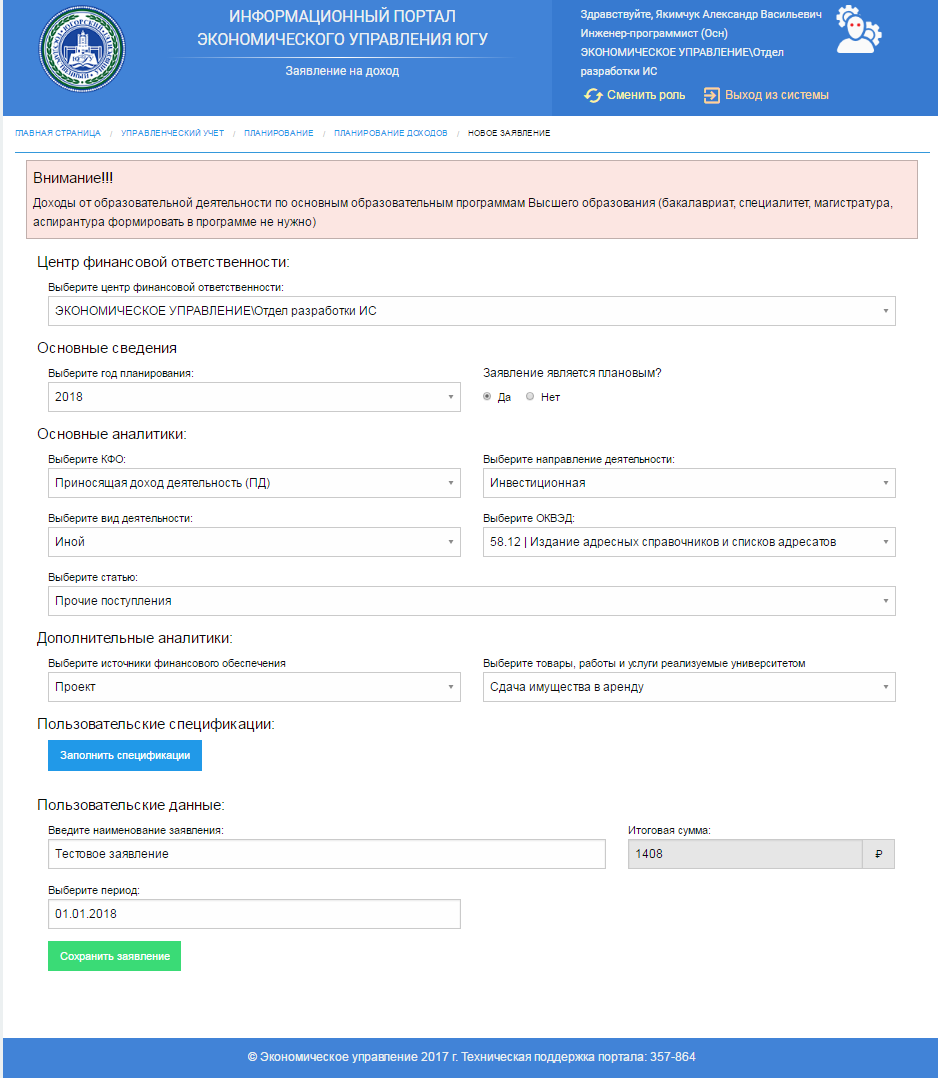


Рисунок В.5 – Интерфейс модуля создание нового заявления на доход