[**Принятые сокращения**](#_684yyycrp8w5)4

[**Глоссарий**](#_ambmbyfpwuz4)4

[**Введение**](#_fr3adoohu858)6

[**Аналитический раздел**](#_1wcaatgyt5ka)7

[Описание системы](#_8x74qxhibivj) 7

[Существующие аналоги](#_b7v2fk7uol5k) 7

[Требования к системе](#_vc5ymd1t2n26) 7

[Требования к функциональным характеристикам](#_ik0j7s4tpdjy) 8

[Функциональные требования к системе](#_1g0h7zovg2tt) 8

[Функциональные требования к системе с точки зрения администратора](#_xk7kxkod6r8p) 8

[Функциональные требования к системе с точки зрения руководителя](#_4w8x2rqp287f) 9

[Функциональные требования к системе с точки зрения студента](#_npv723oy0967) 9

[Входные данные системы](#_rq9zzrembpf7) 10

[Выходные данные системы](#_2knva2yahmrs) 12

[Требования к составу и параметрам технических средств](#_ue186v1gu87g) 14

[Требования к надежности](#_qyfrr0vrwcd) 14

[Требования к реализации](#_luvk9ez86zsm) 14

[Роль географического местоположения пользователей](#_gjdgxs) 23

[Топология системы](#_opwkf4d5ag9v) 15

[Требования по реализации](#_30j0zll) 16

[Функциональные требования по подсистемам](#_1fob9te) 38

[Сценарий взаимодействия фронтенда и агрегационного сервиса](#_ys0iukoiyyfv) 19

[Сценарий взаимодействия агрегационного сервиса и бекендов](#_lucfqqw2741p) 20

[Сценарий взаимодействия бекендов с БД](#_pusa0clh7rda) 20

[Пользовательский интерфейс](#_bwzukqdsr9ys) 21

[**Конструкторский раздел**](#_t9jcm7wpcl5x)23

[Концептуальная модель системы в нотации IDEF0](#_gjdgxs) 23

[Сценарии функционирования системы](#_5g763uikji3s) 24

[Диаграммы прецедентов](#_bz11hh5oqi2r) 29

[Диаграммы классов](#_cgb4e6h1j6ji) 31

[Диаграммы деятельности](#_5cws1mffgmnf) 39

[Диаграммы последовательности действий](#_9z1kw558qkrn) 40

[Диаграммы потоков данных](#_jmvabzi1bb6r) 41

[Высокоуровневый дизайн пользовательского интерфейса](#_3znysh7) 42

[Алгоритм распределения](#_gkpaq0k1dta3) 45

[**Технологический раздел**](#_1jmz9pjrj1wg)47

[Выбор языка программирования](#_hmgl1y1yfzv1) 47

[Выбор фреймворка](#_v9szpyjzj1zj) 48

[Выбор операционной системы](#_m75h3t3ub7ba) 49

[Выбор СУБД](#_3ak97bp03ukx) 50

[**Заключение**](#_axihwe8pgqkj)52

[**Список литературы**](#_7sifowc5ln1d)53

# Принятые сокращения

СУБД — Система Управления Базами Данных;

CI — Continuous Integration;

CSS — Cascading Style Sheets.

# Глоссарий

1. Научные предпочтения — набор направлений из предложенного списка направлений;
2. Личные предпочтения:
   1. для руководителя — список студентов;
   2. для студента — список руководителей.
3. Работа — курсовая (дипломная) работа (проект) по определенной дисциплине. Представляется в виде полного наименования работы (проекта), например: «Курсовая работа по компьютерной графике»;
4. Распределение — набор связей между Работами, руководителями и студентами;
5. Gitlab — сайт и система управления [репозиториями](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%B9) исходного кода для системы управления версиями [Git](https://ru.wikipedia.org/wiki/Git). Позволяет хранить исходный код программы и вести историю ее изменений;
6. Тонкий клиент — метод разработки, при котором происходит формирование веб-страницы для клиента на стороне сервера и последующая отправка этой веб-страницы клиенту;
7. Continuous Integration — непрерывная интеграция, это практика [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), которая заключается в слиянии рабочих копий в общую основную ветвь разработки несколько раз в день и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных [дефектов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BE%D1%88%D0%B8%D0%B1%D0%BA%D0%B0) и решения интеграционных проблем.
8. Cascading Style Sheets — каскадные таблицы стилей, [формальный язык](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) описания внешнего вида документа, написанного с использованием [языка разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8).

# Аналитический раздел

## Описание системы

Основное назначение описываемой системы — распределение руководителей Работ по студентам на основании научных и личных предпочтений руководителей и студентов.

Система состоит из нескольких связанных микро-сервисов:

1. Сервис распределения, который отвечает за автоматическое формирование Распределения и выдачу итоговой информации, а также за редактирование Распределения вручную;
2. Сервис регистрации, аутентификации и авторизации;
3. Сервис для работы с репозиториями в системе gitlab;
4. Агрегационный сервис;
5. Сервис пользовательского интерфейса.

## Существующие аналоги

На момент написания расчетно-пояснительной записки аналогов такого ПО не обнаружено в открытом доступе.

## Требования к системе

1. В случае недоступности микро-сервисов выполнять их перезагрузку;
2. Предусмотреть обработку таймаутов и ошибок сервисов. В случае ошибки/недоступности некритичного функционала выполнять деградацию функциональности (продолжать функционирование системы в случае ошибок или отказа некоторых компонентов за счет сокращения функционала и/или части выходных данных там, где это возможно);
3. Обеспечить масштабируемость за счет строгой регламентации форматов запросов и ответов при взаимодействии сервисов;
4. Предоставлять итоговое распределение в нескольких форматах: \*.xlsx, \*.txt, \*.html.

## Требования к функциональным характеристикам

1. Время отклика системы на запрос пользователя не должно превышать 10 секунд.
2. Время распределения руководителей и выдачи результатов не должно превышать 60 секунд.
3. Обеспечить корректное отображение пользовательского интерфейса в браузерах Internet Explorer и Google Chrome.

## Функциональные требования к системе

1. Разделение пользователей на три роли: администратор, руководитель, студент;
2. Регистрация пользователей от имени администратора с валидацией вводимых данных;
3. Аутентификация и авторизация пользователей (операции объединяются в одну);
4. Предоставлять для каждого пользователя функционал, соответствующий его роли.
5. В системе может быть только один пользователь в роли администратора, логин и пароль для администратора генерируются автоматически при первичном развертывании системы.

## Функциональные требования к системе с точки зрения администратора

1. Регистрация пользователя в роли руководителя;
2. Регистрация пользователя в роли студента;
3. Просмотр списка студентов;
4. Просмотр списка руководителей;
5. Просмотр информации о студенте;
6. Просмотр информации о руководителе;
7. Редактирование информации о студенте;
8. Редактирование информации о руководителе;
9. Удаление пользователя;
10. Просмотр информации о текущем Распределении;
11. Редактирование информации о текущем Распределении;
12. Запуск автоматического формирования нового Распределения;
13. Добавление новой Работы;
14. Добавление нового направления для Работы.
15. Просмотр списка Работ;
16. Просмотр списка направлений;
17. Просмотр информации о Работе;

## Функциональные требования к системе с точки зрения руководителя

1. Просмотр информации о себе;
2. Редактирование информации о себе;
3. Просмотр информации о текущем Распределении;
4. Просмотр списка студентов;
5. Просмотр информации о студенте.
6. Просмотр списка Работ;
7. Просмотр списка направлений;
8. Просмотр информации о Работе;

## Функциональные требования к системе с точки зрения студента

1. Просмотр информации о себе;
2. Редактирование информации о себе;
3. Создание репозитория в системе gitlab;
4. Просмотр информации о текущем Распределении;
5. Просмотр списка руководителей;
6. Просмотр информации о руководителе.
7. Просмотр списка Работ;
8. Просмотр списка направлений;
9. Просмотр информации о Работе;

## Входные данные системы

1. Регистрация пользователя в роли руководителя;
   1. ФИО;
   2. должность;
   3. звание.
2. Регистрация пользователя в роли студента;
   1. ФИО;
   2. номер группы.
3. Аутентификация и авторизация пользователя;
   1. логин;
   2. пароль.
4. Просмотр списка студентов (входные данные не требуются);
5. Просмотр списка руководителей (входные данные не требуются);
6. Просмотр информации о студенте;
   1. логин студента;
7. Просмотр информации о руководителе;
   1. логин руководителя;
8. Редактирование информации о руководителе;
   1. логин руководителя;
   2. ФИО;
   3. должность;
   4. звание;
   5. адрес электронной почты;
   6. научные предпочтения;
   7. личные предпочтения.
9. Редактирование информации о студенте;
   1. логин студента;
   2. ФИО;
   3. номер группы;
   4. адрес электронной почты;
   5. научные предпочтения;
   6. личные предпочтения.
10. Удаление пользователя;
    1. логин.
11. Просмотр информации о текущем Распределении (входные данные не требуются);
12. Редактирование информации о текущем Распределении;
    1. логин руководителя;
    2. логин студента;
    3. наименование Работы;
    4. информация о создании или удалении связи.
13. Запуск автоматического формирования нового Распределения (входные данные не требуются);
14. Добавление новой Работы;
    1. наименование работы.
15. Добавление нового направления.
    1. наименование работы;
    2. наименование направления.
16. Создание репозитория в системе gitlab;
    1. логин для gitlab;
    2. пароль для gitlab.
17. Просмотр списка Работ (входные данные не требуются);
18. Просмотр списка направлений (входные данные не требуются);
19. Просмотр информации о Работе;
    1. наименование работы.

## Выходные данные системы

В выходные данные будет входить общая информация:

* сообщение об успехе или неудаче операции;
* пояснение в случае неудачи.

1. Регистрация пользователя в роли руководителя;
   1. логин;
   2. пароль.
2. Регистрация пользователя в роли студента;
   1. логин;
   2. пароль.
3. Аутентификация и авторизация пользователя (общая информация);
4. Просмотр списка студентов;
   1. список студентов.
5. Просмотр списка руководителей;
   1. список руководителей.
6. Просмотр информации о руководителе;
   1. логин руководителя;
   2. ФИО;
   3. должность;
   4. звание;
   5. адрес электронной почты;
   6. научные предпочтения;
   7. личные предпочтения.
7. Просмотр информации о студенте;
   1. логин студента;
   2. ФИО;
   3. номер группы;
   4. адрес электронной почты;
   5. научные предпочтения;
   6. личные предпочтения.
8. Редактирование информации о руководителе;
   1. новое ФИО;
   2. новая должность;
   3. новое звание;
   4. новый адрес электронной почты;
   5. новые научные предпочтения;
   6. новые личные предпочтения.
9. Редактирование информации о студенте;
   1. новое ФИО;
   2. новый номер группы;
   3. новый адрес электронной почты;
   4. новые научные предпочтения;
   5. новые личные предпочтения.
10. Удаление пользователя (общая информация);
11. Просмотр информации о текущем Распределении;
    1. текущее Распределение;
12. Редактирование информации о текущем Распределении;
    1. новое текущее Распределение.
13. Запуск автоматического формирования нового Распределения;
    1. новое текущее Распределение.
14. Добавление новой Работы (общая информация);
15. Добавление нового направления (общая информация).
16. Создание репозитория в системе gitlab (общая информация);
17. Просмотр списка Работ;
    1. список Работ.
18. Просмотр списка направлений;
    1. список направлений.
19. Просмотр информации о Работе;
    1. наименование Работы;
    2. список направлений по данной Работе.

## Требования к составу и параметрам технических средств

Каждый микро-сервис располагается на отдельной машине и работает с отдельной базой данных. Каждая машина соответствует следующим параметрам: оперативная память не менее чем 4 GiB, частота процессора не менее чем 2,5 GHz для четырехъядерных процессоров, емкость жесткого диска не менее 100 GiB.

## Требования к надежности

1. Предусмотреть обработку всех внутренних ошибок сервисов и вывод понятной пользователю информации об этих ошибках;
2. Предусмотреть деградацию функциональности в случае недоступности некоторых сервисов там, где это возможно;
3. Создавать резервную копию базы данных на каждом микро-сервисе не реже, чем раз в сутки;
4. Для каждого микро-сервиса предусмотреть основной и резервный серверы. При недоступности основного переходить на использование резервного сервера.

## Требования к реализации

1. Использование CSS в пользовательском интерфейсе обязательно;
2. Используется тонкий клиент;
3. Все системы должны собираться и развертываться через CI;
4. Использование реляционной СУБД. Каждый микро-сервис имеет свою собственную базу данных и находится на отдельной физической машине.

## Роль географического местоположения пользователей

При создании этой системы предполагается ее использование только в рамках локальной сети кафедры. Вследствие этого латентность ответов системы имеет незначительную зависимость от географического местоположения пользователей.

## Топология системы

На Рис. 1 изображен предполагаемый вариант топологии разрабатываемой распределенной системы.

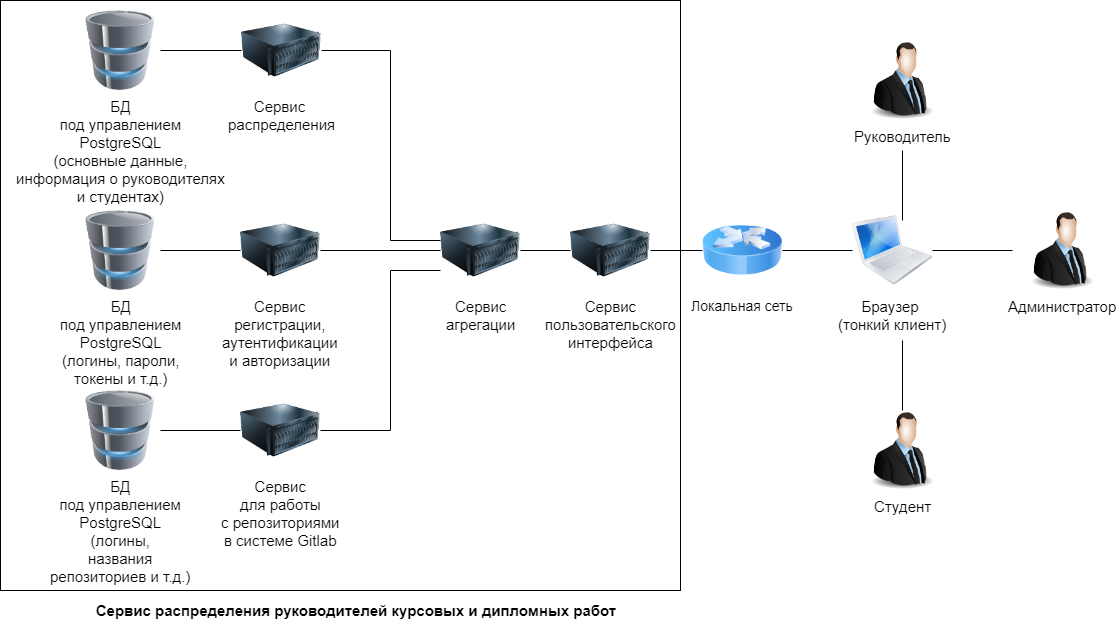


Рис. 1. Топология системы.

Система будет состоять из фронтенда (сервис пользовательского интерфейса), сервиса агрегации и трех бекендов (сервисов), что наиболее целесообразно для реализации ее основного назначения.

**Сервис регистрации, аутентификации и авторизации** отвечает за разграничение прав доступа пользователей системы и реализует следующие функции:

1. регистрация пользователя (руководителя или студента) администратором;
2. удаление пользователя администратором;
3. аутентификация и авторизация пользователя (вход, или «логин»);
4. выход из системы («логаут»);
5. проверка доступа.

**Сервис для работы с репозиториями в системе Gitlab** отвечает за создание аккаунтов и репозиториев в системе Gitlabи за получение данных об аккаунтах и репозиториях.

**Сервис распределения реализует** следующие функции:

1. получение данных о руководителях и студентах;
2. изменение данных о руководителях и студентах;
3. добавление руководителя или студента;
4. удаление руководителя или студента;
5. изменение данных о работах (проектах);
6. изменение данных о направлениях работ;
7. автоматическое формирование распределения руководителей по студентам;
8. редактирование распределения вручную.

**Сервис агрегации** выполняет запросы к бекендам, агрегирует ответы бекендов и отсылает ответ фронтенду.

**Фронтенд** принимает запросы от пользователей по протоколу HTTP и анализирует их. На основе проведенного анализа фронтенд выполняет запрос к сервису агрегации, формирует ответ в виде HTML страницы и отправляет пользователю.

## Требования по реализации

1. Все бекенды, сервис агрегации и фронтенд должны быть запущены изолированно друг от друга.
2. Требуется использовать СОА (сервис-ориентированную архитектуру).
3. Необходимо реализовать один веб-интерфейс для фронтенда. Интерфейс должен быть доступен через тонкий клиент – браузер.
4. Серверы бекендов недоступны пользователю, это реализуется валидацией запросов от сервиса агрегации.
5. Валидация входных данных должна производиться и на стороне пользователя, и на стороне сервиса агрегации. Бекенды не должны валидировать входные данные, так как пользователь не может к ним обращаться напрямую, бекенды должны получать уже отфильтрованные входные данные от сервиса агрегации.

## Функциональные требования по подсистемам

***Фронтенд*** – это серверное приложение, при разработке которого необходимо учитывать следующие факторы:

1. фронтенд должен принимать запросы по протоколу HTTP и формировать ответы пользователям сервиса в формате HTML;
2. в зависимости от типа запроса фронтенд должен формировать соответствующий запрос к агрегационному сервису;
3. запросы к агрегационному сервису необходимо осуществлять по протоколу HTTP. Данные необходимо передавать в формате JSON.

***Агрегационный сервис*** – это серверное приложение, которое должно отвечать следующим требованиям по разработке:

1. Агрегационный сервис должен принимать и возвращать данные в формате JSON по протоколу HTTP;
2. Обрабатывать запросы от фронтенда и осуществлять последовательные запросы на бекенды, агрегировать ответы бекендов, формировать ответ и отправлять фронтенду;
3. Агрегационный сервис не имеет базы данных.

***Сервис регистрации, аутентификации и авторизации*** должен быть серверным приложением, которое:

1. должно принимать и отвечать на запросы в формате JSON по протоколу HTTP;
2. Выполнять авторизацию и аутентификацию пользователей, проверять и инвалидировать токены авторизации, а также регистрировать пользователей;
3. База данных, содержащая информацию о токенах авторизации, должна находиться на этом же сервере. Доступ к СУБД должен осуществляться по протоколу TCP;
4. Резервирование базы данных должно производиться по расписанию.

***Сервис распределения*** должен быть серверным приложением, которое:

1. должно принимать и отвечать на запросы в формате JSON по протоколу HTTP;
2. обрабатывать запросы на создание, удаление, редактирование и получение профиля пользователя по идентификатору пользователя;
3. обрабатывать запросы на получение списка профилей. Профиль представляет набор данных о руководителе или студенте;
4. обрабатывать запросы на формирование, редактирование и получение распределения;
5. при разработке базы данных, содержащей информацию о пользователях, требуется учитывать следующие требования:
   1. доступ к СУБД должен осуществляться по протоколу TCP;
   2. необходимо разработать скрипт для автоматического создания резервной копий базы данных по расписанию;
   3. первичным ключом является идентификатор пользователя.

***Сервис для работы с репозиториями в системе Gitlab*** должен быть серверным приложением, которое:

1. принимает и отвечает на запросы в формате JSON по протоколу HTTP;
2. обрабатывать запросы на создание аккаунта или репозитория и получение информации об аккаунте или репозитории по идентификатору пользователя;
3. обрабатывать запросы на получение списка репозиториев;
4. разработка базы данных, содержащей информацию о репозиториях, требует учета следующих требований:
   1. доступ к СУБД должен осуществляться по протоколу TCP;
   2. необходимо разработать скрипт для автоматического создания резервной копий базы данных по расписанию.

## Сценарий взаимодействия фронтенда и агрегационного сервиса

Рассмотрим, как взаимодействуют фронтенд и агрегационный сервис на примере выполнения запроса от пользователя на получение результатов распределения студентов по научным руководителям:

1. На фронтенд приходит данный запрос пользователя;
2. Фронтенд анализирует запрос пользователя и формирует соответствующий запрос к агрегационному сервису. Далее выполняется запрос к агрегационному сервису. На запрос устанавливают максимальное время ожидания ответа (таймаут). Если не удалось выполнить успешный запрос, пользователю возвращается ошибка;
3. Если ошибки не произошло, то производится генерация HTML содержимого страницы ответа пользователю с использованием данных, полученных от агрегационного сервиса.

## Сценарий взаимодействия агрегационного сервиса и бекендов

Рассмотрим, как происходит взаимодействие агрегационного сервиса с бекендами на примере запроса изменения профиля студента.

1. Агрегационный сервис получает запрос от фронтенда;
2. Агрегационный сервис, отправляет запрос на проверку доступа к сервису регистрации, аутентификации и авторизации;
3. В случае если студент не авторизован, то формируется ответ с сообщением о необходимости авторизации;
4. В случае если пользователь авторизован, но не имеет доступа к данному профилю студента, то формируется ответ с сообщением об отсутствии прав доступа;
5. В случае если пользователь авторизован и имеет доступ к редактированию данного профиля студента, то агрегационный сервис далее отправляет запрос к сервису распределения с новыми данными для профиля студента. На запрос устанавливается максимальное время ожидания ответа (таймаут). Если запрос не успевает выполниться за данное время, то формируется ответ о недоступности сервиса в данный момент времени;
6. Производится генерация JSON объекта ответа, который отправляется на фронтенд.

## Сценарий взаимодействия бекендов с БД

Рассмотрим, как происходит взаимодействие бекендов с БД на примере бекенда сервиса распределения и запроса на получение списка студентов.

1. Сервис распределения получает запрос от агрегационного сервиса и преобразовывает его в запрос к СУБД;
2. Если БД успешно обработала запрос, то генерируется ответ с использованием полученных данных, иначе пользователю возвращается ошибка.

## Пользовательский интерфейс

Для реализации серверного приложения фронтенда используется схема «модель – вид – элемент управления» (Model – View – Controller, MVC). Пользовательский интерфейс представляет собой “вид” из этой схемы. Такое разделение позволяет осуществлять разработку пользовательского интерфейса отдельной командой разработчиков, которые специализируются на этом, так как для реализации интерфейса не требуется умений программировать бекенд или фронтенд: разработка интерфейса обычно осуществляется с помощью языка разметки HTML и языка шаблонов, а также с помощью используемого фреймворка.

Пользовательский интерфейс в разрабатываемой системе должен обладать следующими характеристиками:

1. Адаптивность к размеру экрана устройства пользователя – пользовательский интерфейс «подстраивается» под всевозможные размеры экранов устройств: мобильных телефонов, планшетов, ноутбуков и т.д. В зависимости от размера экрана меняется размер элементов управления и структура некоторых элементов;
2. Кроссбраузерность – способность интерфейса работать практически в любом браузере любой версии;
3. Расширяемость – возможность легко расширять и модифицировать пользовательский интерфейс. Это свойство обеспечивается за счет модульности исходного кода пользовательского интерфейса (разбиение на файлы, общие блоки, наследование от общего шаблона);
4. Доступность в 3 клика – любая страница портала доступна максимум в 3 клика с главной страницы портала. Это стандартное правило при проектировании пользовательского интерфейса, оно позволяет сильно упростить доступ пользователей к контенту на портале и приводит к хорошей структуре страниц портала;
5. Интуитивно понятный интерфейс – все кнопки имеют подписи при наведении на них, многие содержат иконки, облегчающие восприятие пользователем. Используются стандартные элементы управления;
6. Быстрый отклик – используются специальные техники для ускорения работы страниц портала (приведение исходного кода к такому виду, что код становится трудночитаемым, но полностью сохраняет свою функциональность).

# Конструкторский раздел

## Концептуальная модель системы в нотации IDEF0

Общая концепция разрабатываемой системы строится на основе обработки научных и личных предпочтений студентов и руководителей для формирования распределения. Список курсовых и дипломных работ и их направлений определяется спецификой кафедры. Концептуальная модель системы представлена на рисунках 2 и 3.

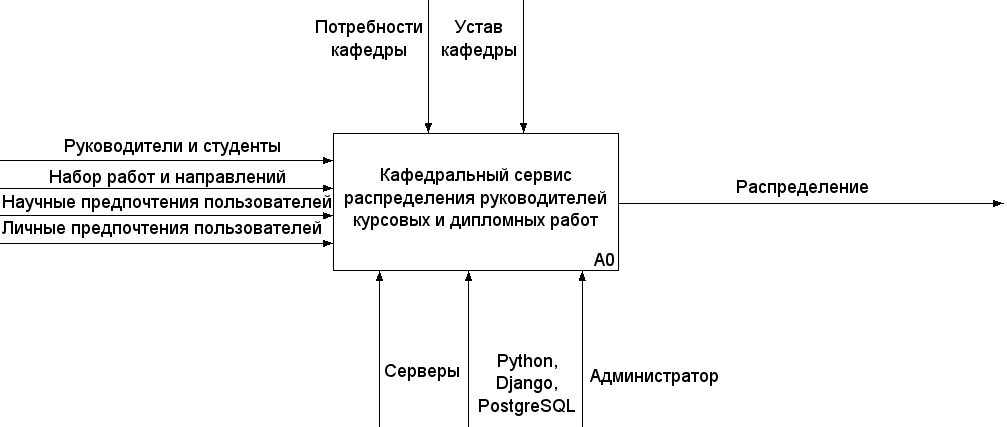


Рис. 2. Концептуальная модель системы в нотации IDEF0.

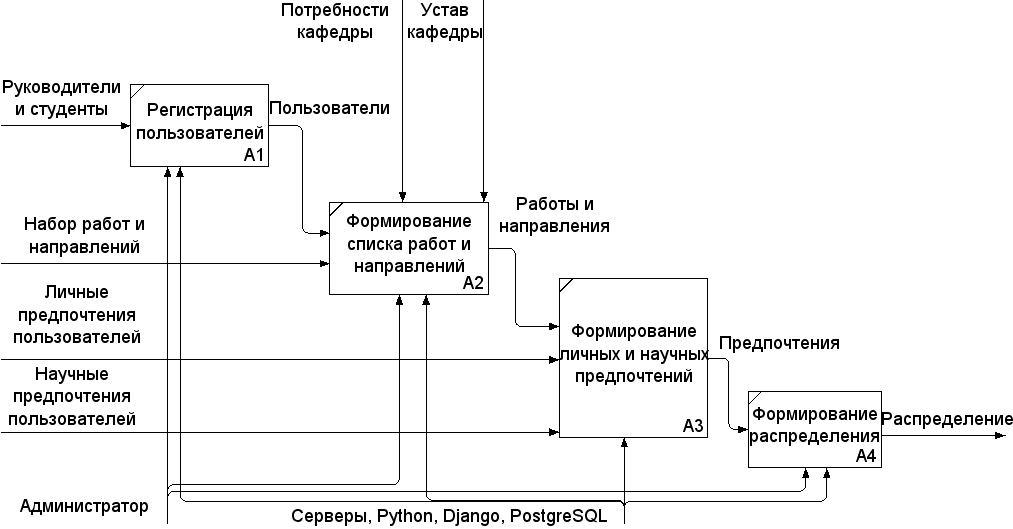


Рис. 3. Детализированная концептуальная модель системы в нотации IDEF0.

## Сценарии функционирования системы

Авторизация пользователя:

1. Пользователь заходит на главную страницу сайта;
2. Пользователь нажимает на кнопку «Войти» и перенаправляется на страницу авторизации;
3. Пользователь вводит логин и пароль и нажимает на кнопку «Войти»;
4. В случае успешной авторизации пользователя перенаправляется на главную страницу;
5. В случае неуспешной авторизации пользователю выводится сообщение о причине неуспешной авторизации (заполнены не все поля, данные логин и пароль неверны и т. п.).

Регистрация пользователя:

1. Администратор проходит авторизацию;
2. Администратор нажимает кнопку «Зарегистрировать нового пользователя»;
3. Администратор выбирает группу пользователя («Руководитель» или «Студент»);
4. Администратор вводит обязательные и, возможно, необязательные поля и нажимает кнопку «Зарегистрировать»;
5. В случае успешной регистрации Администратор перенаправляется на страницу с информацией о зарегистрированном пользователе;
6. В случае неуспешной регистрации Администратору выводится сообщение о причине неуспешной регистрации (заполнены не все обязательные поля и т. п.).

Просмотр информации о пользователе:

1. Пользователь проходит авторизацию;
2. Пользователь нажимает на кнопку «Руководители» («Студенты»);
3. Пользователь перенаправляется на страницу со списком руководителей (студентов);
4. Пользователь нажимает на элемент списка, содержащий фамилию и инициалы руководителя (фамилию, инициалы и группу студента);
5. Пользователь перенаправляется на страницу с информацией о руководителе (студенте).

Редактирование информации о пользователе:

1. Администратор (пользователь) проходит авторизацию;
2. Администратор выполняет шаги для просмотра информации о пользователе (пользователь нажимает кнопку «Профиль»);
3. Администратор (пользователь) нажимает на кнопку «Редактировать»;
4. Администратор перенаправляется на страницу редактирования пользователя (пользователь перенаправляется на страницу редактирования своего профиля);
5. Администратор (пользователь) заполняет поля профиля пользователя нужным образом и нажимает на кнопку «Сохранить»;
6. В случае успешного редактирования Администратор (пользователь) перенаправляется на страницу просмотра информации о пользователе и видит внесенные изменения;
7. В случае неуспешного редактирования Администратор (пользователь) получает сообщение о причине неуспешного редактирования (заполнены не все обязательные поля и т. п.).

Удаление пользователя:

1. Администратор проходит авторизацию;
2. Администратор выполняет шаги для просмотра информации о пользователе;
3. Администратор нажимает кнопку «Удалить»;
4. В появившемся диалоговом окне подтверждения удаления пользователя Администратор нажимает «Да» или «Нет»;
5. В случае нажатия кнопки «Да» Администратор перенаправляется на страницу со списком руководителей или студентов и видит внесенные изменения;
6. В случае нажатия кнопки «Нет» диалоговое окно закрывается, и Администратор остается на странице просмотра информации о пользователе.

Просмотр информации о текущем распределении:

1. Пользователь проходит авторизацию;
2. Пользователь нажимает на кнопку «Распределение»;
3. Пользователь перенаправляется на страницу с информацией о текущем распределении.

Редактирование информации о текущем распределении:

1. Администратор проходит шаги для просмотра информации о текущем распределении;
2. Администратор нажимает на кнопку «Редактировать»;
3. Администратор перенаправляется на страницу редактирования текущего распределения;
4. Администратор изменяет нужные связи студент-руководитель-работа;
5. Администратор нажимает кнопку «Сохранить»;
6. Администратор перенаправляется на страницу просмотра информации о текущем распределении и видит внесенные изменения.

Запуск автоматического формирования нового распределения:

1. Администратор проходит шаги для просмотра информации о текущем распределении;
2. Администратор нажимает на кнопку «Сформировать новое распределение автоматически»;
3. Администратор перенаправляется на страницу с индикатором прогресса формирования нового распределения;
4. По завершении формирования нового распределения Администратор перенаправляется на страницу просмотра информации о текущем распределении и видит внесенные изменения.

Просмотр списка работ (направлений):

1. Пользователь проходит авторизацию;
2. Пользователь нажимает на кнопку «Работы» («Направления»);
3. Пользователь перенаправляется на страницу со списком названий курсовых и дипломных работ (направлений).

Просмотр информации о работе:

1. Пользователь выполняет шаги для просмотра списка работ;
2. Пользователь нажимает на элемент списка;
3. Пользователь перенаправляется на страницу с информацией о работе (название, курс, семестр, список руководителей).

Добавление (Редактирование) работы:

1. Администратор выполняет шаги для просмотра списка работ;
2. Администратор нажимает на кнопку «Добавить» («Редактировать»);
3. Администратор перенаправляется на страницу создания новой работы (редактирования работы);
4. Администратор заполняет название работы, номер курса и семестра;
5. Администратор нажимает на кнопку «Добавить» («Сохранить»);
6. В случае успешного добавления (редактирования) Администратор перенаправляется на страницу со списком работ и видит внесенные изменения (на страницу с информацией о работе);
7. В случае неуспешного добавления (редактирования) Администратор получает сообщение о причине неуспешного добавления (редактирования).

Добавление направления:

1. Администратор выполняет шаги для просмотра списка направлений;
2. Администратор нажимает кнопку «Добавить»;
3. Администратор перенаправляется на страницу добавления нового направления;
4. Администратор вводит название направления;
5. Администратор нажимает кнопку «Добавить»;
6. В случае успешного добавления Администратор перенаправляется на страницу со списком направлений и видит внесенные изменения;
7. В случае неуспешного добавления Администратор получает сообщение о причине неуспешного добавления;

Удаление работы (направления):

1. Администратор выполняет шаги для просмотра информации о работе (списка направлений);
2. Администратор нажимает кнопку «Удалить» (нажимает на направление и затем на кнопку «Удалить»);
3. В появившемся диалоговом окне подтверждения удаления работы (направления) Администратор нажимает «Да» или «Нет»;
4. В случае нажатия кнопки «Да» Администратор перенаправляется на страницу со списком работ (направлений) и видит внесенные изменения;
5. В случае нажатия кнопки «Нет» диалоговое окно закрывается, и Администратор остается на странице просмотра информации о работе (странице просмотра списка направлений).

## Диаграммы прецедентов

Для описанных сценариев представлены диаграммы прецедентов для каждой роли системы: администратора (рис. 4), руководителя (рис. 5) и студента (рис. 6).

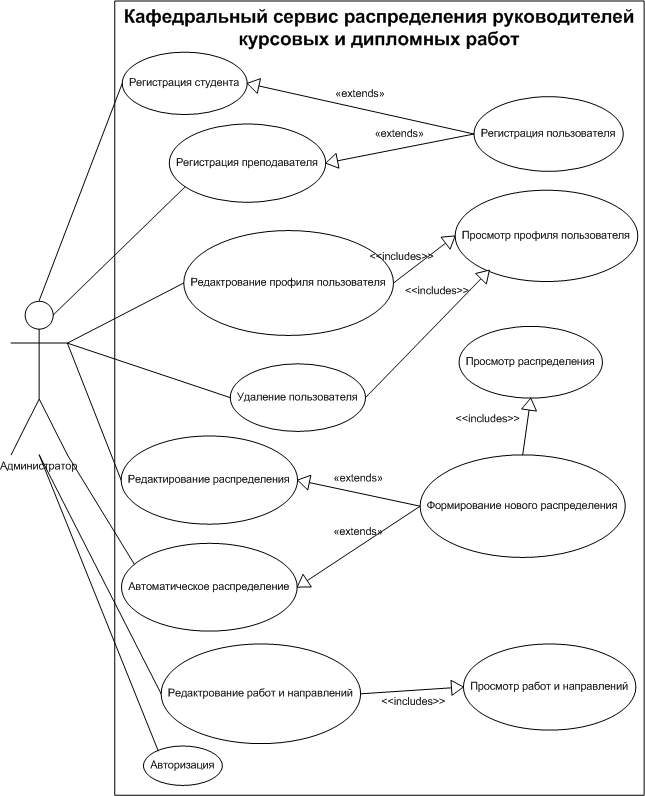


Рис. 4. Диаграмма прецедентов с точки зрения администратора.



Рис. 5. Диаграмма прецедентов с точки зрения руководителя.

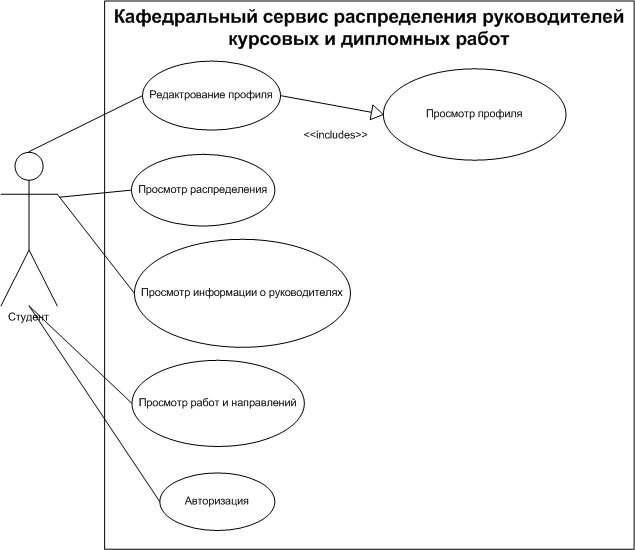


Рис. 6. Диаграмма прецедентов с точки зрения студента.

## Диаграммы классов

Сервис авторизации содержит 3 сущности, которые задействованы в регистрации, аутентификации и авторизации: пользователь (User), группа (Group), право (Permission). Диаграмма классов этого сервиса представлена на рисунке 7.

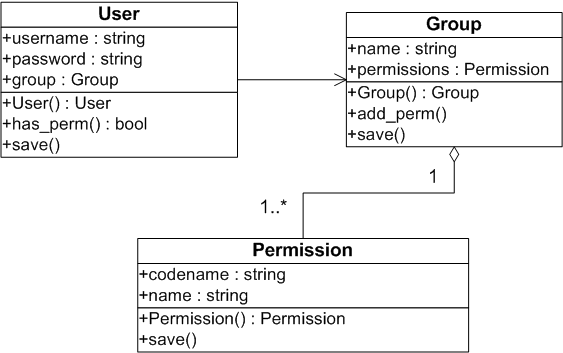


Рис. 7. Диаграмма классов сервиса авторизации.

Класс **User** представляет собой пользователя системы как носителя определенных прав доступа к системе.

Методы класса **User**:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| User() | Конструктор класса |
| has\_perm() | Определяет наличие у пользователя определенного права доступа |
| save() | Сохранить текущие значения атрибутов пользователя в базу данных |

Атрибуты класса **User**:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| username | public : *string* | Имя пользователя |
| password | public : string | Пароль пользователя в хешированном виде |
| group | public : Group | Группа, к которой принадлежит пользователь |

Класс **Group** представляет собой группу (роль) пользователя в системе с соответствующими правами.

Методы класса **Group**:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| Group() | Конструктор класса |
| add\_perm() | Добавляет к группе определенное право доступа |
| save() | Сохранить текущие значения атрибутов группы в базу данных |

Атрибуты класса **Group**:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| name | public : *string* | Наименование группы пользователя |
| permissions | public : Permission | Набор прав, которыми обладают пользователи, принадлежащие группе |

Класс **Permission** представляет собой право пользователя на доступ к определенному разделу системы.

Методы класса **Permission**:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| Permission() | Конструктор класса |
| save() | Сохранить текущие значения атрибутов права в базу данных |

Атрибуты класса **Permission**:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| name | public : *string* | Наименование права с точки зрения системы |
| codename | public : string | Наименование права, понятное пользователю |

Сервис распределения содержит 5 сущностей, которые используются в формировании распределения руководителей по студентам: направление (Direction), работа (Work), руководитель (Mentor), студент (Student), распределение (Distribution). Диаграмма классов этого сервиса представлена на рисунке 8.

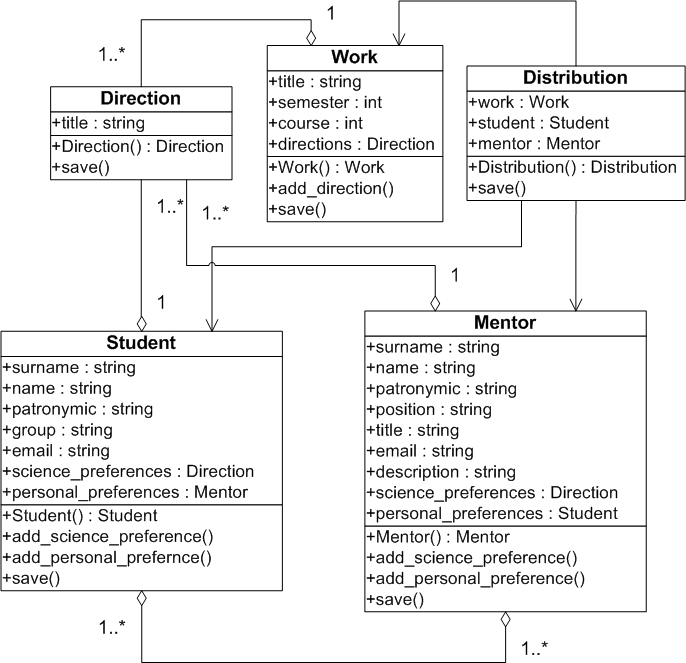


Рис. 8. Диаграмма классов сервиса распределения.

Класс **Direction** представляет собой направление для курсовой или дипломной работы.

Методы класса **Direction**:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| Direction() | Конструктор класса |
| save() | Сохранить текущие значения атрибутов направления в базу данных |

Атрибуты класса **Direction**:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| title | public : *string* | Наименование направления |

Класс **Work** представляет собой курсовую или дипломную работу.

Методы класса **Work**:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| Work() | Конструктор класса |
| add\_direction() | Добавляет к работе определенное направление |
| save() | Сохранить текущие значения атрибутов работы в базу данных |

Атрибуты класса **Work**:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| title | public : *string* | Наименование работы |
| semester | public : *int* | Номер семестра |
| course | public : *int* | Номер курса |
| directions | public : Direction | Набор направлений, которые относятся к работе |

Класс **Mentor** представляет собой пользователя в роли руководителя.

Методы класса **Mentor**:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| Mentor() | Конструктор класса |
| add\_science\_preference() | Добавить в научные предпочтения одно из существующих направлений |
| add\_personal\_preference() | Добавить в личные предпочтения одного из существующих студентов |
| save() | Сохранить текущие значения атрибутов руководителя в базу данных |

Атрибуты класса **Mentor**:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| name | public : *string* | Имя |
| surname | public : *string* | Фамилия |
| patronymic | public : *string* | Отчество |
| position | public : *string* | Должность |
| title | public : *string* | Звание |
| email | public : string | Адрес электронной почты |
| description | public : string | Дополнительная информация |
| sciensce\_preferences | public : Direction | Научные предпочтения |
| personal\_prefernces | public : Mentor | Личные предпочтения |

Класс **Student** представляет собой пользователя в роли студента.

Методы класса **Student**:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| Student() | Конструктор класса |
| add\_science\_preference() | Добавить в научные предпочтения одно из существующих направлений |
| add\_personal\_preference() | Добавить в личные предпочтения одного из существующих руководителей |
| save() | Сохранить текущие значения атрибутов студента в базу данных |

Атрибуты класса **Student**:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| name | public : *string* | Имя |
| surname | public : *string* | Фамилия |
| patronymic | public : *string* | Отчество |
| group | public : *string* | Номер группы |
| email | public : string | Адрес электронной почты |
| sciensce\_preferences | public : Direction | Научные предпочтения |
| personal\_prefernces | public : Mentor | Личные предпочтения |

Сервис по работе с репозиториями в системе gitlab содержит 1 сущность: репозиторий. Диаграмма классов этого сервиса представлена на рисунке 9.

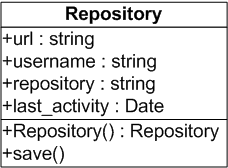


Рис. 9. Диаграмма классов сервиса по работе с репозиториями.

Класс **Repository** представляет собой репозиторий в системе gitlab.

Методы класса **Repository**:

|  |  |
| --- | --- |
| **Название метода** | **Описание метода** |
| Repository() | Конструктор класса |
| save() | Сохранить текущие значения атрибутов репозитория в базу данных |

Атрибуты класса **Repository**:

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| url | public : *string* | Ссылка на репозиторий |
| username | public : *string* | Имя пользователя в системе gitlab |
| repository | public : *string* | Наименование репозитория |
| last\_activity | public : Date | Дата и время последнего коммита |

Все сценарии взаимодействия с системой можно разделить на две группы с точки зрения участия того или иного сервиса при выполнении данного сценария:

1. фронтенд, сервис агрегации, сервис авторизации, сервис распределения;
2. фронтенд, сервис агрегации, сервис авторизации, сервис работы с репозиториями.

Таким образом, для построения диаграмм деятельности, последовательности действий и потоков данных было выбрано два типовых сценария: регистрация и создание репозитория. Это разделение было произведено с целью формирования представления о взаимодействии сервисов между собой. Все остальные сценарии взаимодействия пользователя с системой почти идентичны выбранным типовым сценариям, за исключением лишь отличий в типах передаваемых данных, что не оказывает влияния на структуру логического дизайна.

## Диаграммы деятельности

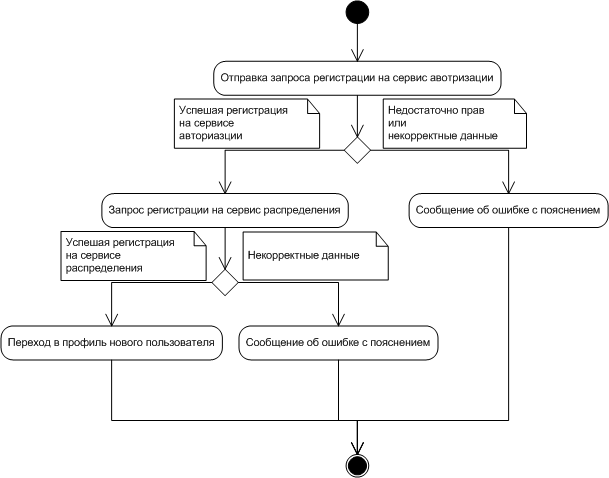


Рис. 10. Диаграмма деятельности для варианта использования «Регистрация» для администратора системы.

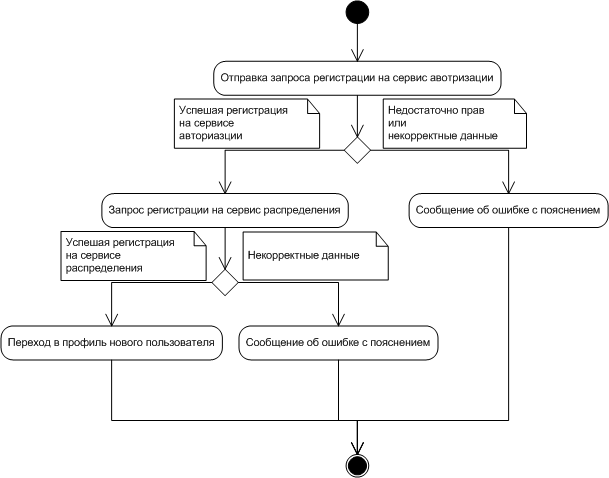


Рис. 11. Диаграмма деятельности для варианта использования «Создание репозитория» для пользователя системы в роли студента.

## Диаграммы последовательности действий

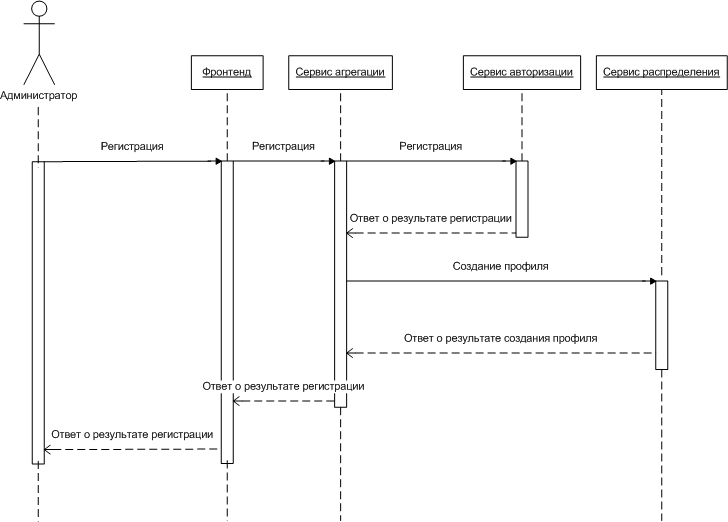


Рис. 12. Диаграмма последовательности действий при регистрации.

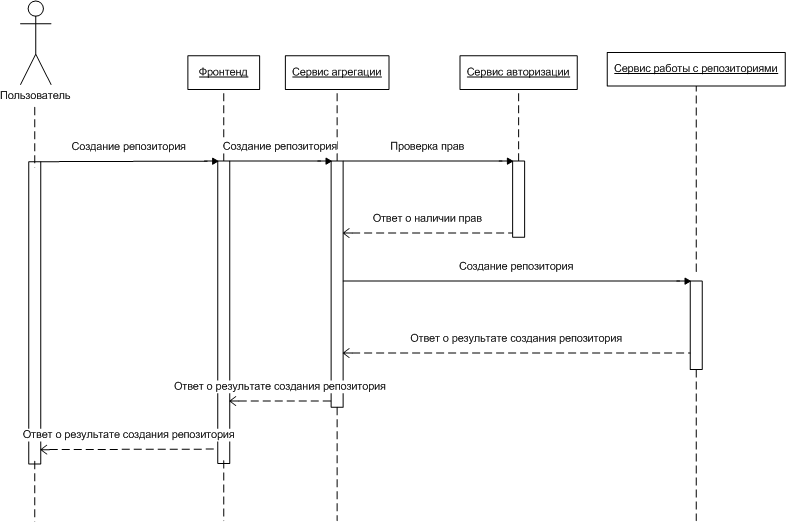


Рис. 13. Диаграмма последовательности действий при создании репозитория.

## Диаграммы потоков данных

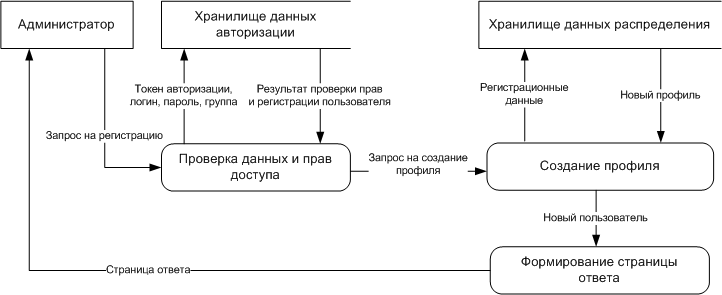


Рис. 14. Диаграмма потоков данных при регистрации.

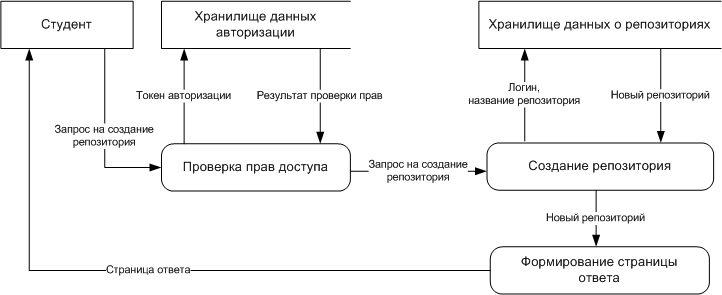


Рис. 15. Диаграмма потоков данных при создании репозитория.

## Высокоуровневый дизайн пользовательского интерфейса

Пользовательский интерфейс в разрабатываемой системе представляет собой Web-интерфейс, доступ к которому осуществляется через браузер (тонкий клиент).

Страница состоит из «шапки» (верхней части страницы, в которой находится герб МГТУ им. Н.Э. Баумана и верхнее меню со ссылками на основные разделы) и основной части.

Структуру страниц портала можно представить следующим образом:

1. Главная страница;
2. Профиль (Личный кабинет);
   1. Студент;
   2. Руководитель.
3. Страница со списком распределений;
   1. Распределение студентов по руководителям.
4. Список студентов;
5. Список руководителей;
6. Список направлений
7. Список работ;
8. Страница авторизации.

Ниже приведены основные формы портала. В форме авторизации, изображенной на рисунке 16, пользователь должен ввести свой логин и пароль.

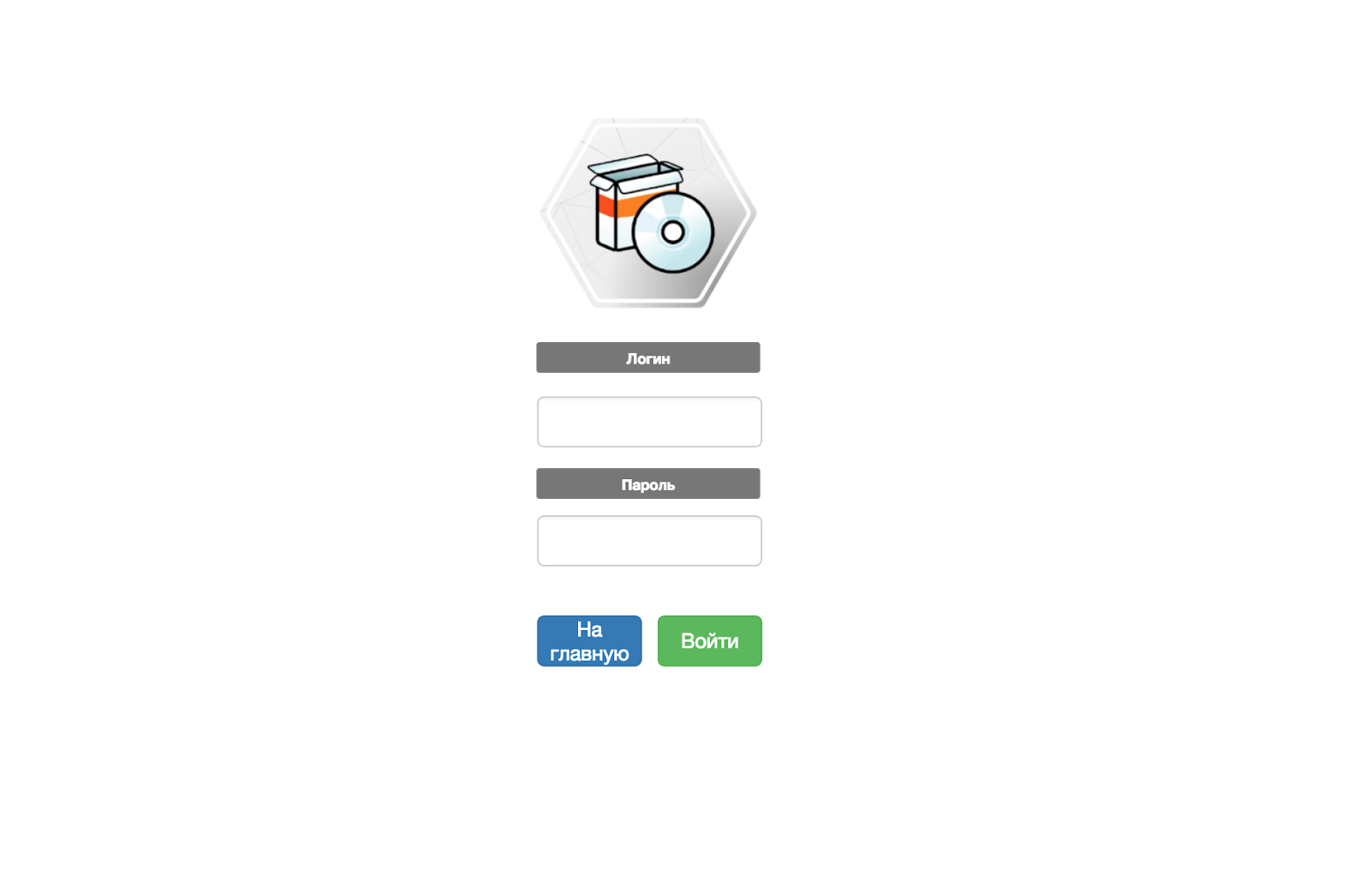


Рис. 16. Форма авторизации.

Профиль руководителя (рисунок 17) содержит информацию о должности, звании, электронной почте, а также список личных и научных предпочтений. Также существует поле для дополнительной информации.

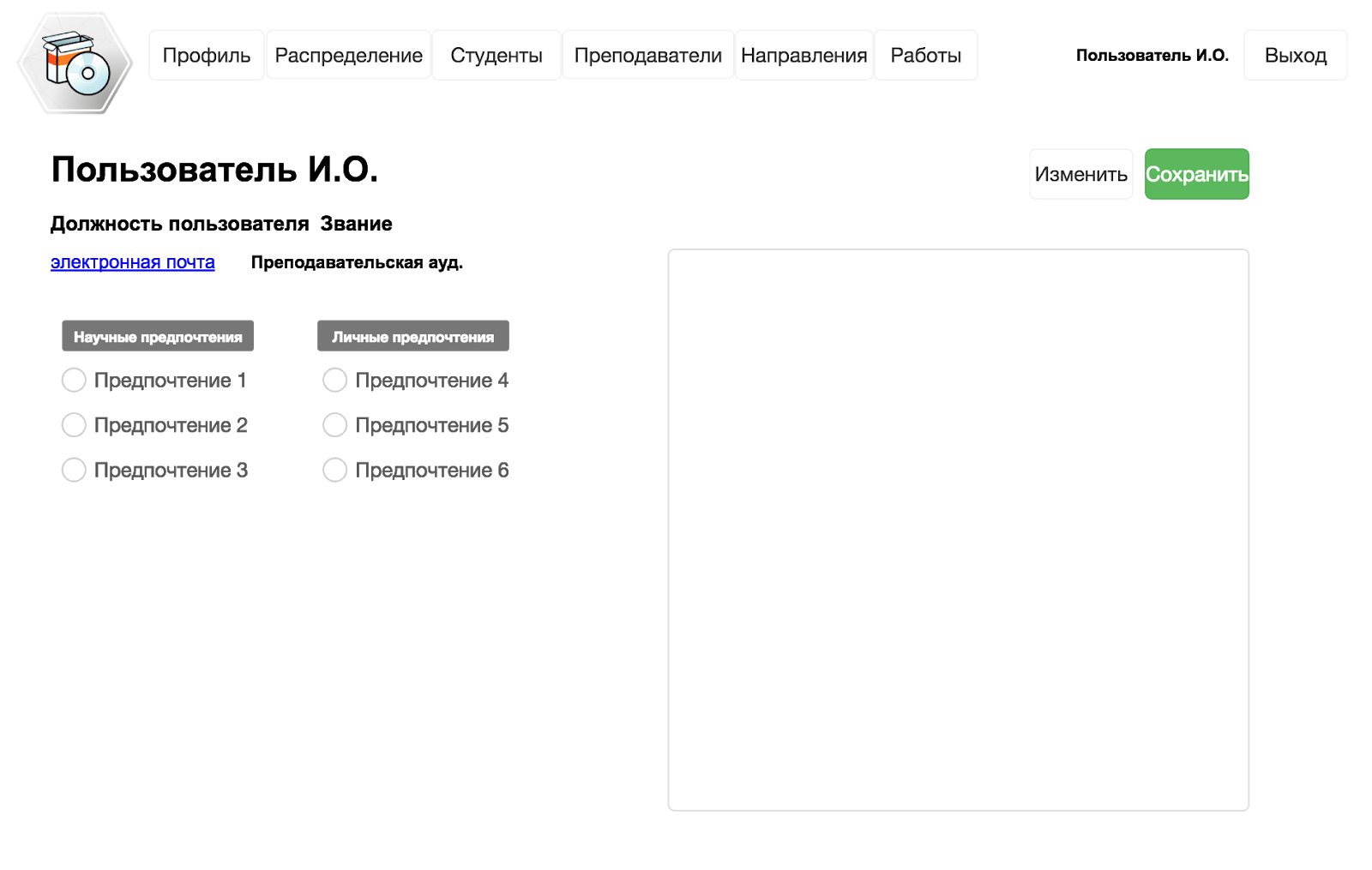


Рис 17. Профиль руководителя.

Форма профиля студента (рисунок 18) отличается от формы руководителя отсутствием поля дополнительной информации и наличием информации о группе студента.

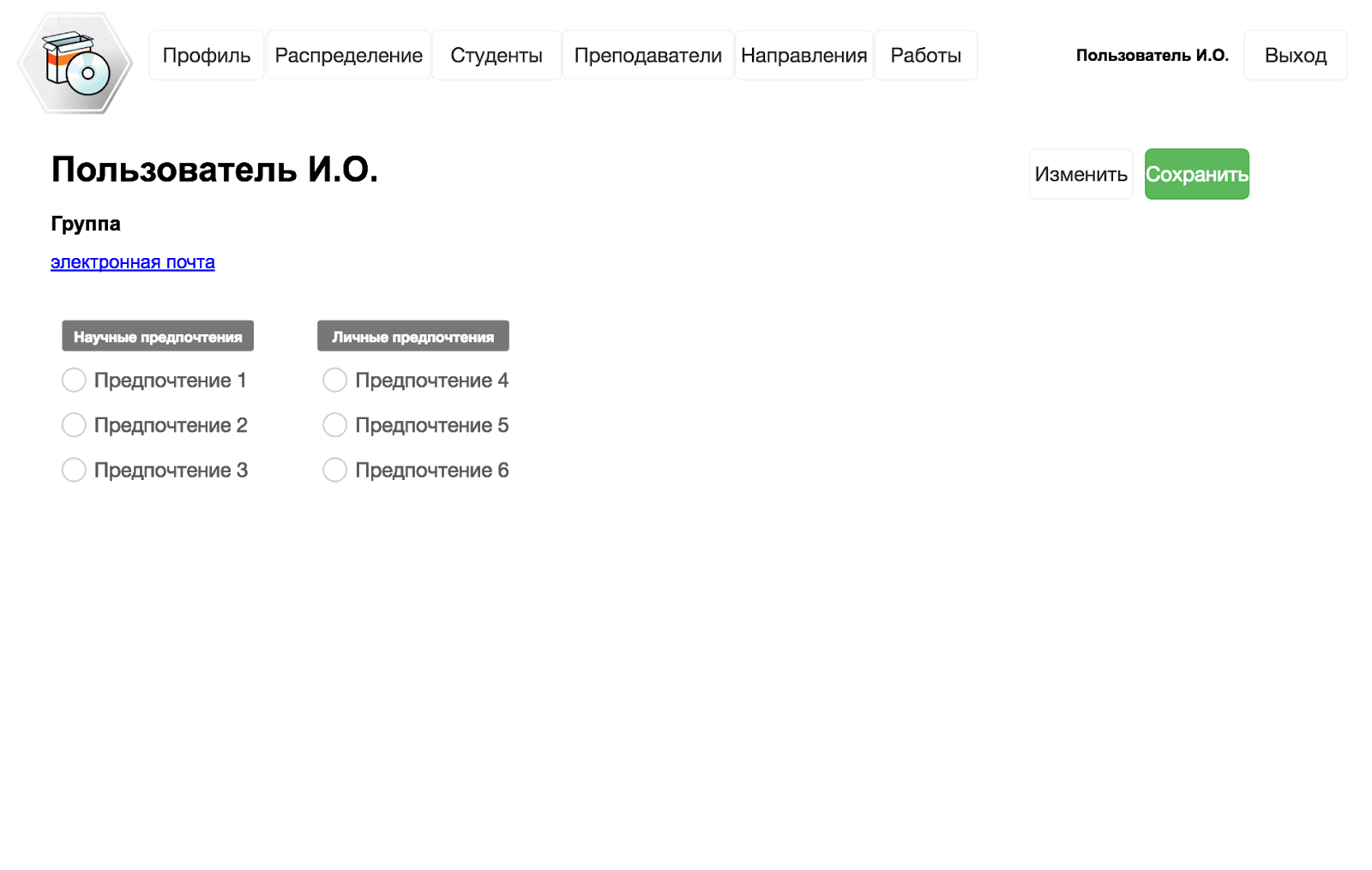


Рис. 18. Профиль студента.

Страница распределений, изображенная на рисунке 18, представляет собой список учебных групп и работ. При нажатии на необходимую группу, отображается страница распределения студентов (рисунок 19). Она представляет собой таблицу, по горизонтали которой находится список руководителей, а по вертикали студентов. Также на странице имеются ссылки на распределение других групп по данной работе.

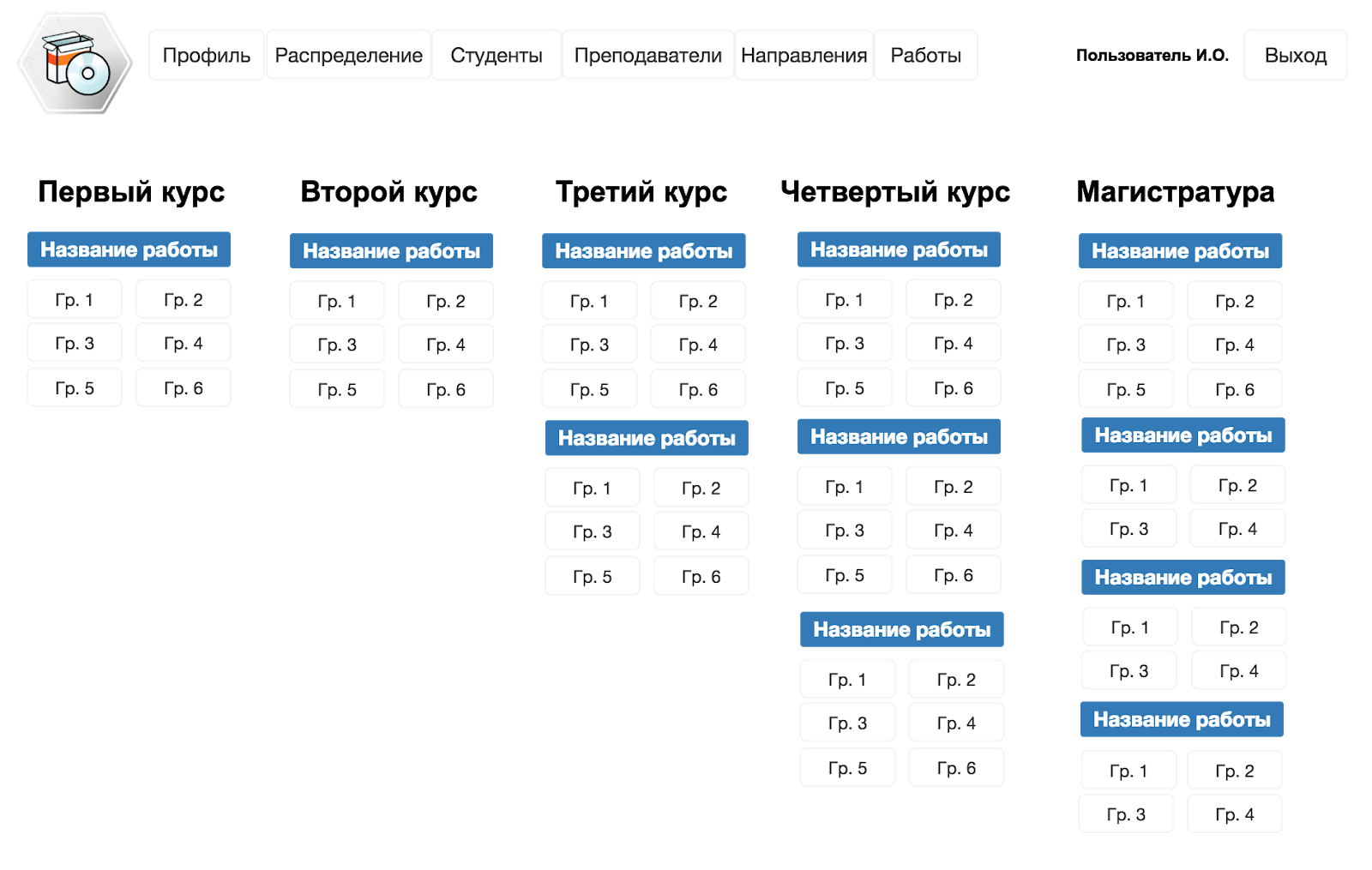


Рис. 18. Распределение (список групп).

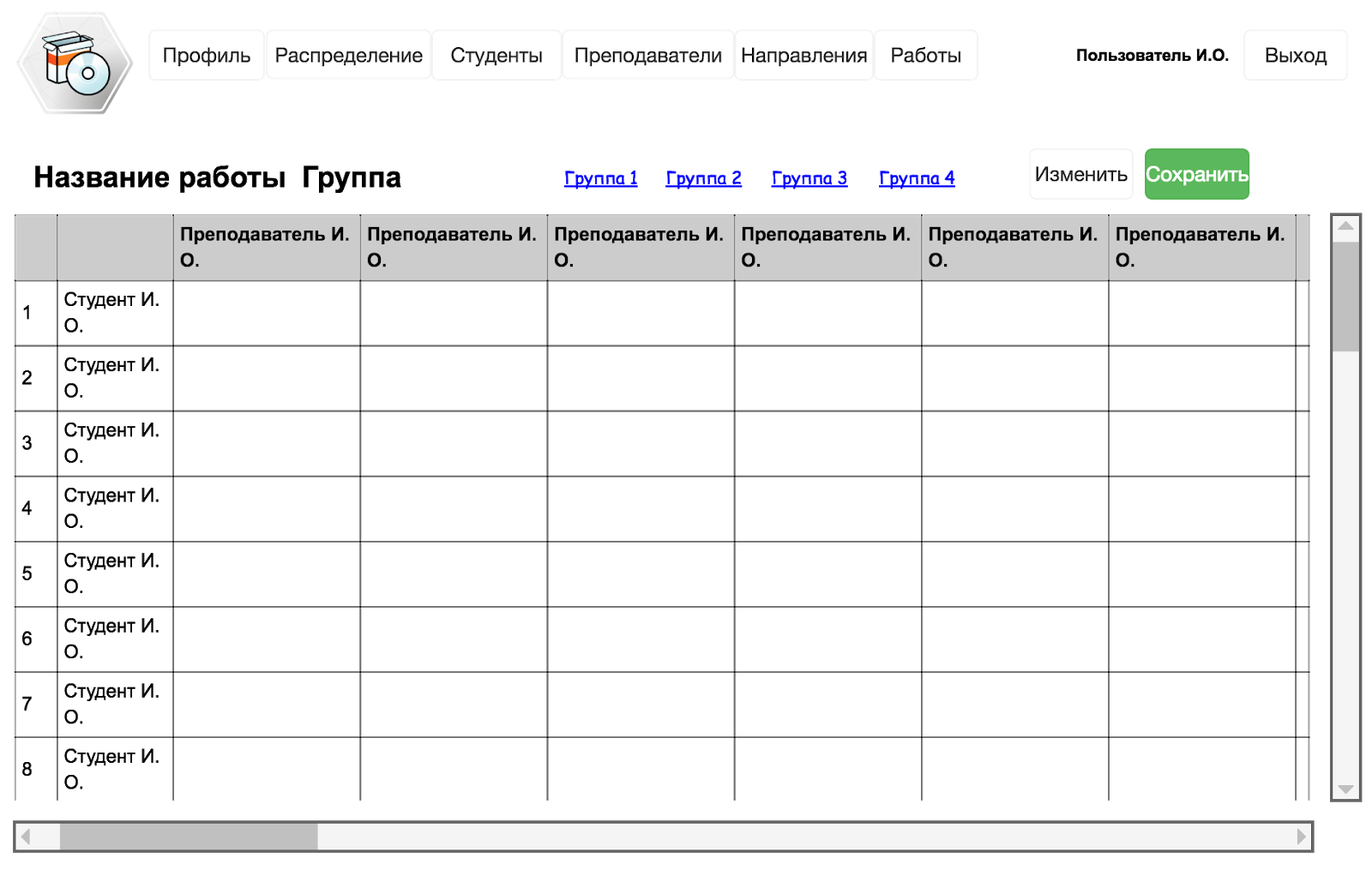


Рис. 19. Распределение (для конкретной группы и работы).

## 

## 

## Алгоритм распределения

Поставленная задача схожа с задачей о назначениях. В данном случае мы строим матрицу исходя из правил:

* первоначально создается нулевая матрица;
* если преподаватель в списке предпочтений студента добавляем 0.5 на их пересечение;
* если студент в списке предпочтений преподавателя добавляем 1 на их пересечение;
* если в списке предпочтений преподавателя есть предпочтение студента добавляем 1 на их пересечение;
* если количество преподавателей/студентов меньше количества студентов/преподаватель,добавить количество преподавателей/ студентов в соответствующий список, пока не получится квадратная матрица.

Применяем к полученной матрице Венгерский алгоритм. Система независимых нулей полученная из него будет означать искомое распределение.

# Технологический раздел

Согласно требованиям технического задания, разрабатываемая система должна быть распределенной. Характерной особенностью распределенных систем является высокое многообразие используемых технологий, языков программирования и библиотек. Особенно непростая ситуация возникает, когда разные приложения используют разные версии одной и той же библиотеки.

## Выбор языка программирования

Проанализируем техническое задание на разработку портала. Исходя из приведенных требований к системе, можно выявить требования к языку программирования:

**Наличие разнообразных библиотек**. Использование готовых библиотек ускоряет разработку программного обеспечения. Также важно, что благодаря использованию распространенных протестированных библиотек, снижается вероятность ошибки. Это повышает надежность программного обеспечения. К этому же требованию можно отнести наличие стандартных средств управления библиотеками.

Совместимость с выбранными ранее технологиями. Выбранный язык должен уметь взаимодействовать с ОС Linux, СУБД PostgreSQL.

**Высокая скорость разработки.** На ранних этапах разработки портала технические требования часто меняются. Язык программирования должен позволять как можно быстрее вносить изменения в коды программ . Это позволяет успешнее конкурировать с аналогичными сервисами.

Исходя из вышеизложенного был выбран язык Python. Современный Python – это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется. Относится к классу языков с динамической типизацией. Для него существуют различные фреймворки (Django, Pyramid, TurboGears) для разработки веб-приложений. Python активно развивается.

По сравнению с C/C++ этот язык избавляет от ручного управления памятью. Но при этом скорость его работы ниже.

Python позволяет вести разработку быстрее за счет высоко уровневости языка, а наличия большого числа библиотек для разработки серверных приложений дает ему дополнительные конкурентные преимущества.

Важными преимуществами языка Python являются возможность создавать на нем кроссплатформенные приложения. Кроме того, он является свободно распространяемым языком программирования, что также является большим преимуществом.

## Выбор фреймворка

К выбору фреймворка в нашей работе предъявляются те же требования, что и к выбору языка программирования:

* большое число стандартных возможностей;
* совместимость с выбранными ранее технологиями;
* высокая скорость разработки.

Для разработки системы будем использовать фреймворки, основанные на парадигме MVC (Мodel-View-Controller, модель - вид - элемент управления):

* модель представляет объекты, хранимые в системе: информацию о пользователях, данные платежных документов, информацию об активных сессиях;
* вид отвечает за визуализацию моделей. К примеру, для мобильной версии портала и для портала, ориентированного на ПК, могут применяться разные варианты отображения;
* элемент управления отвечает за взаимодействие пользователя и программного обеспечения.

Django — свободно распространяемый фреймворк для разработки серверных приложений на языке Python. Фреймворк Django позволяет разрабатывать серверные приложения по парадигме MVC.

Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV). Модели отображают информацию о данных, с которыми происходит работа. Они содержат поля и поведение данных. Обычно одна модель представляет одну таблицу в базе данных.

Среди аналогов Django стоит отметить Pyramid, но его возможности сильно ограничены по сравнению с Django.

По сравнению с аналогичными фреймворками Django обладает следующими преимуществами:

* проработанная документация (существует и на русском языке);
* гибкая настройка проекта
* проработанная система обработки запросов.
* разнообразие возможностей.

В результате проведенного анализа для разработки портала был выбран

фреймворк Django.

## Выбор операционной системы

Согласно требованиям технического задания, разрабатываемый портал должен обладать высокой доступностью, работать на типичных архитектурах ЭВМ (Intel x86, Intel x64), а также быть экономически недорогим для сопровождения. Таким образом, можно сформулировать следующие требования к операционной системе:

* **Распространенность.** На рынке труда должно быть много специалистов, способных администрировать распределенную систему, работающую под управлением выбранной операционной системы.
* **Надежность**. Операционная система должна широко использоваться в стабильных проектах, таких как Mail.Ru, yander.ru, Google.com. Эти компании обеспечивают высокую работоспособность своих сервисов, и на их опыт можно положиться.
* **Наличие требуемого программного обеспечения.** Выбор операционной системы не должен ограничивать разработчиков в выборе программного обеспечения, библиотек.
* **Цена.**

Под данные требования лучше всего подходит ОС Ubuntu.Ubuntu — это дистрибутив, использующий ядро Linux. Как и все дистрибутивы Linux, Ubuntu является ОС с открытым исходным кодом, бесплатным для использования.. Ubuntu простой в эксплуатации дистрибутив, имеющий систему управления пакетами (APT). Поставляется как в клиентской (с графическим интерфейсом), так и в серверной (без графического интерфейса) версиями.

## Выбор СУБД

Проанализируемтехническое задание на разработку системы. Из его сценариев использования можно выделить следующие важные моменты:

* **Структура может меняться.** В процессе развития портала могут появляться новые поля. По этой причине лучше всего использовать СУБД, которая не задает жесткую схему хранения данных.
* **СУБД должна быть быстрой и масштабируемой.** По этой причине СУБД должна иметь высокую скорость отклика, поддерживать репликацию и щардинг.

Реляционная СУБД PostgreSQL полностью отвечает приведенным выше требованиям. В качестве языка запросов используется язык, в основе которого лежит SQL. SQL легок для освоения, что снижает порог вхождения для эксплуатации этой СУБД. Эта СУБД поддерживает индексацию, профилирование запросов (выявление статистики потребления ресурсов запросами), репликацию, шардирование.

В данном проекте выбрана PostgreSQL. Данная СУБД использует асинхронную репликацию (при асинхронной репликации изменение данных в репликах базы данных происходит асинхронно, то есть не обязательно во время транзакции, возможно, после ее завершения) типа «ведущий – ведомый» (данные может изменять только «ведущий», читать данные может как «ведущий», так и «ведомый»).

# Заключение

В ходе выполнения курсового проекта был разработан алгоритм распределения руководителей курсовых и дипломных работ по студентам на основании научных и личных предпочтений. Этот алгоритм был реализован как составная часть кафедрального портала, представляющего собой систему распределенной обработки информации.

# Список литературы

1. Исаев Г.Н. Проектирование информационных систем. – М.: Издательство “Омега-Л” , 2013. – 424 с.

2. Вишневская Т.И., Романова Т.Н. Технология программирования: Мет. указания к лабораторному практикуму. - Ч. 1. – М: Изд-во МГТУ им. Н.Э. Баумана, 2007, 59с.

3. Вишневская Т.И., Романова Т.Н. Технология программирования: Мет. указания к лабораторному практикуму. - Ч. 2. – М: Изд-во МГТУ им. Н.Э. Баумана, 2010, 46с.

4. Вишневская Т.И., Романова Т.Н. Технология программирования: Мет. указания к лабораторному практикуму. - Ч. 3. – ФГБОУ ВПО МГТУ им. Н.Э. Баумана, 2012 , №0321203886. [Электронный ресурс] Режим доступа: URL:http://catalog.inforeg.ru/Inet/GetEzineByID/293408

5. Техническое задание. Требования к содержанию и оформлению (ГОСТ 19.201-78). [Электронный ресурс] Режим доступа: URL: http:/​/​fmi.asf.ru/​library/​book/​Gost/​19201-78.html.

6. Python. [Электронный ресурс] Режим доступа: www.python.su