Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Курсовая работа

по курсу «Технологии разработки программного обеспечения»

Тема: «Файловое хранилище»

Выполнил
Студент группы **ИУ5-28**
Ларионов Вадим

Москва — 2017

Содержание

1 Общее описание

1.1 Цель работы

Изучить теоретические принципы унифицированного процесса разработки СОИУ и составляющих его этапов. Получить практические навыки применения шаблонов при проектировании и разработке СОИУ. Освоить применение CASE средств для разработки СОИУ.

1.2 Постановка задачи

Выполнить разработку СОИУ в соответствии с описанием её функциональности (определяется вариантом) на основе моделей унифицированного процесса (RUP). Написать программу, реализующую фрагмент СОИУ, и реализовать в ней паттерны бизнес-логики, работы с БД, gof (по варианту).

1.2.1 Задание по варианту

Пользователи имеют свои каталоги, в которые загружают файлы. Для каждого файла создаётся его паспорт, хранящийся в базе данных. Паспорт содержит название, описание, атрибуты файла и набор дополнительных атрибутов. Реализовать подсистему АРМ администратора, в обязанности которой входит: ведение БД пользователей и групп. Назначение пользователям и группам прав доступа к каталогам.

АСУ: файловое хранилище.

Подсистема: АРМ администратора.

Паттерны: модель предметной области (бизнес-логика), активная запись (обращение к БД), приспособленец (GoF).

2 Этап анализа и планирования требований

2.1 Спецификация основных проектных требований

2.1.1 Функциональные требования

К функциональным требованиям, представляемым к подсистеме относятся:

1) Ведение БД пользователей;
2) Ведение БД групп;
3) Ведение БД каталогов;
4) Ведение БД авторизационных сессий пользователей;
5) Отображение групп в системе (через поиск);
6) Отображение пользователей в системе (через поиск);
7) Отображение каталогов пользователя;
8) Отображение каталогов, относящихся к группам, к которым принадлежит пользователь;
9) Отображение каталогов в системе с возможностью разграничения прав доступа (чтение, запись, чтение и запись).

2.1.2 Нефункциональные требования

К нефункциональным требованиям, представляемым к подсистеме относятся:

1) Удобный, минималистичный и эргономичный интерфейс клиентской части;
2) Доступность клиентской части через браузер;
3) Стабильная работа с базой данных;
4) Уведомление пользователя об ошибках ввода;
5) Возможность масштабирования системы (до 10 физических либо виртуальных серверов);
6) Максимальное время загрузки html-страницы — 200 мс.

2.2 Модель предметной области

2.2.1 Диаграмма классов предметной области

Диаграмма классов предметной области изображена на рис. 2.1. Основные классы, выделенные в подсистеме:

- Пользователь — хранит информацию о пользователе в системе, в том числе и администраторе;
- Сессия — служит для обеспечения механизма авторизации и идентификации администратора в системе;
- Группа — оперирует с группами, в которые входят пользователи;
- Каталог — оперирует с каталогами, которые создают пользователи;
- ГруппаКаталог — вспомогательный класс, служащий для разграничения прав доступа к каталогам для групп.

Рисунок 2.1 — Диаграмма классов предметной области

2.2.2 Бизнес-модели

Описание проекта выполнено с помощью канвы бизнес-модели Остервальдера (рис. 2.2).

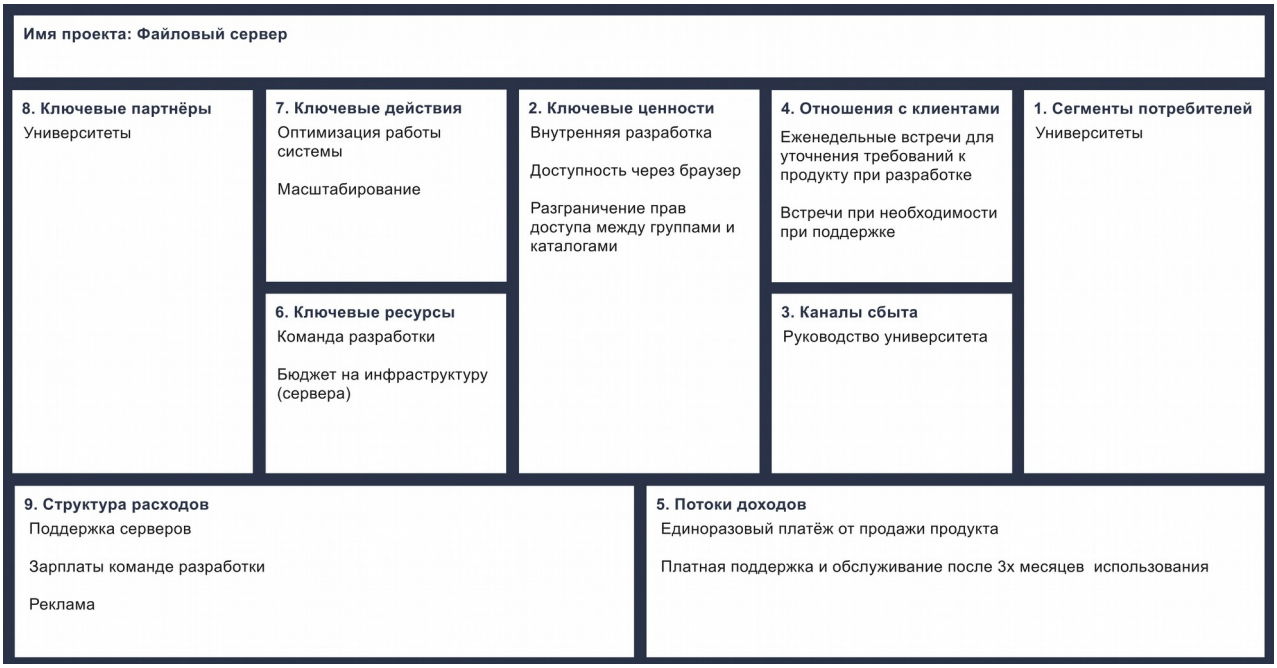

Рисунок 2.2 — Бизнес-модель

2.3 Актёры

Для подсистемы АРМ администратора файлового сервера выявлен один актёр — **администратор** системы. Он занимается созданием новых пользователей, групп, каталогов; разграничивает права доступа между каталогами и группами.

2.4 Выявленные прецеденты, их приоритеты и описание

По результатам проведённого анализа выделены следующие прецеденты:

— **Создать пользователя** — создание нового пользователя в системе, который может быть обычным либо администратором. Пользователи могут иметь свои каталоги, принадлежать к группам, загружать файлы.

— **Создать группу** — создание новой группы. К одной группе может принадлежать несколько пользователей, пользователь может входить в несколько групп. Для группы предоставляются права доступа к каталогам;

— **Добавить пользователя в группу** — администратор добавляет пользователя в группу. После для пользователя становятся доступны каталоги, к которым имеет права доступа группа;

— **Удалить пользователя из группы** — запрет просмотра и редактирования каталогов, к которым имеет права группа, из которой был удалён пользователь;

— **Выдать права доступа группе для каталога** — выдача прав для просмотра/редактирования каталогов для группы;

— **Найти пользователя** — поиск пользователя в системе;

— **Найти группу** — поиск группы в системе;

— **Найти каталог** — поиск каталога в системе.

Приоритеты основных прецедентов представлены в табл. 2.1

Таблица 2.1. Приоритет прецедентов

Прецедент	Приоритет
Создать пользователя	Высокий
Создать группу	Высокий
Добавить пользователя в группу	Высокий
Удалить пользователя из группы	Низкий
Выдать права доступа группе для каталога	Высокий
Найти пользователя	Средний
Найти группу	Средний
Найти каталог	Средний

2.5 Диаграмма основных прецедентов

Диаграмма основных прецедентов изображена на рис. 2.3.

Рисунок 2.3 — диаграмма основных прецедентов

2.6 Описание архитектуры

2.6.1 Общая архитектура приложения

Исходя из нефункциональных требований, таких как возможность масштабирования и доступность клиентской части через браузер, в качестве каркаса (framework) был выбран Django. Соответственно, язык программирования Python 3. Использование этого языка и фреймворка позволяется в относительно короткие сроки сделать готовое приложение (относительно других языков программирования, таких как C/C++, Java). Приложения, разработанные с помощью Django, без особых сложностей масштабируются на десятки серверов. Также к плюсам каркаса можно отнести положительные опыт использования в крупных проектах, таких как Instagram, Pinterest, Bitbucket, Disqus.

Использование фреймворка диктует использование его архитектуры. Для Django это MVC (Model-View-Controller). Шаблон MVC позволяет отделить бизнес логику (модель) от её визуализации (представление), что упрощает повторное использование кода. Основная цель применения этой концепции состоит в отделении бизнес-логики (модель либо контроллер, зависит от реализации) от её визуализации (представление, вид), что упрощает повторное использование кода и особенно полезно, если пользователь должен видеть одни и те же данные в различных контекстах и (или) с различных точек зрения одновременно.

Для передачи данных используется защищённый протокол HTTPS. В качестве веб-сервера — Nginx. Он умеет терминировать HTTPS-соединение и работать в связке с uWSGI сервером, передавая запросы в Django-приложение. Эта связка обеспечивает наиболее производительное решение.

2.6.2 Паттерн реализации бизнес-логики

Для реализации бизнес логики выбран паттерн «Модель предметной области». Он охватывает поведение (функции) и свойства (данные). Типовое решение модель предметной области предусматривает создание сети взаимосвязанных объектов, каждый из которых представляет некую осмысленную сущность. Паттерн является простым в реализации и понятным разработчикам, так как программируемые классы описывают сущности реального мира.

2.6.3 Паттерн доступа к БД

Объект, выполняющий роль оболочки для строки таблицы или представления базы данных. Он инкапсулирует доступ к базе данных и добавляет к данным логику домена. Активная запись хорошо подходит для реализации не слишком сложной логики домена, в частности операций создания, считывания, обновления и удаления. Кроме того, она прекрасно справляется с извлечением и проверкой на правильность отдельной записи.

2.7 Перечень критических рисков

В ходе анализа выделены три категории источников риска: проектные, технические и коммерческие.

2.7.1 Проектные риски

1) **Риски, связанный с командой разработки** — члены команды разработки могут заболеть либо уйти в другие проекты. Для того, чтобы потеря участника не наносила огромных потерь проекту, как можно большее количество других разработчиков должны хорошо ориентироваться в проекте и при наличии ошибок уметь их быстро находить и исправлять. Это, так называемое, повышение bus factor проекта. Для уменьшения влияния риска стоит использовать системы документирования кода, фиксировать знания в специальных системах (например, Confluence), поддерживать документацию в актуальном

состоянии и проводить между разработчиками практику просмотра кода (code review).

2) **Риск изменения требований к программному продукту** — для того, чтобы конечный вариант продукта соответствовал ожиданиям заказчика, необходимо как можно чаще уточнять требования у заказчика, показывая промежуточные варианты. ТЗ должно быть составлено наиболее полно, все изменения фиксируются в ТЗ.

2.7.2 Технические риски

1) **Риск недостаточной оценки сложности** — на этапах Начало и Конструирование могут возникнуть сложности связанные с проектированием системы. Для их устранения может понадобиться консультации более квалифицированных специалистов, издержки на которых должны быть учтены в бюджете проекта;
2) **Риск неверной реализации** — на этапах Конструирование и Переход может выясниться, что система не отвечает всем заявленным требованиям, либо содержит ошибки. Для минимизации риска следует как можно чаще проводить тестирование на соответствие требований, интеграционное тестирование модулей, тестирование стабильности и надёжности, а также нагрузочное тестирование.

2.7.3 Коммерческие риски

1) **Риск создания ненужного продукта** — за время разработки у заказчика может исчезнуть потребность в разрабатываемом продукте. Если такое случится, то лучше определить это как можно раньше. Для этого в короткие сроки разрабатывается минимально жизнеспособный продукт (MVP). Он реализует основные функции, предъявляемые к системе. В случае успешного запуска MVP продолжается его разработка до конечной системы.

2) **Риск потери финансирования** — для минимизации этого риска необходимо предусмотреть дополнительную статью в бюджете.

2.8 Оценка проекта по COCOMO II этапа композиции приложения

На этапе анализа требований были выделены следующие экранные формы:

— Авторизации администратора;

— Создания пользователя;

— Создания группы;

— Добавление/удаление пользователя к группе;

— Добавления и редактирования прав доступа к каталогу для группы;

— Поиска групп или пользователей.

Для оценки стоимости, затрат и длительности проекта используется конструктивная модель стоимости (COCOMO II) композиции приложения на основе объектных указателей. Оценка объектных указателей с указанием сложности экранных форм приведена в таблице 2.2. Под количеством таблиц понимается количество таблиц БД на сервере, клиентские таблицы не используются. Отображение количества представлений и таблиц в сложность (количество объектных указателей) приведено в [3, стр. 31, табл. 2.16].

Таблица 2.2. Оценка количества объектных указателей

Экран	**Кол-во представлений**	**Кол-во таблиц**	**Сложность**
Авторизация	1	2 (Пользователь, Сессия)	x1 (простой)
Создание пользователя	1	1 (Пользователь)	x1 (простой)
Создание группы	1	1 (Группа)	x1 (простой)
Добавление/удаление пользователя к группе	1	2 (Группа, Пользователь)	x1 (простой)
Добавление и изменение прав доступа группы к каталогу	1	4 (Каталог, Группа, ГруппаКаталог, Пользователь)	x2 (средний)
Поиск групп или	1	2 (Группа, Пользователь)	x1 (простой)

пользователей			
Всего представлений	**6**	**Объектные указатели**	**7**

Разработка подобного рода системы для команды разработчиков является новой, поэтому процент повторного использования кода *%REUSE* равен 0.

Новые объектные указатели рассчитаны по формуле 2.1.

$$NOP = OP \times \frac{100 - \%REUSE}{100} = 7 \times \frac{100 - 0}{100} = 7 \quad (2.1)$$

Субъективно оценим опытность разработчика и зрелость среды разработки как номинальные, тогда **PROD** (производительность разработки, выраженная в терминах объектных указателей) составит **13** (значение взято из [3, стр. 31, табл. 2.18].

Рассчитаем проектные затраты по формуле 2.2.

$$ЗАТРАТЫ = \frac{NOP}{\text{PROD}} = \frac{7}{13} = 0{,}5385 [\text{чел.-мес}] \quad (2.2)$$

Примем, что разработчик с номинальной опытностью разработки зарабатывает 90 000 рублей в месяц, тогда стоимость разработки определяется формулой 2.3.

$$СТОИМОСТЬ = ЗАТРАТЫ \times \text{РАБ\_КОЭФ} = 0{,}5385 \times 90\,000 = 48\,465 [\text{руб.}] \quad (2.3)$$

Произведём оценку коэффициента B, который отражает нелинейную зависимость затрат от размера проекта и зависит от масштабных факторов W_i (см. таблицу 2.3).

Таблица 2.3. Характеристика масштабных факторов.

Масштабный фактор (W_i)	**Пояснение**
Предсказуемость (PREC)	Отражает предыдущий опыт организации в реализации проектов этого типа. Очень низкий означает отсутствие опыта. Сверхвысокий означает, что организация полностью знакома с этой прикладной областью
Гибкость разработки (FLEX)	Отражает степень гибкости процесса разработки. Очень низкий означает, что используется заданный процесс. Сверхвысокий означает, что клиент установил только общие цели
Разрешение архитектуры / риска (RESL)	Отражает степень выполняемого анализа риска. Очень низкий означает малый анализ. Сверхвысокий означает полный и сквозной анализ риска

Связность группы (TEAM)	Отражает, насколько хорошо разработчики группы знают друг и насколько удачно они совместно работают. Очень низкий означает очень трудные взаимодействия. Сверхвысокий означает интегрированную группу без проблем взаимодействия
Зрелость процесса (PMAT)	Означает зрелость процесса в организации. Вычисление этого фактора может выполняться по вопроснику CMM

В качестве значений масштабных факторов выберем следующее:

- Предсказуемость PREC = 2 (высокий) — большей частью знакомый. Выбор связан с тем, что команда разработки ранее не занималась построением систем администрирования файловых серверов, но имеет опыт разработки в смежных областях;
- Гибкость разработки FLEX = 1 (очень высокий) — некоторое согласование процесса. Выбор связан с тем, что заказчик определил общие цели (требуемые прецеденты), а также некоторые технические аспекты разрабатываемой системы, например, шаблоны проектирования;
- Разрешение архитектуры / риска RESL = 1 (очень высокий) — почти всегда. Выбор связан с тем, что произведён глубокий и тщательный анализ рисков, но оставляются 10% неисследованных рисков (субъективно) на форс-мажорные ситуации;
- Связность группы TEAM = 1 (очень высокий) — высокая кооперативность. Выбор сделан на основании предыдущего опыта: команда разработки хорошо друг друга знает и имеет опыт совместной разработки;
- Зрелость процесса PMAT = 3 (номинальное) — номинальная зрелость. В работе [3, стр. 33] автор предлагает использовать взвешенное среднее значение от количества ответов «Yes» на вопросник CMM Maturity. Ввиду отсутствия в организации ведения опросника CMM, выбрано номинальное значение параметра.

Тогда значение показателя B, отражающего нелинейную зависимость затрат от размера проекта, можно рассчитать по следующей формуле:

$$B = 1{,}01 + 0{,}01 \cdot \sum_{i=1}^{5} W_i = 1{,}01 + 0{,}01 \cdot (2+1+1+1+3) = 1{,}01 + 0{,}01 \cdot 8 = 1{,}09 \quad (2.4)$$

$$\textit{Длительность}(TDEV) = [3{,}0 \times \textit{ЗАТРАТЫ}^{0{,}33 + 0{,}2 \cdot (B - 1{,}01)}] = 2{,}4\,[\text{мес}] \quad (2.5)$$

Таким образом, учитывая, что подсистему администратора файлового сервера создаёт один разработчик, получим следующие результаты (таблица 2.4).

Таблица 2.4. Рассчитанные значения затрат, длительности и стоимости разработки

Параметр	**Значение**
Затраты	0,5385 чел.-мес
Длительность разработки	2,4 мес
Стоимость разработки	48 465 руб.
Скорость разработки (PROD)	13

3 Этап проектирования (развитие)

3.1 Прототип пользовательского интерфейса

Прототипы пользовательского интерфейса изображены на рис. 3.1 — 3.3.

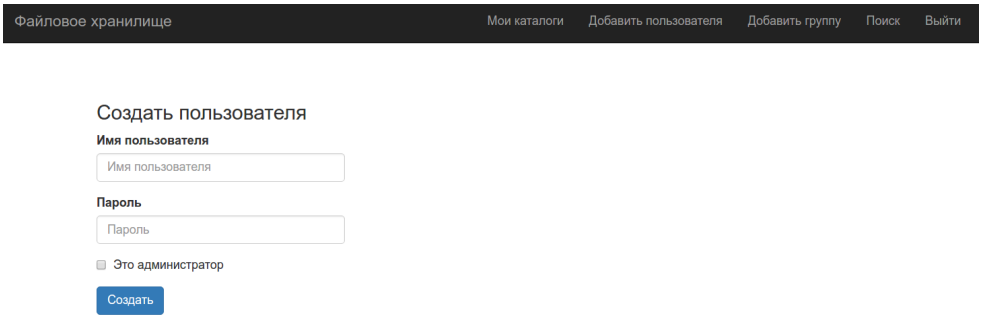

Рисунок 3.1 — Прототип пользовательского интерфейса экрана создания пользователя

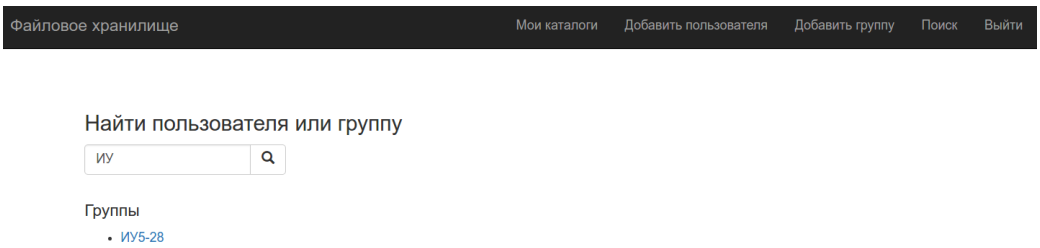

Рисунок 3.2 — Прототип пользовательского интерфейса экрана поиска групп и пользователей

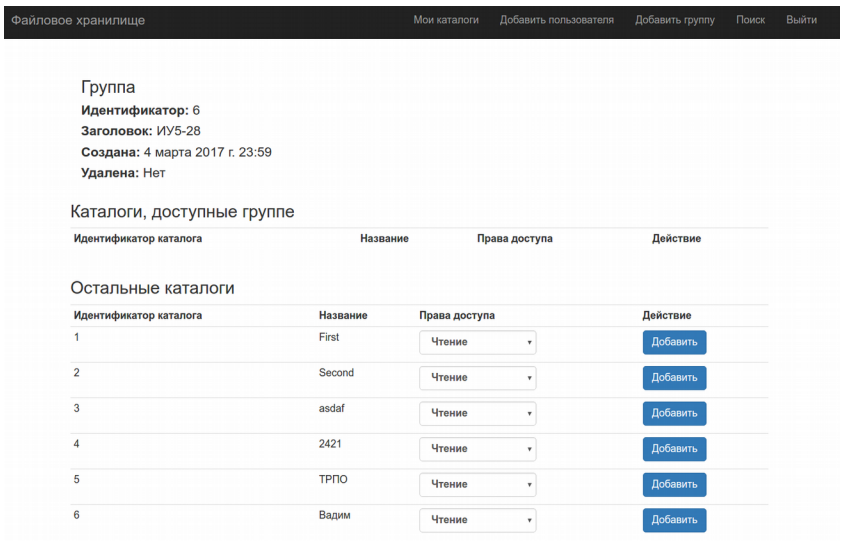

Рисунок 3.3 — Прототип пользовательского интерфейса экрана редактирования прав доступа для группы к каталогам

3.2 Уточнённая диаграмма прецедентов

Уточнённая диаграмма прецедентов изображена на рис. 3.4. Её словесное описание представлено в разделе 3.3.

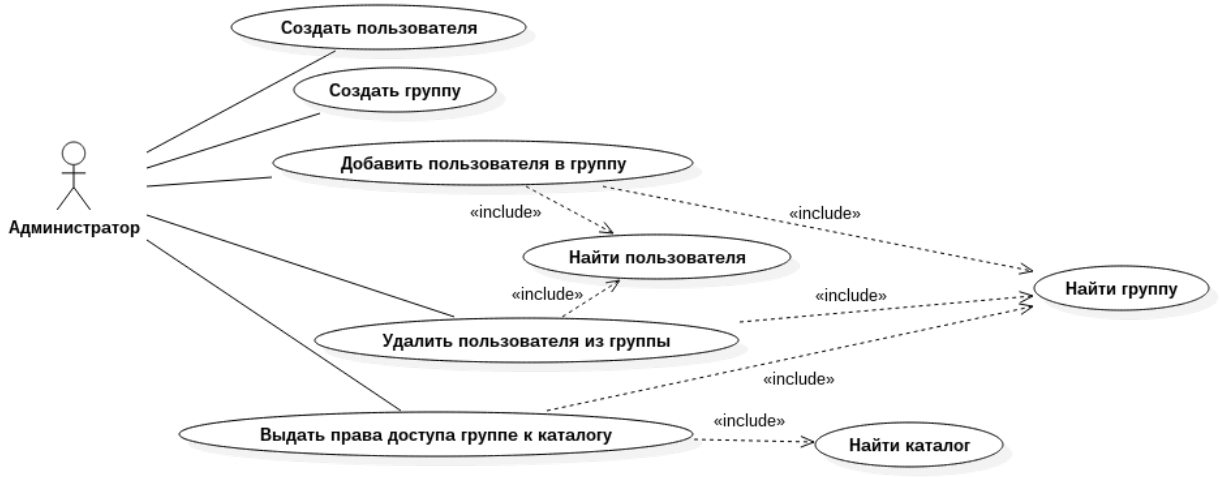

Рисунок 3.4 — Уточнённая диаграмма прецедентов

3.3 Расширенные описания основных прецедентов

3.3.1 Спецификация прецедента «Создать пользователя»

Предусловие: система должна содержать хотя бы одного администратора; администратор должен быть авторизован.

Основной поток

Выбрать пункт «Добавить пользователя». Ввести имя пользователя (E-1) для нового пользователя, ввести пароль (E-2) для нового пользователя, выставить флажок «Это администратор» (при необходимости). Нажать кнопку «Создать» (E-3). Система сохраняет нового пользователя в системе.

Альтернативные потоки для прецедента «Создать пользователя»

E-1: Введено недопустимое имя пользователя (минимальная длина имени — 4 символа). Администратор может повторить ввод с новым именем пользователя или прекратить use case.

E-2: Введён недопустимый пароль. Администратор может повторить ввод с новым паролем или прекратить use case.

E-3: Такой пользователь уже существует. Администратор может повторить ввод с новыми данными либо прекратить use case.

3.3.2 Спецификация прецедента «Добавить пользователя в группу»

Предусловие: система должна содержать хотя бы одного администратору, группу и пользователя, которого требуется добавить в группу; администратор должен быть авторизован.

Основной поток

Администратор выбирает пункт «Поиск», на открывшейся странице в поле ввода вписывает имя пользователя, которого необходимо добавить в группу, нажимает кнопку «Найти». В списке из найденных пользователей выбирает нужного (E-1). В открывшемся окне выбирает из списка всех групп выбирает целевую (E-2). Рядом с выбранной группой нажимает на кнопку «Добавить в группу». Система сохраняет изменения.

E-1: Введённый пользователь отсутствует в системе. Администратору следует проверить введённое имя пользователя либо создать нового пользователя. Данный use case можно продолжить, изменив имя пользователя на корректное, либо завершить.

E-2: Целевой группы нет в списке. Администратору необходимо сначала создать требуемую группу. Use case можно продолжить, если требуется добавить пользователя в другую группу, либо завершить его.

3.4 Диаграммы классов анализа

3.4.1 Диаграмма кооперацій

Диаграмма кооперацій разрабатываемой системы для основных прецедентов представлена на рис. 3.5.

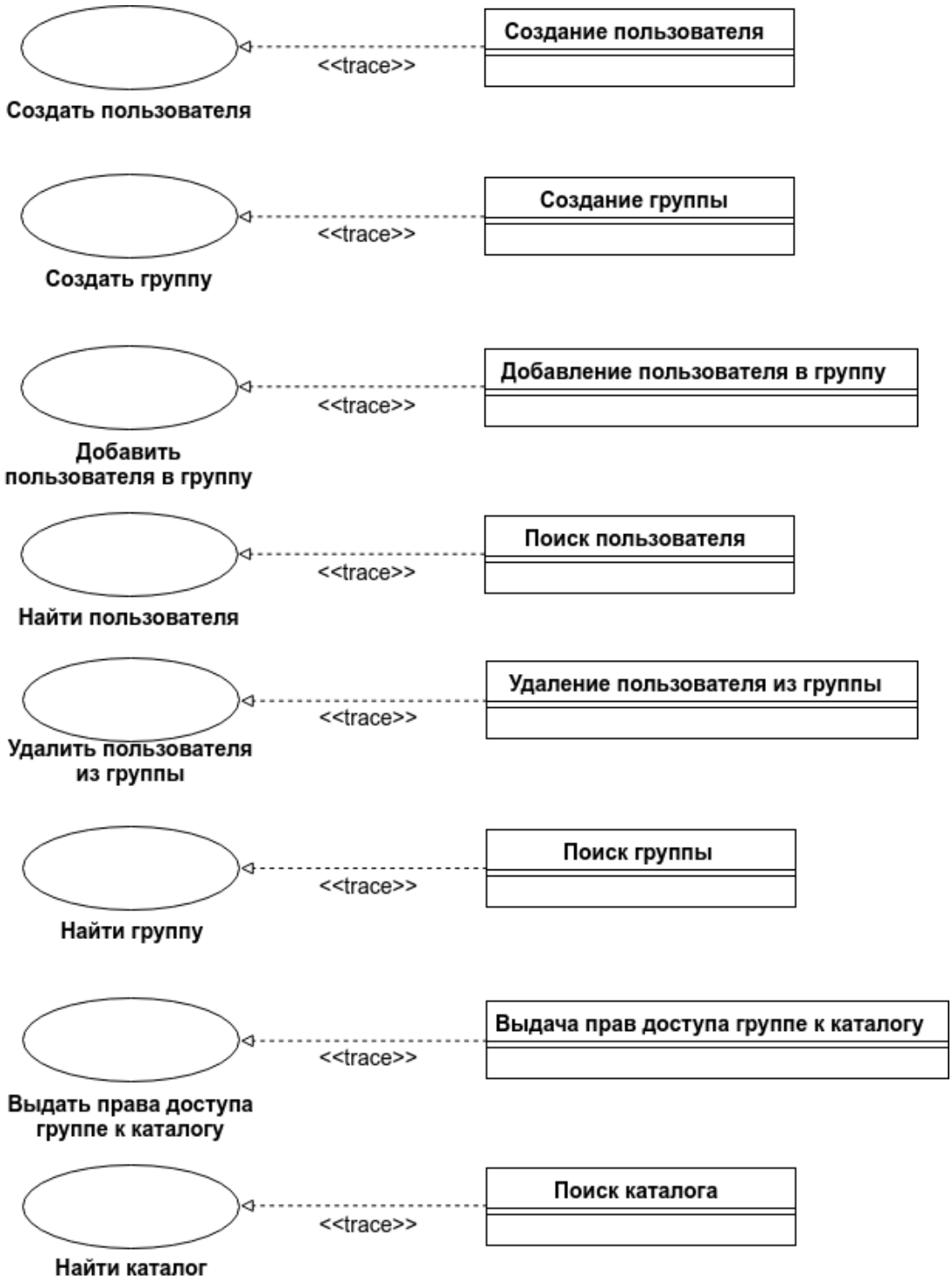

Рисунок 3.5 — Диаграмма кооперацій

3.4.2 Граничные классы

Граничный класс Администратор изображен на рис. 3.6.

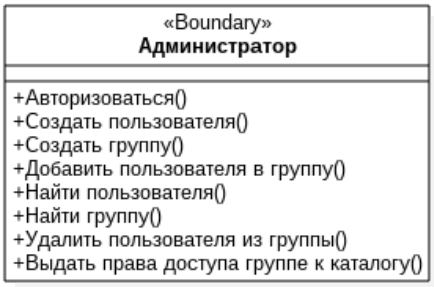

Рисунок 3.6 — Диаграмма граничных классов

3.4.3 Управляющие классы

Управляющие классы составлены в соответствии с основными прецедентами. Каждый управляющий класс на этапе анализа реализует 1 прецедент. Их диаграмма изображена на рис. 3.7.

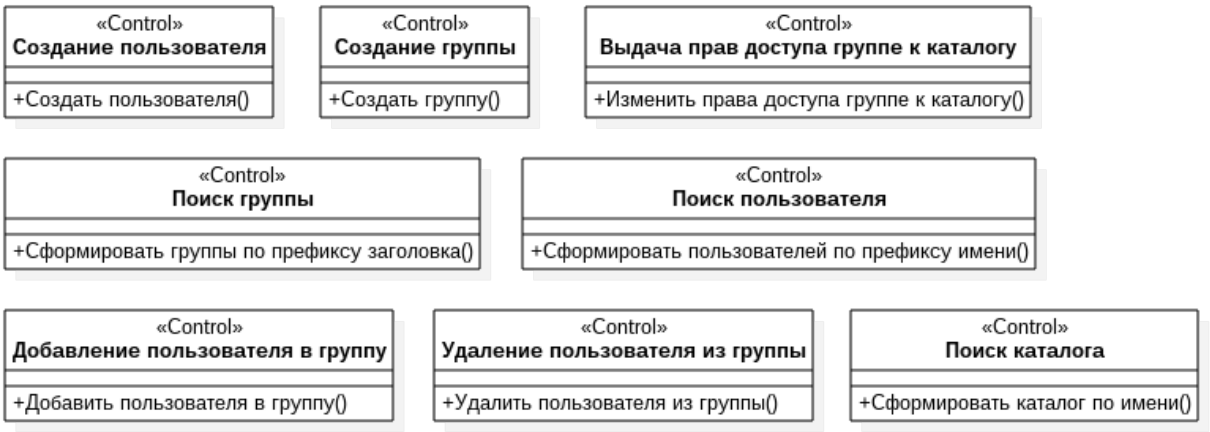

Рисунок 3.7 — Диаграмма управляющих классов

3.4.4 Классы сущностей

На рис. 3.8 изображена диаграмма классов сущностей. Классы, изображённые на диаграмме:

- Сессия — класс с информацией об авторизационной сессии пользователя;
- Пользователь — класс с персональной информацией о пользователе;
- Группа — класс с информацией о группе;

- Каталог — класс с информацией о каталоге. Каталог создаёт пользователь (т. е. является его автором), доступ к каталогам предоставляется для групп;
- ГруппаКаталог — содержит права доступа к каталогу для группы.

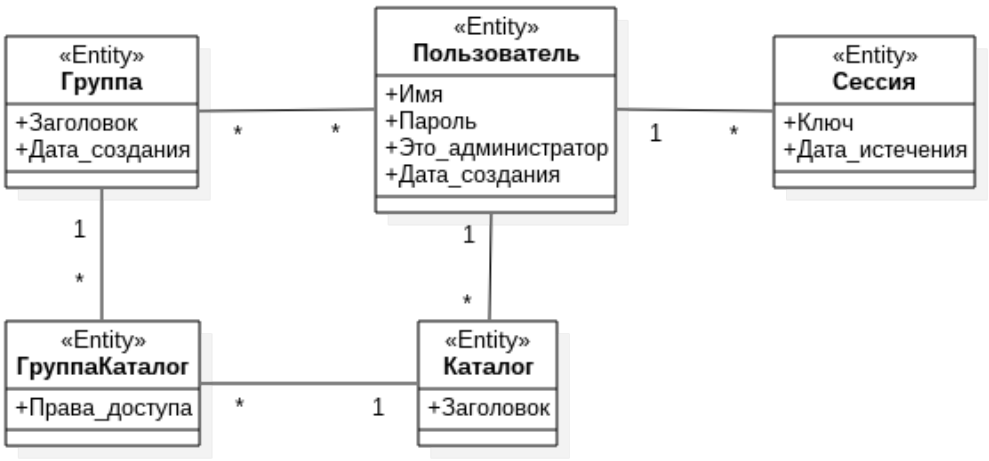

Рисунок 3.8 — Диаграмма классов сущностей

3.5 Диаграммы взаимодействия

3.5.1 Кооперация «Создать пользователя»

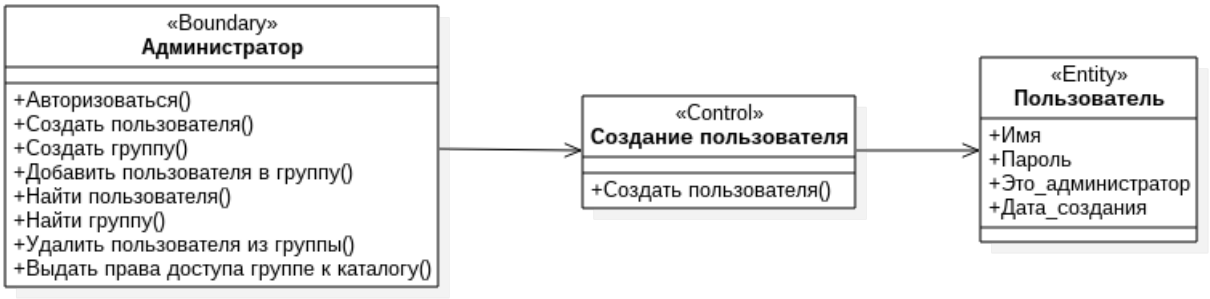

Рисунок 3.9 — Диаграмма используемых классов для кооперации «Создать пользователя»

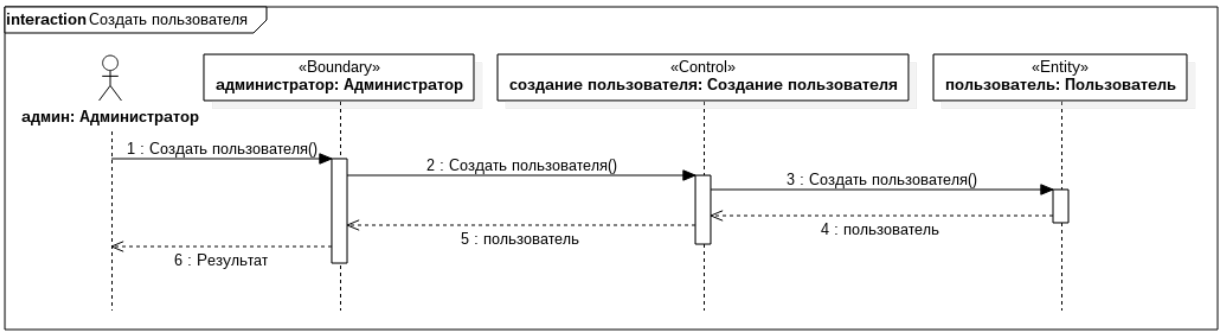

Рисунок 3.10 — Диаграмма взаимодействие для кооперации «Добавить пользователя»

3.5.2 Кооперация «Найти пользователя»

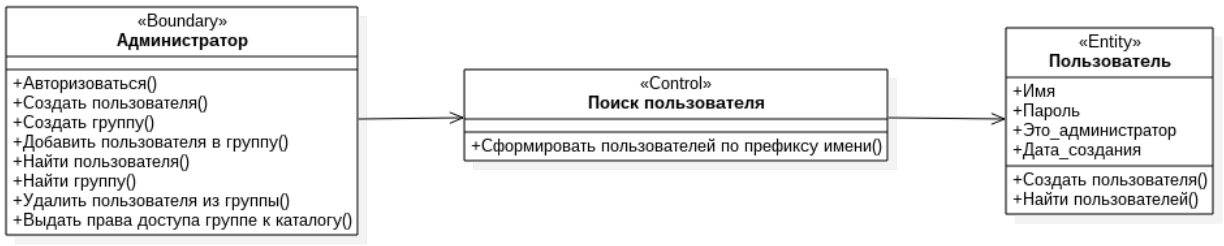

Рисунок 3.11 — Диаграмма используемых классов для кооперации «Найти пользователя»

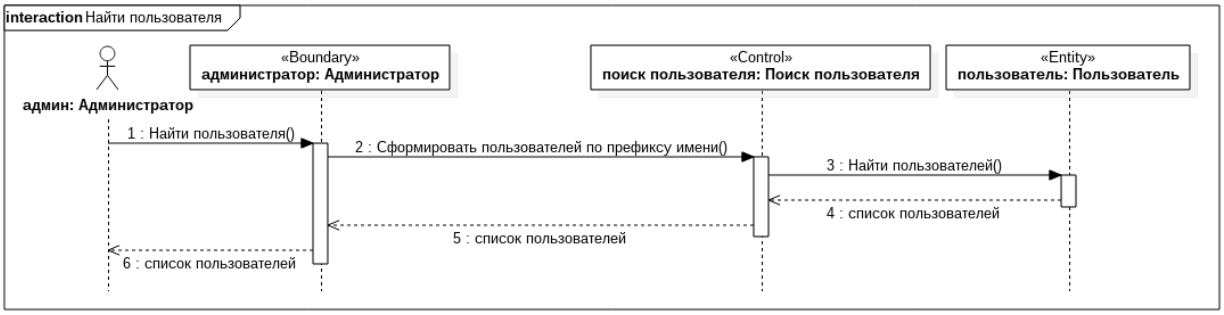

Рисунок 3.12 — Диаграмма взаимодействия для кооперации «Найти пользователя»

3.5.3 Кооперация «Добавить права доступа группе к каталогу»

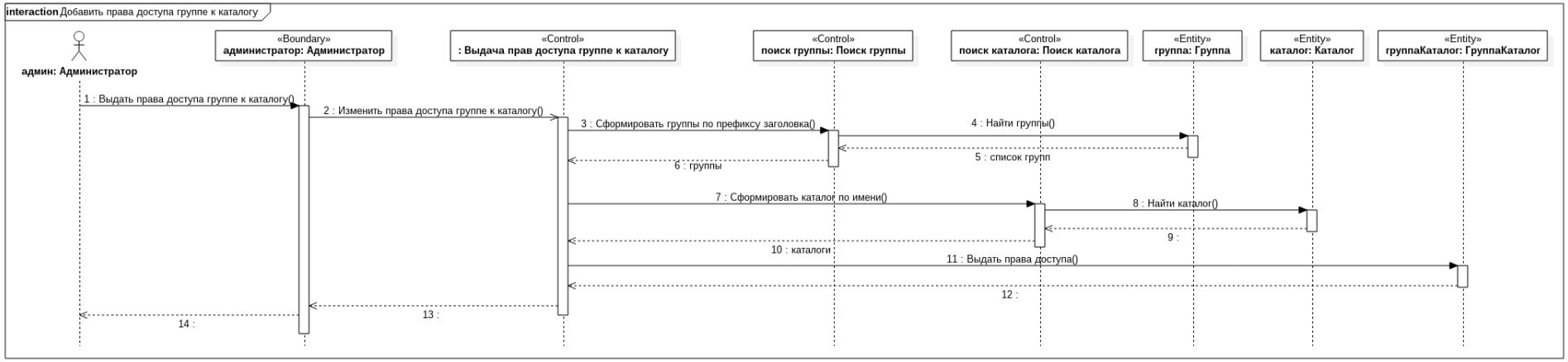

Рисунок 3.13 — Диаграмма последовательности для кооперации «»Добавить права доступа группе к каталогу»

3.6 Пакеты анализа

3.6.1 Пакет «Пользователь»

На рисунке 3.14 представлен пакет анализа «Пользователь». В него входят 2 класса сущностей — «Пользователь» и «Сессия», а также два управляющих класса «Поиск пользователя» и «Создание пользователя».

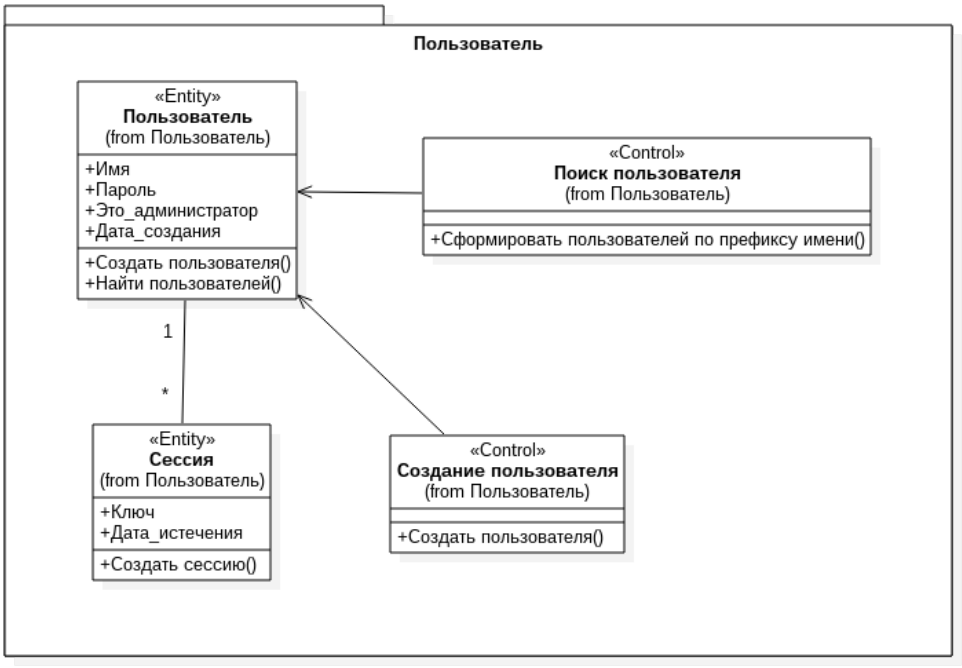

Рисунок 3.14 — Диаграмма пакета анализа «Пользователь»

3.6.2 Пакет «Группа»

Диаграмма пакета анализа «Группа» изображена на рис. 3.15. В него входят 4 управляющих класса: «Создание пользователя», «Добавление пользователя в группу», «Удаление пользователя из группы» и «Поиск группы», а также класс сущностей — «Группа».

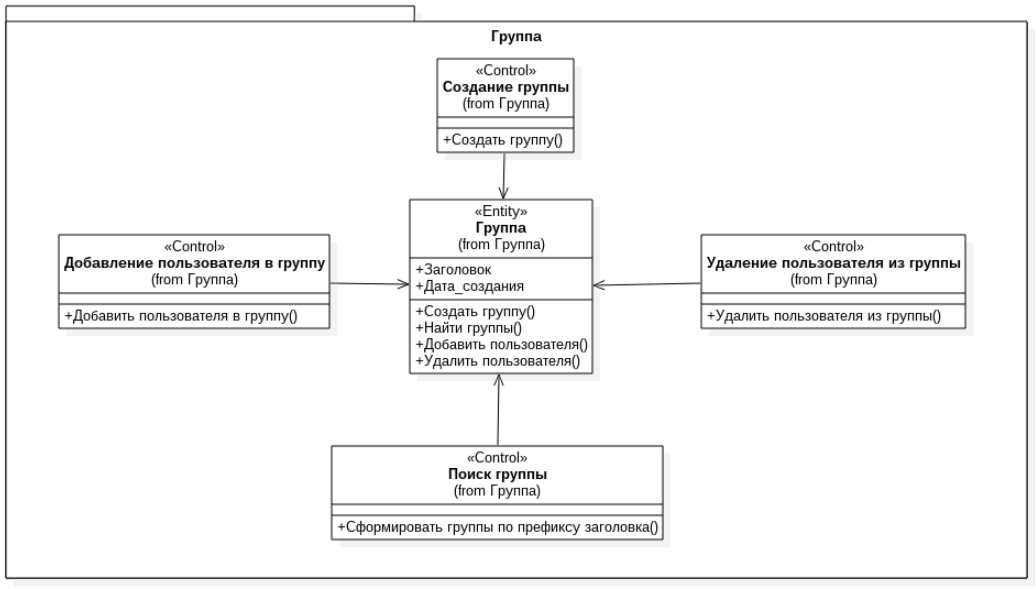

Рисунок 3.15 — Диаграмма пакета анализа «Группа»

3.6.3 Пакет «Каталог»

Диаграмма пакета каталог представлена на рис. 3.16.

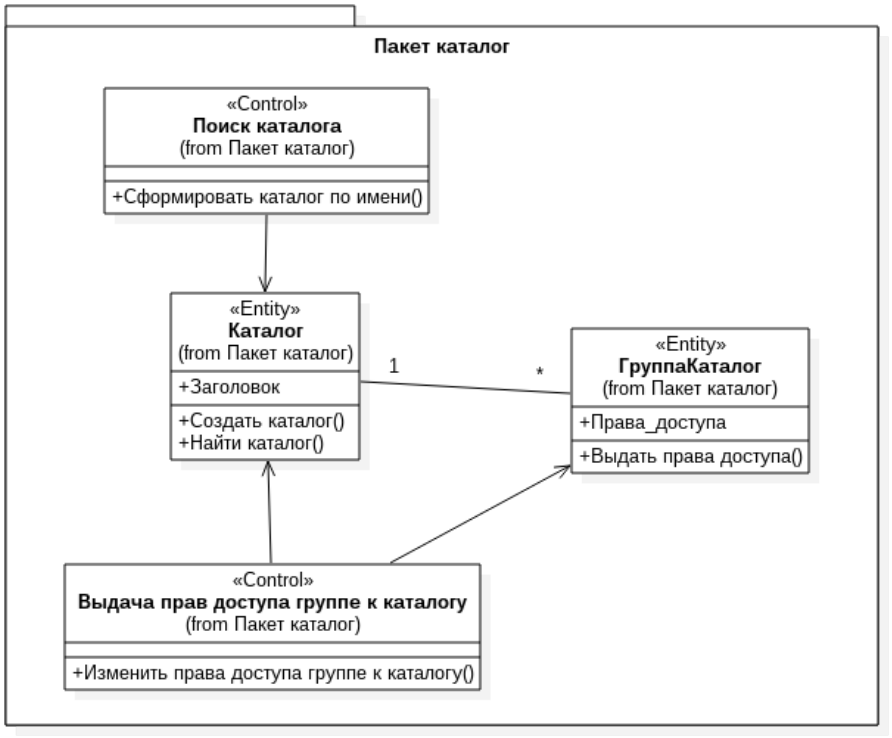

Рисунок 3.16 — Диаграмма пакета анализа «Каталог»

3.6.4 Пакет «Актёр»

Пакет анализа «Актёр» создан с целью объединения всех граничных классов в один пакет. Поскольку в подсистеме администрирования файлового сервера 1 актёр, то пакет состоит из единственного граничного класса «Администратор».

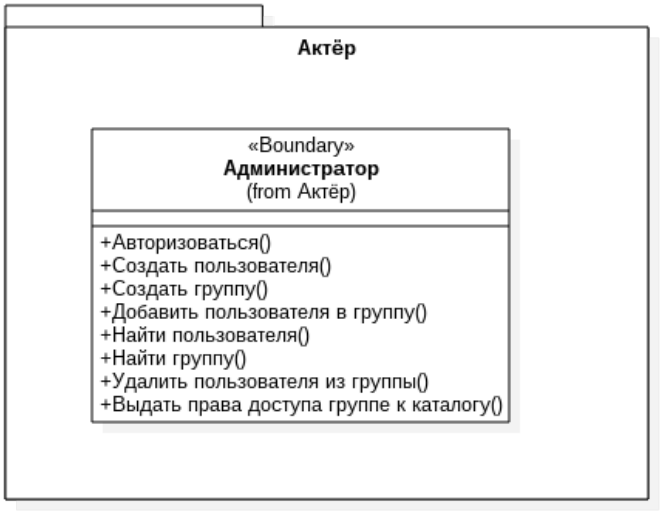

Рисунок 3.17 — Диаграмма пакета анализа «Актёр»

3.6.5 Диаграмма пакетов анализа

Описанные ранее пакеты анализа объединены на диаграмме рис. 3.18. Диаграмма демонстрирует классы, входящие в пакеты; зависимости между классами и, следовательно, пакетами.

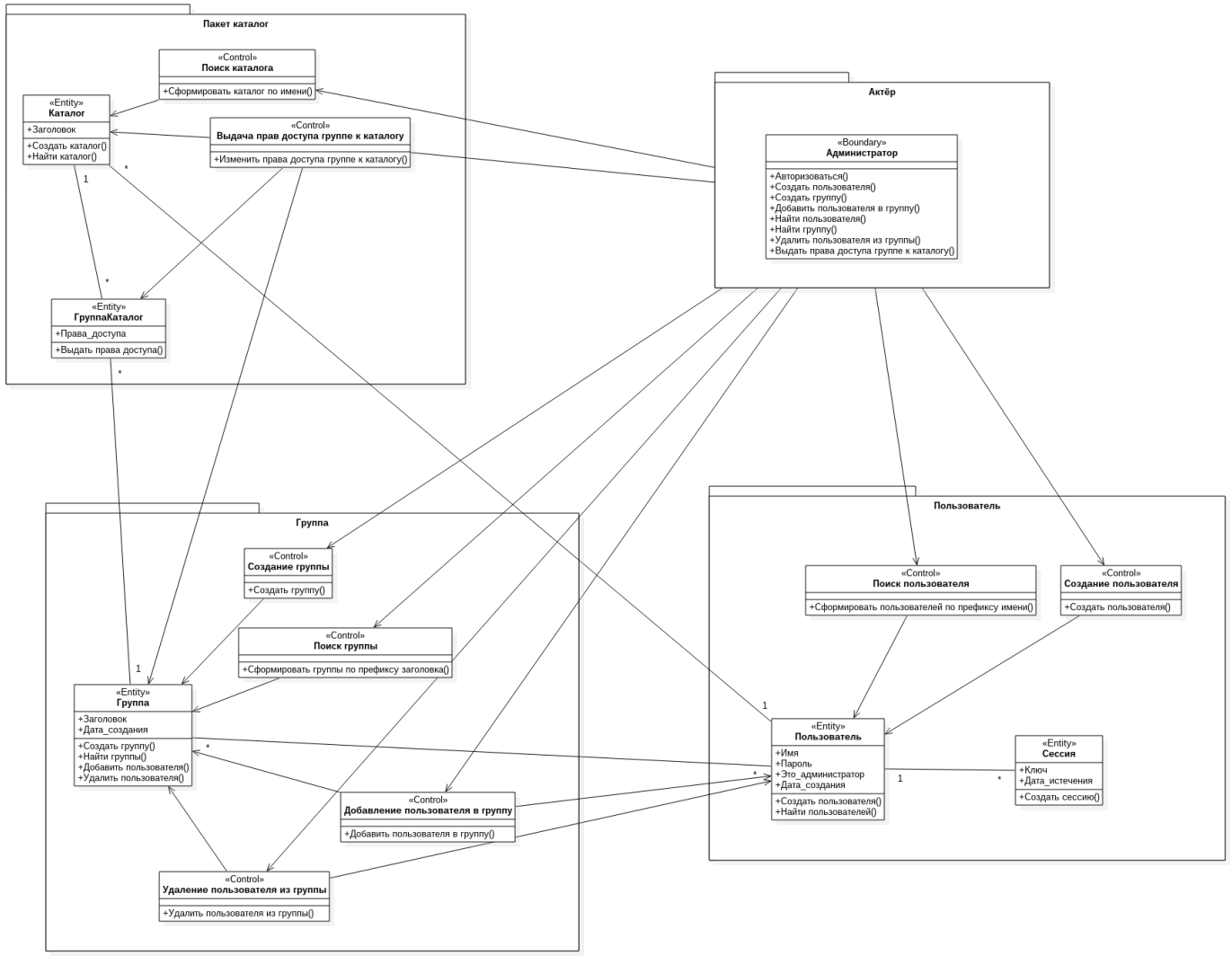

Рисунок 3.18 — Диаграмма пакетов анализа

3.7 Трассировка пакетов анализа в подсистемы

Пакеты анализа трассируются в подсистемы. Соответствие пакетов анализа и подсистем показано на рис. 3.19.

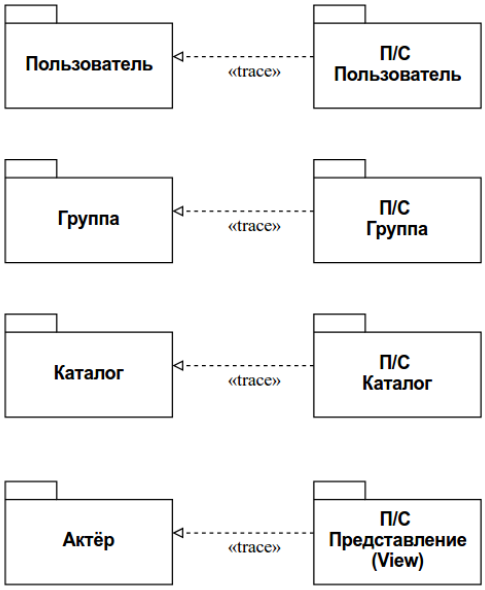

Рисунок 3.19 — Трассировка пакетов анализа в подсистемы

3.8 Уровни подсистем проектирования

На рис. 3.20 показаны три уровня подсистем проектирования. На прикладном уровне располагаются разработанные подсистемы. Ввиду того, что подсистемы не имеют совместно используемых классов/пакетов, то разделение прикладного уровня на общий и специальный не производится. К среднем уровню проектирования относятся СУБД, интерпретатор языка Python 3 и каркас (англ. f*ramework*) Django. На системном уровне представлены протокол TCP/IP для передачи данных по сети. Подсистемы верхних уровней зависят от подсистем нижних уровней.

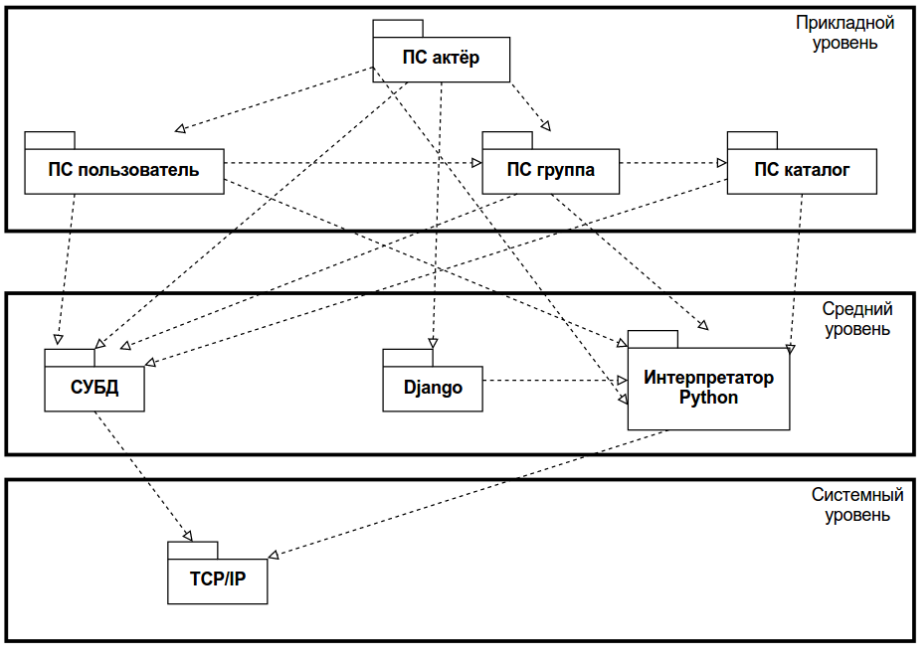

Рисунок 3.20 — Уровни подсистем проектирования для системы файлового сервера

3.9 Диаграмма развёртывания

На рис. 3.21 показана диаграмма развёртывания. Все разрабатываемые артефакты разворачиваются на сервере приложений. Клиент обращается на сервер с помощью браузера.

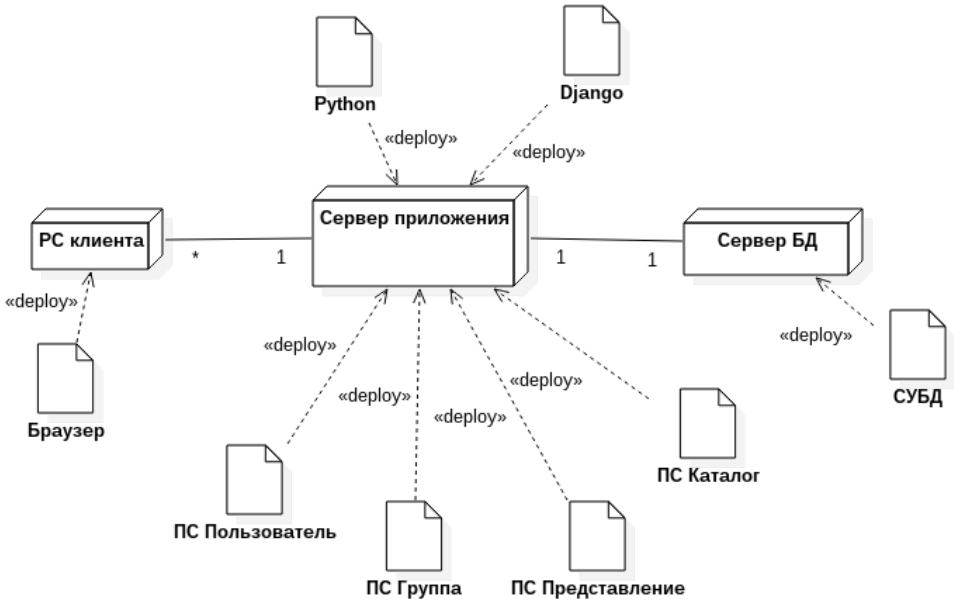

Рисунок 3.21 — Диаграмма развёртывания

3.10 Преобразование классов анализа

3.10.1 Преобразование граничного класса «Администратор»

Трассировка граничного класса «Администратор» в класс View изображена на рис. 3.22. Класс View существует на логическом уровне ввиду использования фреймворка Django. На этапе разработки класс View представляет собой файл views.py с методами класса View.

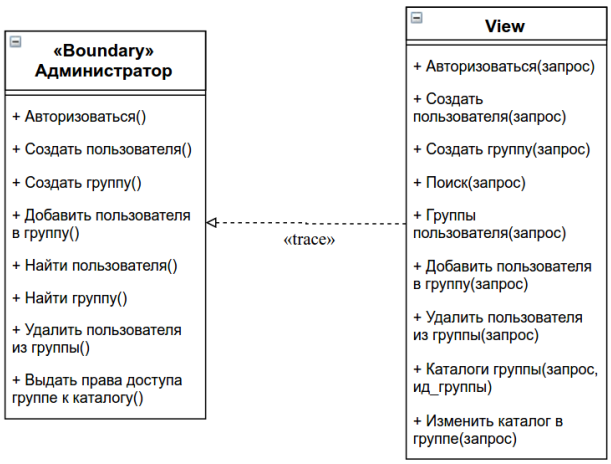

Рисунок 3.22 — Трассировка граничного класса «Администратор»

Класс View использует формы, которые отображаются пользователю, а также валидируют входные данных. Их диаграмма классов изображена на рис. 3.23.

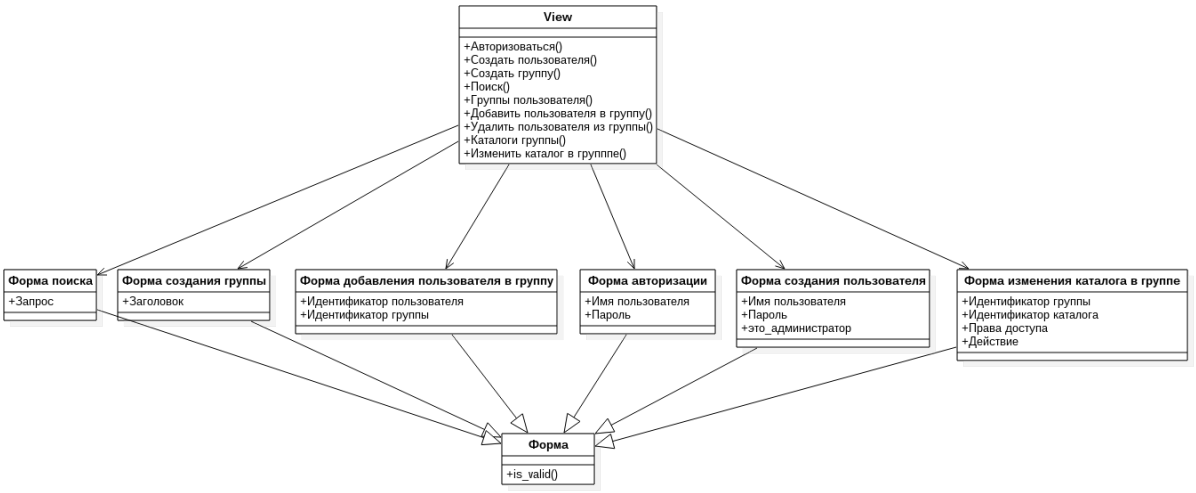

Рисунок 3.23 — Диаграмма классов форм и View

3.10.2 Преобразование управляющих классов с учётом паттерном бизнес-логики

Бизнес-логика может быть сложной, с множеством правил и условий, оговаривающих различные варианты использования и особенностей поведения системы. Для облегчения таких трудностей предназначены объекты. Паттерн «Модель предметной области» предусматривает создание сети взаимосвязанных объектов, каждый из которых представляет некую осмысленную сущность. В ходе анализа предметной области были выделены следующие управляющие классы с учётом паттерна «Модель предметной области»: Group, User, Catalogue. Трассировки управляющих классов анализа в управляющие классы с учётом бизнес-логики представлены на рис. 3.24 — 3.26.

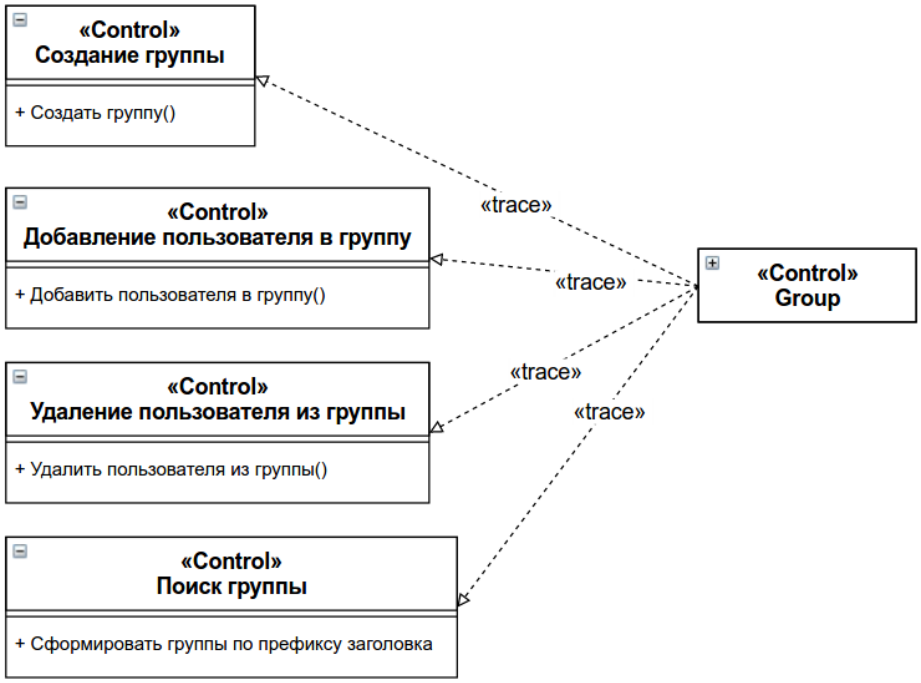

Рисунок 3.24 — Трассировка управляющих классов по работе с группой в класс Group

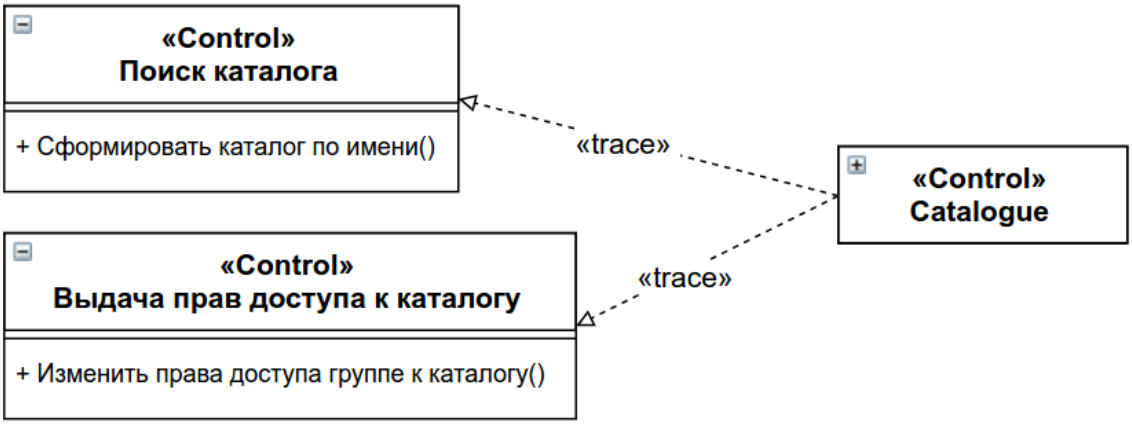

Рисунок 3.25 — Трассировка управляющих классов по работе с каталогом в класс Catalogue

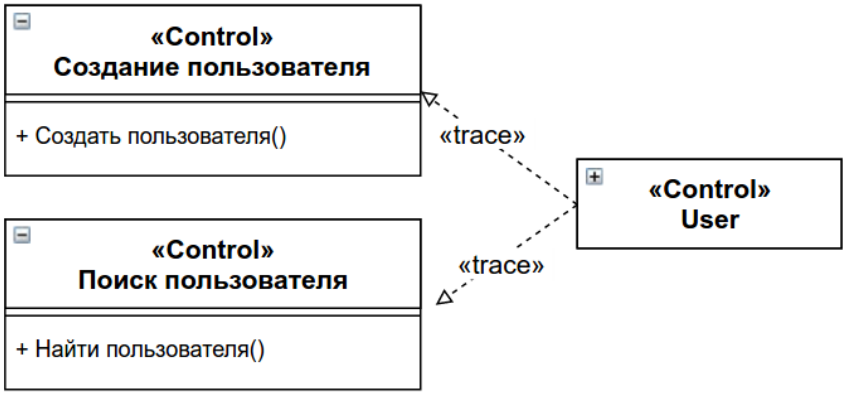

Рисунок 3.26 — Трассировка управляющих классов по работе с пользователем в класс User

3.10.3 Преобразование классов сущностей с учётом паттернов доступа к БД

В качестве паттерна доступа к БД используется активная запись. Каждая активная запись отвечает за сохранение и загрузку информации в базу данных, а также за логику домена, применяемую к данным. Структура активной записи в точности соответствует записи в таблице БД. Каждый класс сущностей этапа анализа трассируется в активную запись с сохранением полей и добавлением методов создания, сохранения и удаления. Поскольку активная запись содержит логику домена, то классы обладают статическими метода поиска экземпляров в БД. Классы активных записей представлены далее на диаграммах классов. Они имеют в имени постфикс в виде ActiveRecord.

3.11 Диаграмма взаимодействия

Диаграмма взаимодействия для прецедента «Добавить пользователя в группу» изображена на рис. 3.27.

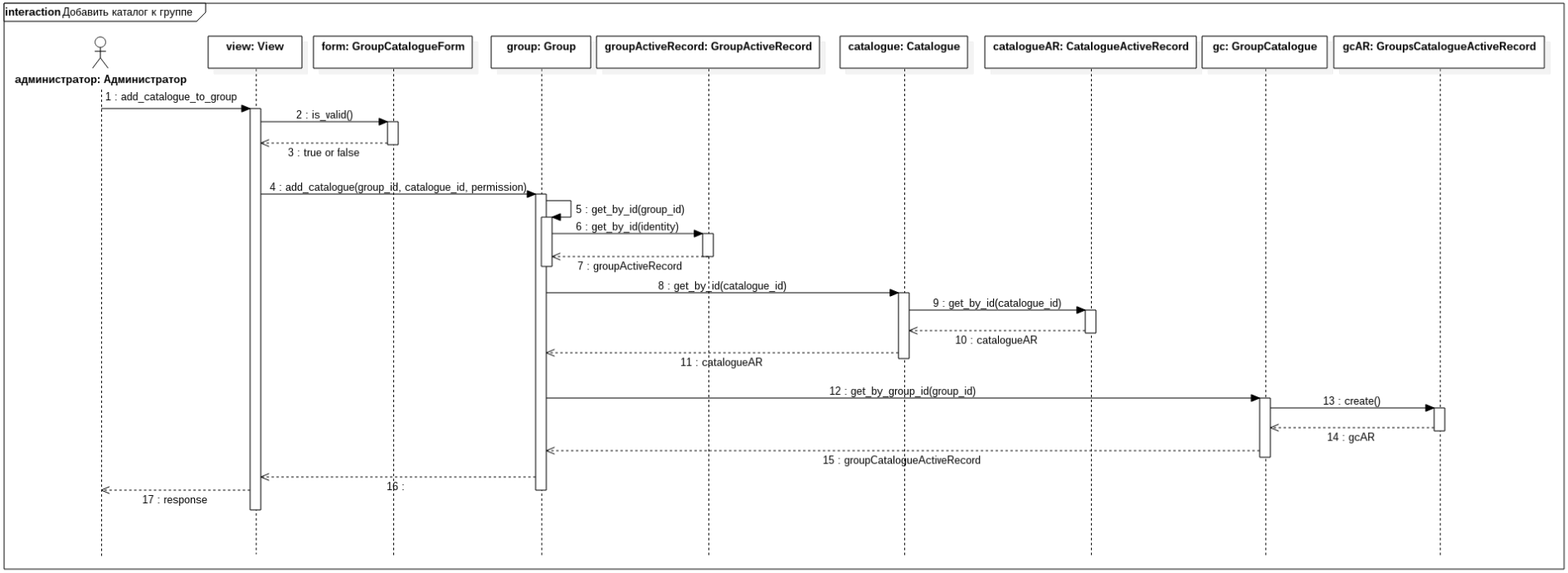

Рисунок 3.27 — Диаграмма взаимодействия для прецедента «Добавить каталог к группе»

3.12 Диаграмма взаимодействия для главного прецедента в терминах подсистем

Диаграмма взаимодействия для прецедента «Выдать права доступа группе к каталогу» в терминах взаимодействия подсистем изображена на рис. 3.28.

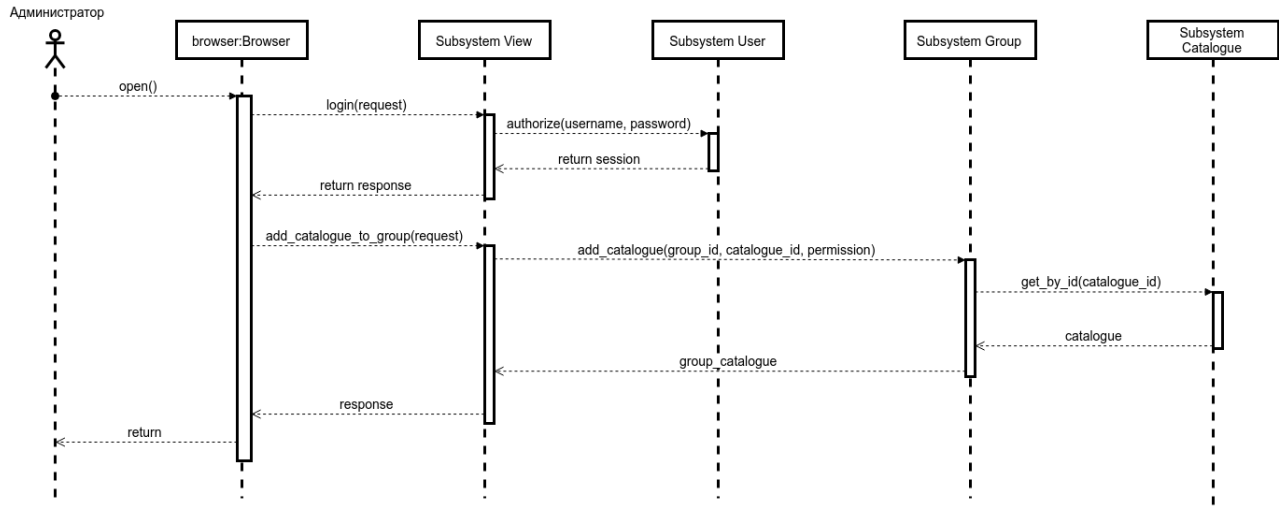

Рисунок 3.28 — Диаграмма взаимодействия в терминах подсистем

3.13 Диаграмма классов

3.13.1 Диаграмма граничных классов

Граничный класс View и используемые им формы для валидации входных данных изображены на рис. 3.29.

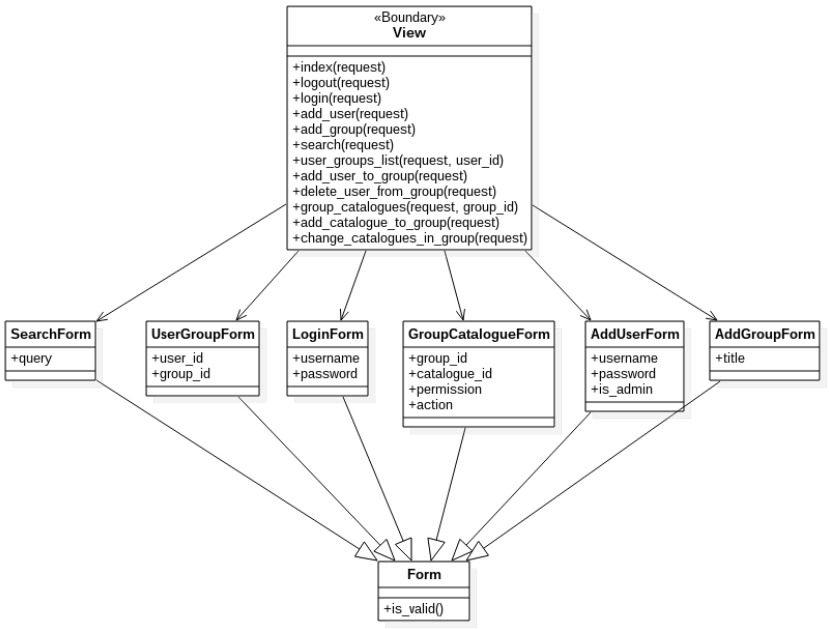

Рисунок 3.29 — Диаграмма граничных классов

3.13.2 Диаграмма управляющих классов

Управляющие классы реализованы с учётом паттерна «Модель предметной области». Их диаграмма на рис. 3.30.

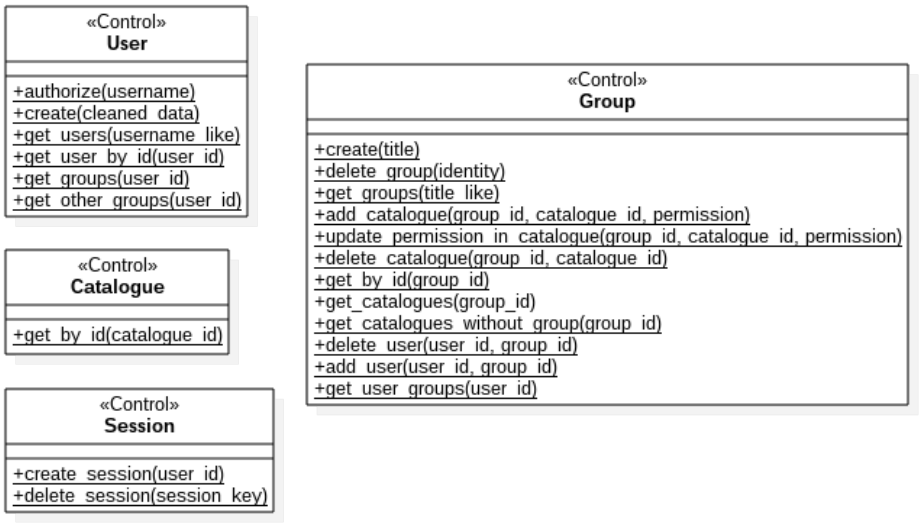

Рисунок 3.30 — Диаграмма управляющих классов

3.13.3 Диаграмма классов сущностей

Классы сущностей с учётом паттерна «Активная запись» изображены на рис. 3.31.

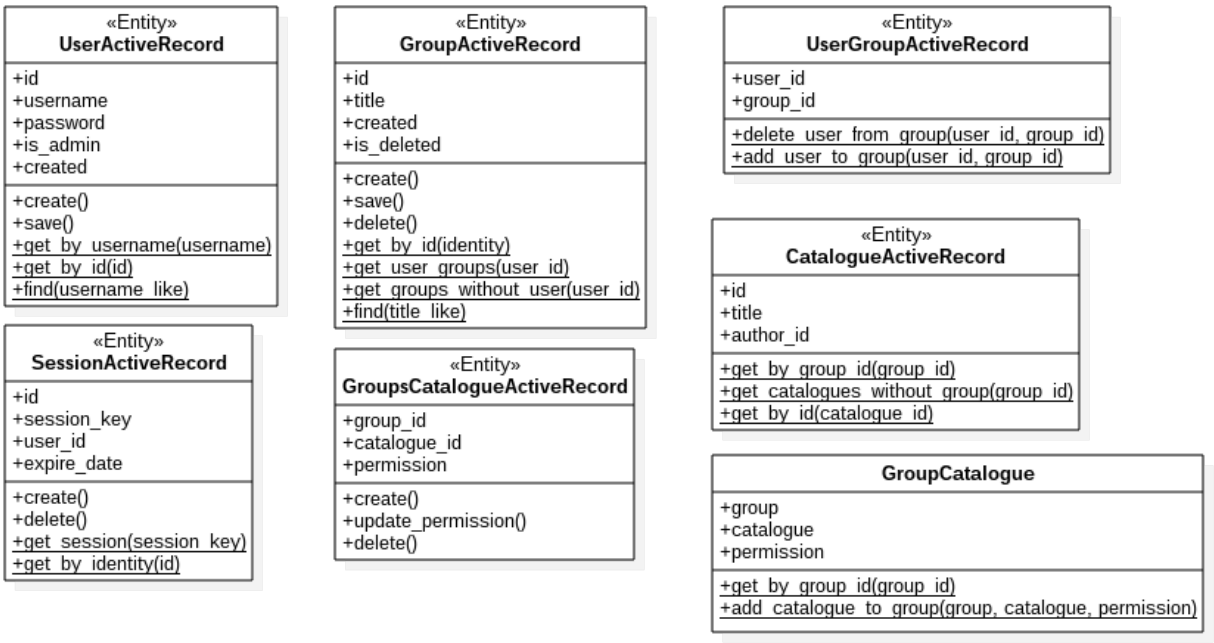

Рисунок 3.31 — Диаграмма классов сущностей

3.14 Диаграмма пакетов

Пакеты и зависимости между ними показаны на рис. 3.32. Диаграмма пакетов и классов, входящих в них — рис. 3.33. На диаграммах пакетов были достигнуты следующие цели:

— Сильная связность классов внутри пакета;

— Слабое сцепление между пакетами;

— Возможность параллельной разработки отдельных пакетов за счёт слабого сцепления с другими пакетами.

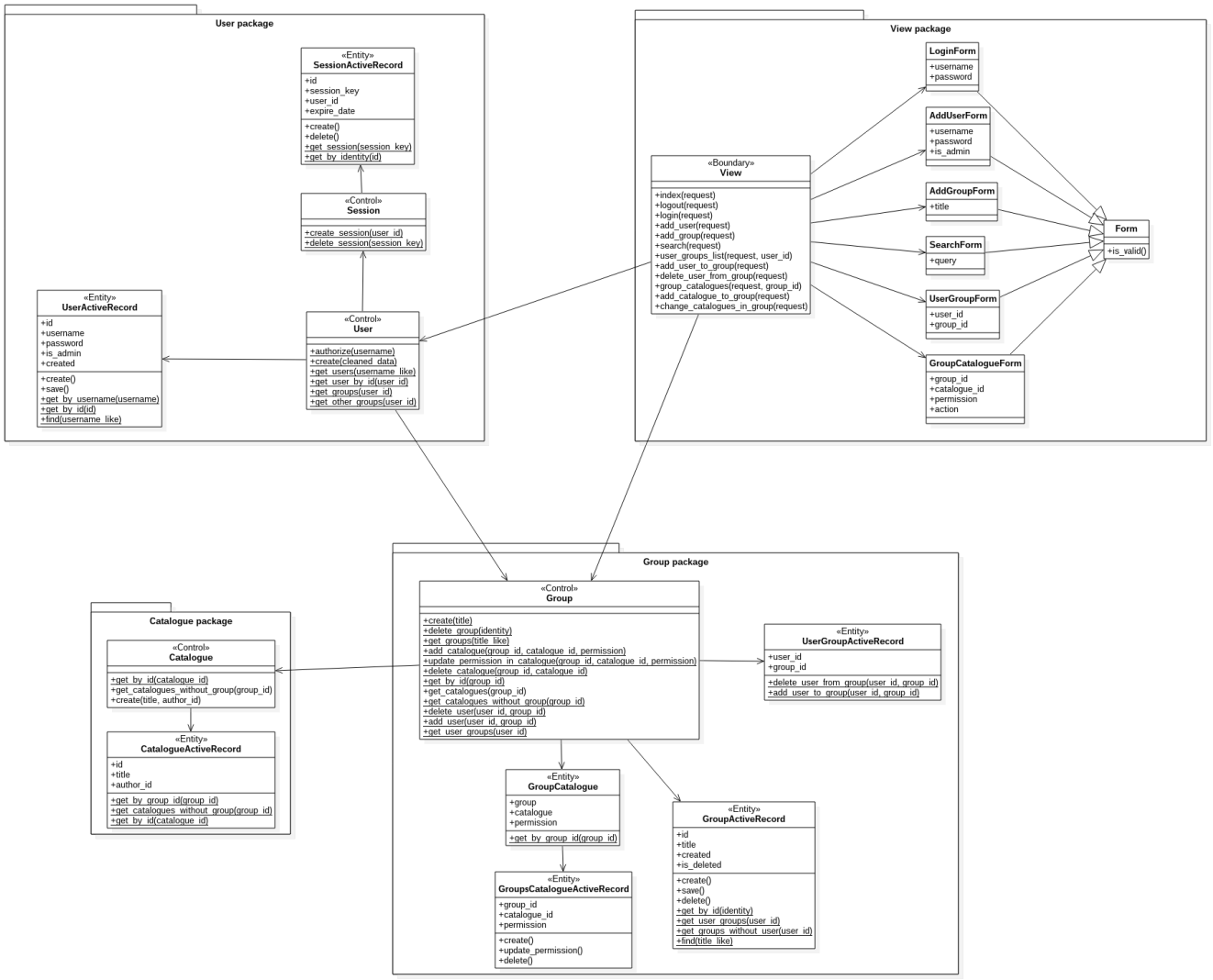

Рисунок 3.32 — Диаграмма пакетов с классами, входящими в них

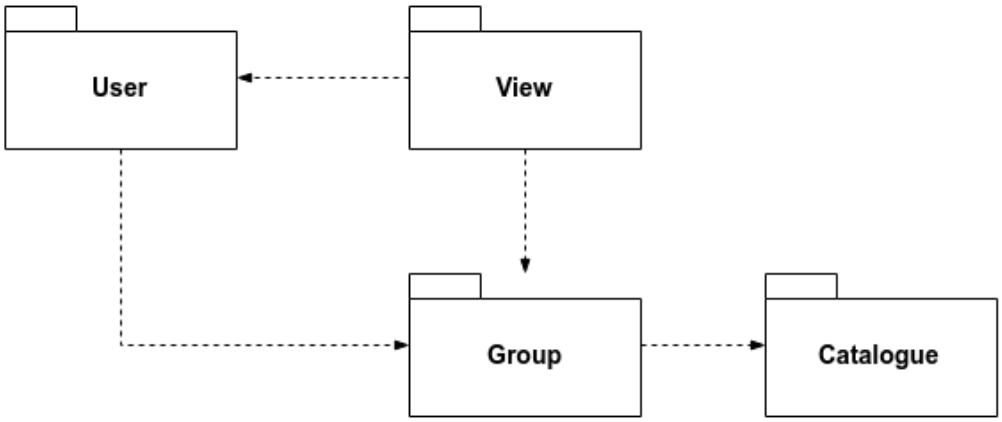

Рисунок 3.33 — Диаграмма пакетов

3.15 Диаграмма схемы базы данных

Схема реляционной базы данных состоит из 7 таблиц, которые представлены на рис. 3.34.

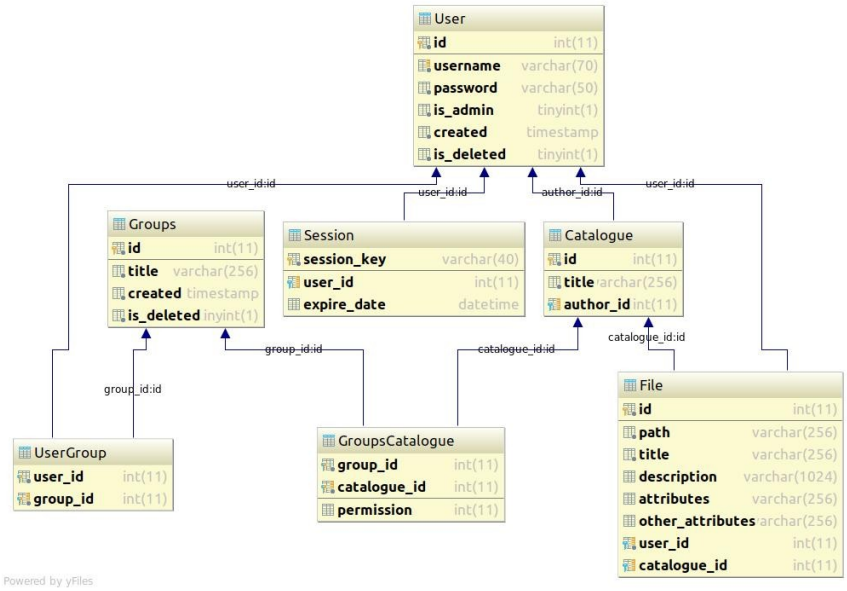

Рисунок 3.34 — Схема базы данных

3.16 Диаграмма компонентов

Каждая подсистема представляет собой компонент с интерфейсом. Так, наапример, подсистема User представляет собой компонент User с интерфейсом IUser. Управляющий классы User реализует методы интерфейса IUser. Методы управляющих классов совпадают с интерфейсами подсистемы, в которой они расположены. Диаграмма компонентов на рис. 3.35.

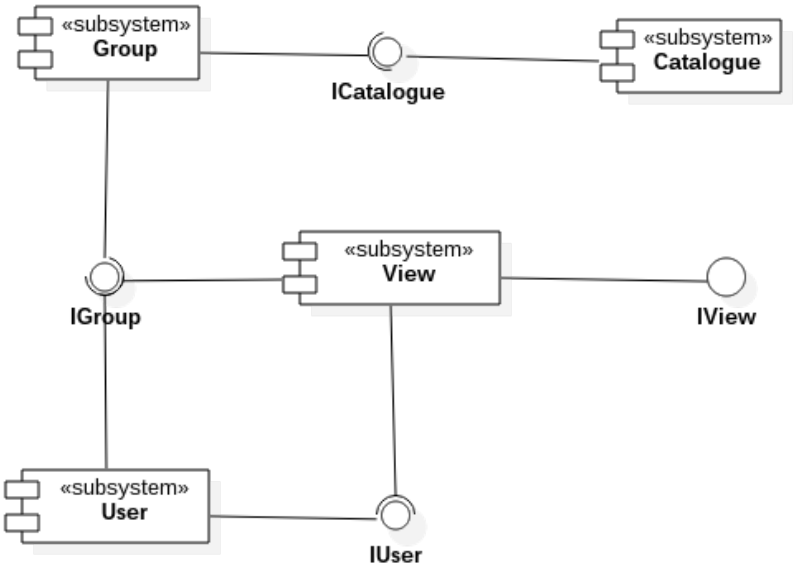

Рисунок 3.35 — Диаграмма компонентов

3.17 Диаграмма развёртывания

Диаграмма развёртывания (рис. 3.36) моделирует физическое развёртывание артефактов и компонентов на узлах. Все разработанные компоненты располагаются на сервере приложения.я

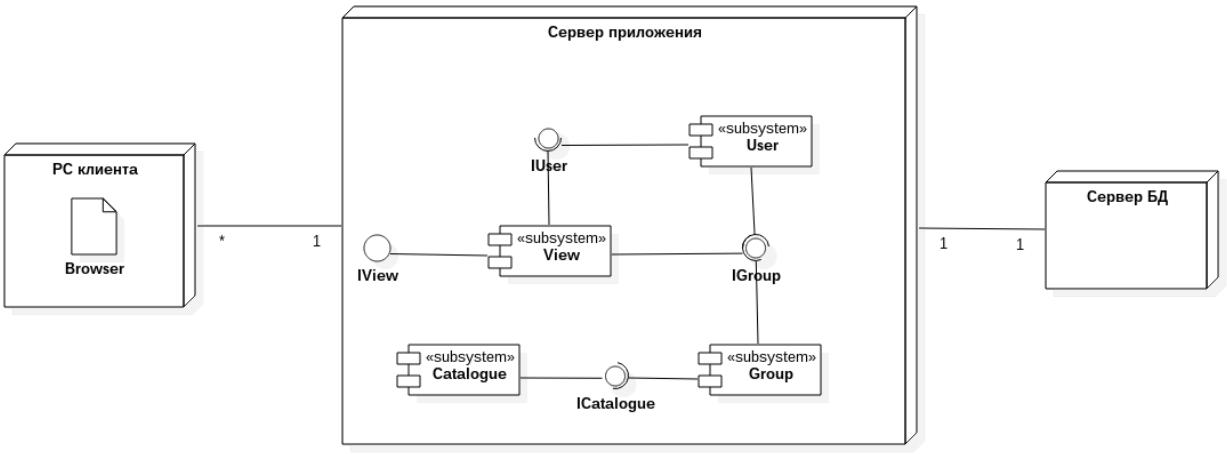

Рисунок 3.36 — Диаграмма развёртывания

3.18 Исходный код программы, реализующий архитектурно-значимые прецеденты

Ниже представлен исходный код программы, реализующий прецедент «Добавить каталог к группе с правами доступа».

 # views.py

```
@login_required
@admin_required
def add_catalogue_to_group(request):
    """Добавить каталог к группе"""
    form = GroupCatalogueForm(request.POST)
    if form.is_valid():
        try:
            Group.add_catalogue(form.cleaned_data['group_id'],
                                form.cleaned_data['catalogue_id'],
                                form.cleaned_data['permission'])
        except Exception as e:
            messages.warning(request, e)
    return redirect(request.META.get('HTTP_REFERER', '/'))
```

```
# groups.py

class Group:
    @staticmethod
    def add_catalogue(group_id, catalogue_id, permission):
        group_id = int(group_id)
        catalogue_id = int(catalogue_id)
        permission = int(permission)
        if any([group_id <= 0, catalogue_id <= 0, permission <= 0, permission > 3]):
            raise ValueError()
        group = Group.get_by_id(group_id)
        if not group:
            print('Group not found')
            return None
        catalogue = Catalogue.get_by_id(catalogue_id)
        if not catalogue:
            print('Catalogue not found')
            return None
        return GroupCatalogue.add_catalogue_to_group(group, catalogue, permission)


      @staticmethod
      def get_by_id(group_id):
      if int(group_id) <= 0:
            raise ValueError
      return GroupActiveRecord.get_by_id(group_id)

class GroupCatalogue:

    def __init__(self):
        self.group = None
        self.catalogue = None
        self.permission = None
    @staticmethod
    def get_by_group_id(group_id):
        sql = 'SELECT * FROM Groups INNER JOIN GroupsCatalogue ON Groups.id =
GroupsCatalogue.group_id ' \
              'INNER JOIN Catalogue ON GroupsCatalogue.catalogue_id = Catalogue.id WHERE
group_id = %s'
        with DbService.get_connection() as cursor:
```

```
            cursor.execute(sql, (group_id,))
            rows = cursor.fetchall()
        if rows:
            return [GroupCatalogue.__deserialize__(row) for row in rows]
        return []
    @staticmethod
    def add_catalogue_to_group(group, catalogue, permission):
        group_catalogue = GroupsCatalogueActiveRecord()
        group_catalogue.group_id = group.id
        group_catalogue.catalogue_id = catalogue.id
        group_catalogue.permission = permission
        group_catalogue.create()
        return group_catalogue

class GroupActiveRecord:
    def __init__(self):
        self.id = None
        self.title = None
        self.created = None
        self.is_deleted = False

    def create(self):
        sql = 'INSERT INTO Groups(title) VALUES (%s)'
        with DbService.get_connection() as cursor:
            cursor.execute(sql, (self.title,))
            self.id = cursor.lastrowid
        if self.id is not None:
            return GroupActiveRecord.get_by_id(self.id)

    def save(self):
        sql = 'UPDATE Groups SET title = %s, self.is_delete = %s WHERE id = %s'
        with DbService.get_connection() as cursor:
            cursor.execute(sql, (self.title, self.is_deleted, self.id))

    @staticmethod
    def get_by_id(identity):
        sql = 'SELECT * FROM Groups WHERE id = %s'
        with DbService.get_connection() as cursor:
            cursor.execute(sql, (int(identity),))
            return GroupActiveRecord.deserialize(cursor.fetchone())

@staticmethod
def deserialize(row):
    if row is None:
        return None
    group = GroupActiveRecord()
    group.id = int(row['id'])
    group.title = str(row['title'])
    group.created = row['created']
    group.is_deleted = bool(row['is_deleted'])
    return group

class GroupsCatalogueActiveRecord:
    def __init__(self):
        self.group_id = None
        self.catalogue_id = None
        self.permission = None
    def create(self):
        sql = 'INSERT INTO GroupsCatalogue(group_id, catalogue_id, permission) VALUES
(%s, %s, %s)'
        with DbService.get_connection() as cursor:
            cursor.execute(sql, (self.group_id, self.catalogue_id, self.permission))
    def update_permission(self):
```

```
        sql = 'UPDATE GroupsCatalogue SET permission = %s WHERE group_id = %s AND
catalogue_id = %s'
        with DbService.get_connection() as cursor:
            cursor.execute(sql, (self.permission, self.group_id, self.catalogue_id))
    def delete(self):
        sql = 'DELETE FROM GroupsCatalogue WHERE group_id = %s AND catalogue_id = %s'
        with DbService.get_connection() as cursor:
     cursor.execute(sql, (self.group_id, self.catalogue_id))
```

3.19 Переработанный список рисков

В ходе этапа проектирования перечень рисков был переработан, в частности минимизирован следующие риски:

- Риск недостаточной оценки сложности — команда разработки разобралась с предметной областью и новыми технологиями. Таким образом, задача проектирования и реализации является посильной для нанятых разработчиков;
- Риск, связанный с командой — минимизирован за счёт введения практик комментирования кода и его просмотра (англ. *code review*);

В ходе проектирования новые риски не были обнаружены. Таким образом, переработанный список рисков выглядит следующим образом:

- Проектный риск — риск изменения требований к программному продукту;
- Технический риск — риск неверной реализации;
- Коммерческие риски — риск создания ненужного продукта и риск потери финансирования.

В виду того, что в ТЗ вносятся изменения (добавление нового функционала), то проектный риск изменения требований был верно подмечен. Также сохраняется коммерческий риск создания ненужного продукта.

3.20 Расчёт функциональных указателей на основе прототипа пользовательского интерфейса и коэффициентов сложности

Рассмотрим каждый из экранов и определим их ранг и сложности. Расчёты представлены в таблице 3.1.

Таблица 3.1. Ранг и оценка сложности экранов

Экран	**Тип**	**Кол-во**	**Ссылки на файлы / типы элементов-записей**	**Элементы данных**	**Ранг и оценка сложности**
Авторизация	Внешний ввод (логин)	1	1	3	Низкий(3)
Главная страница	Запрос (ссылки в заголовке)	5	0	1	Низкий(3)
Добавить пользователя	Внешний ввод (форма ввода)	1	1	4	Низкий(3)
	Внешний вывод (ошибки)	1	0	1	Низкий(3)
Добавить группу	Внешний ввод (форма ввода)	1	1	2	Низкий(3)
Поиск	Запрос (поисковая форма)	1	2	2	Низкий(3)
	Внешний интерфейсный файл (результаты)	2	2	1	Низкий(5)
Пользователь	Внешний вывод (информация о пользователе)	1	1	5	Низкий(4)
	Внешний вывод (информация о группах пользователя)	1	2	2	Низкий(4)
	Внешний вывод (информация об остальных группах)	1	1	2	Низкий(4)
	Внешний ввод (добавить и удалить)	2	2	1	Низкий(3)
Группа	Внешний вывод (информация о группе)	1	1	4	Низкий(4)

	Внешний вывод (информация о доступных каталогах)	1	2	3	Низкий(4)
	Внешний вывод (информация об остальных каталогах)	1	1	3	Низкий(4)
	Внешний ввод (добавить каталог к группе)	1	2	2	Низкий(3)
	Внешний ввод (редактирование прав доступа)	1	2	3	Низкий(3)

Просуммировав значения сложностей с учётом количества элементов получим общее количество = 13*3 + 6*4 + 2*5 = 73

$$FP = Общее\ количество \times (0{,}65 + 0{,}01 \times \sum_{i=1}^{14} F_i) \quad (3.1),$$

где Fi — коэффициет регулировки сложности (см. таблицу 3.2). Каждый коэффициент может принимать следующие значения: 0 — нет влияния, 1 — случайное, 2 — небольшое, 3 — среднее, 4 — важное, 5 — основное. Значения выбираются эмпирически в результата ответов на вопросы [3, стр. 27, табл. 2.11].

Таблица 3.2. Значения системных параметров приложения

№	Системный параметр	Значение коэффициента
1	Передачи данных	2
2	Распределенная обработка данных	0
3	Производительность	4
4	Распространенность используемой конфигурации	3
5	Скорость транзакций	4
6	Оперативный ввод данных	2
7	Эффективность работы конечного пользователя	5
8	Оперативное обновление	1
9	Сложность обработки	2

10	Повторная используемость	3
11	Легкость инсталляции	1
12	Легкость эксплуатации	1
13	Разнообразные условия размещения	5
14	Простота изменений	3

Таким образом, $FP = 73 \times (0{,}65 + 0{,}01 \times 36) = 73{,}73$

Модель COCOMO II не предусматривает коэффициента перевода функциональных указателей в LOC-оценки (lines of code) для языка программирования Python. Коэффициент перевод FP в LOC подобран эмпирически и равен 24.

Тогда в разрабатываемой системе предполагается:

$$LOC = 24 \times FP = 24 \times 73{,}73 = 1769 \quad (3.2)$$

3.21 Оценка проекта по COCOMO II этапа постархитектуры

Модель этапа постархитектуры используется в период, когда уже сформирована архитектура и выполняется дальнейшая разработка разработка программного продукта. Затраты можно определить по следующей формуле:

$$ЗАТРАТЫ = A \times K_{\sim req} \times РАЗМЕР^{B} \times M_{p} + ЗАТРАТЫ_{auto} [\text{чел.-мес}] \quad (3.3),$$

где:

- Коэффициент $K_{\sim req}$ учитывает возможные изменения в требованиях;
- A — масштабный коэффициент, равный 2,5;
- Показатель B отражает нелинейную зависимость затрат от размера проекта (размер выражается в KLOC) (рассчитан в разделе 2.8);
- В размере проекта учитываются 2 составляющие: новый код и повторно используемый код. В нашем случае повторно используемый код не используется;
- Множитель поправки Mp зависит от 17 факторов затрат, характеризующих продукт, аппаратуру, персонал и проект.

На основе факторов затрат определим множитель поправки M_p. Разобъём факторы затрат на 4 категории. Для каждого фактора определяется оценка по 6ти балльной шкале. Далее каждая оценка переводится по таблице Боэма в множитель затрат EM_i. Перемножение всех множителей затрат даёт множитель поправки пост-архитектурной модели. Оценка факторов затрат приведена в таблице 3.3.

Таблица 3.3. Оценка факторов затрат

Категория	**Фактор**	**Значение**	**EM_i**
Фактор продукта	Требуемая надёжность ПО (RELY)	3	1,00
	Размер базы данных (DATA)	2	1,00
	Сложность продукта (CPLX)	0	0,75
	Требуемая повторная используемость (RUSE)	1	0,91
	Документирование требований жизненного цикла (DOCU)	2	1,00
Факторы платформы (виртуальной машины)	Ограничения времени выполнения (TIME)	2	1,00
	Ограничения оперативной памяти (STOR)	2	1,00
	Изменчивость платформы (PVOL)	1	0,87
Факторы персонала	Возможности аналитика (ACAP)	2	1,00
	Возможности программиста (PCAP)	4	0,74
	Опыт работы с приложением (AEXP)	2	1,00
	Опыт работы с платформой (PEXP)	1	1,12
	Опыт работы с языком и утилитами (LTEX)	2	1,00
	Непрерывность персонала (PCON)	1	1,10
Факторы проекта	Использование программных утилит (TOOL)	3	0,86
	Мультисетевая разработка (SITE)	3	0,92
	Требуемый график разработки (SCED)	0	1,29

$$M_p = \prod_{i=1}^{17} EM_i = 0{,}5525 \quad (3.4)$$

Рассчитаем коэффициент $K_{\sim req}$, учитывающий возможные изменения в требованиях.

$$K_{\sim req} = 1 + \frac{BRAK}{100} \quad (3.5),$$

где BRAK — процент кода, отброшенного (модифицированного) из-за изменения требований.

На основе профессионального опыта заложимся на то, что BRAK = 20% ввиду возможных модификаций требований к продукту в процессе разработки. Тогда $K_{\sim req} = 1,2$.

Тогда:

$$ЗАТРАТЫ = 2,5 \times 1,2 \times 1,769^{1,09} \times 0,5525 + 0 = 3,0726 \quad (3.6)$$

$$СТОИМОСТЬ = ЗАТРАТЫ \times РАБ\_КОЭФ = 3,0726 \times 90000 = 276534 [руб.] \quad (3.7)$$

$$Длительность(TDEV) = [3,0 \times ЗАТРАТЫ^{0,33+0,2\cdot(B-1,01)}] \times SCEDPercentage / 100 [мес] \quad (3.8),$$

где SCEDPercentage — процент увеличения (уменьшения) номинального графика.

Примем SCEDPercentage = 100, т. к. требуется определить номинальный график. Тогда длительность разработки $TDEV = 4,3873$ мес. Вычисленные значения приведены в таблице 3.4.

Таблица 3.4. Вычисленные значения параметров по модели COCOMO II этапа постархитектуры

Параметр	**Значение**
Показатель B	1,09
Множитель поправки M_p	0,5525
Коэффициент изменений $K_{\sim req}$	1,2
Затраты	3,0726
Стоимость	276 534 руб.
Длительность	4 мес.

3.22 Перечень и состав итераций

В следующем релизе будет реализован следующий функционал:

— Экспорт отчётов по пользователям (прецеденты — экспорт отчёта по пользователям в pdf; экспорт отчёта по пользователям в csv);

— Экспорт отчётов по группам (прецеденты — экспорт отчёта по группам в pdf; экспорт отчёта по группам в csv).

Описанные прецеденты требуется реализовать в течение одной итерации.

4 Этап построения (конструирования)

4.1 Состав итерации

В данной итерации будет реализован следующий функционал:

— Экспорт отчётов по пользователя в pdf и csv файлы;

— Экспорт отчётов по группам в pdf и csv файлы.

4.2 Модель требований для новых прецедентов

4.2.1 Диаграмма прецедентов

Новые прецеденты для итерации показаны на рис. 4.1. Их спецификация описана в разделе 4.2.2.

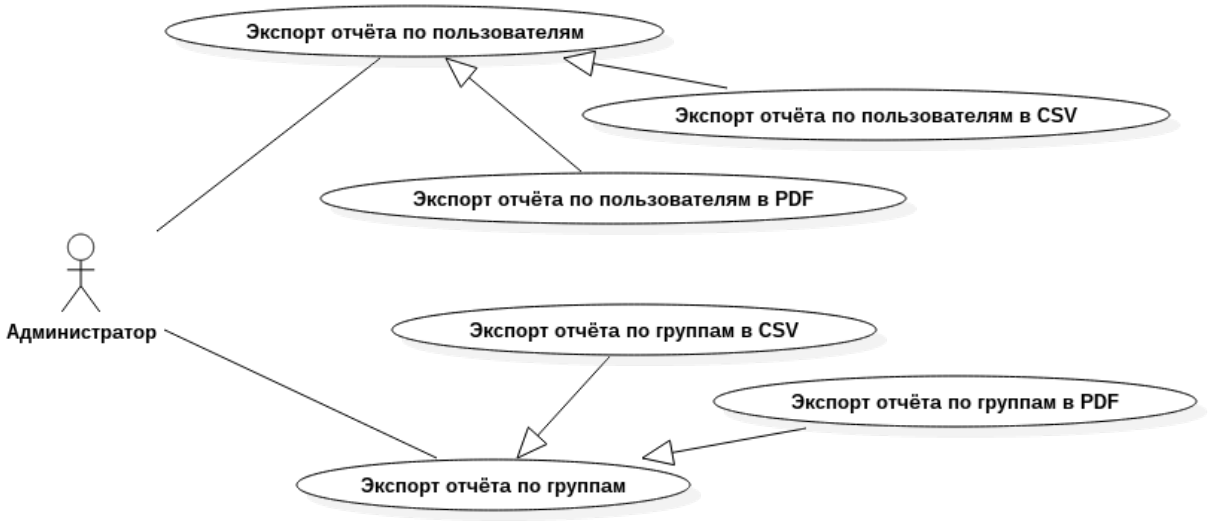

Рисунок 4.1 — Диаграмма прецедентов для новых требований

4.2.2 Спецификация

4.2.2.1 Спецификация для прецедента «Экспорт отчёта по пользователям в PDF»

Предусловие: система должна содержать хотя бы одного администратора; администратор должен быть авторизован.

Основной поток

Зайти на главную системы. Выбрать пункт «Отчёт по пользователя в PDF». Браузер скачает pdf-файл с отчётом по пользователям.

4.2.2.2 Спецификация для прецедента «Экспорт отчёта по пользователям в CSV»

Предусловие: система должна содержать хотя бы одного администратора; администратор должен быть авторизован.

Основной поток

Зайти на главную системы. Выбрать пункт «Отчёт по пользователя в CSV». Браузер скачает csv-файл с отчётом по пользователям.

4.2.2.3 Спецификация для прецедента «Экспорт отчёта по группам в PDF»

Предусловие: система должна содержать хотя бы одного администратора; администратор должен быть авторизован.

Основной поток

Зайти на главную системы. Выбрать пункт «Отчёт по группам в PDF». Браузер скачает pdf-файл с отчётом по группам.

4.2.2.4 Спецификация для прецедента «Экспорт отчёта по группам в CSV»

Предусловие: система должна содержать хотя бы одного администратора; администратор должен быть авторизован.

Основной поток

Зайти на главную системы. Выбрать пункт «Отчёт по группам в CSV». Браузер скачает csv-файл с отчётом по группам.

4.2.3 Прототип пользовательского интерфейса

Прототипы пользовательского интерфейса для новых прецедентов показаны на рис. 4.2.

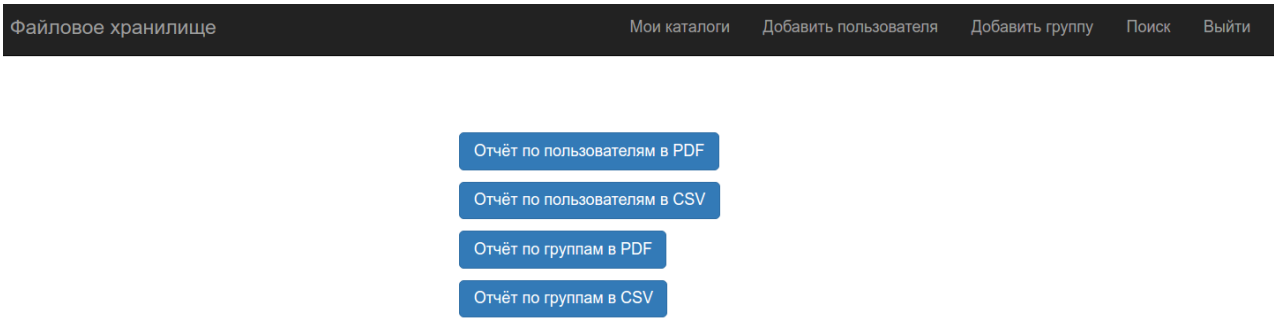

Рисунок 4.2 — Прототип пользовательского интерфейса для новых

прецедентов

4.3 Модель анализа для новых прецедентов

4.3.1 Диаграмма кооперaций

Диаграмма кооперaций разрабатываемой системы для новых прецедентов представлена на рис. 4.3.

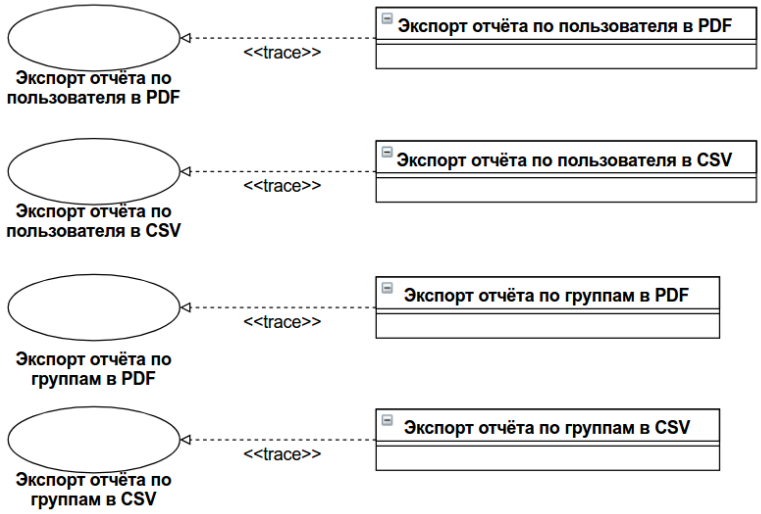

Рисунок 4.3 — Диаграмма кооперaций для новых прецедентов

4.3.1 Диаграмма классов анализа

4.3.1.1 Граничные классы

К граничному классу «Администратор» добавляются два новых метода: отчёт по пользователям; отчёт по группам.

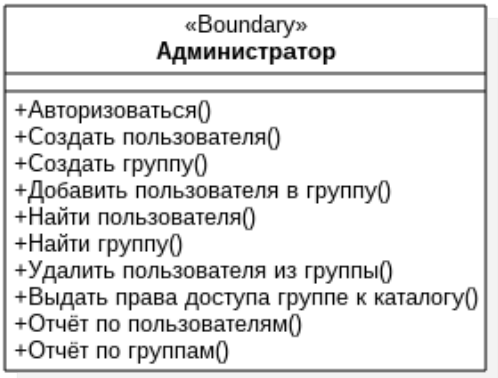

Рисунок 4.4 — Диаграмма граничных классов для новых прецедентов

4.3.1.2 Управляющие классы

Управляющие классы создаются по прецедентам. Базовый класс — «Отчёт». Это класс с абстрактным методом «создать отчёт». Его два наследника «Отчёт по пользователя» и «Отчёт по группам» — также абстрактные, т. к. не переопределяют метод. Они реализуют общую логику получения информации о пользователях и группах. Листовые классы определяют реализацию метода «создать отчёт».

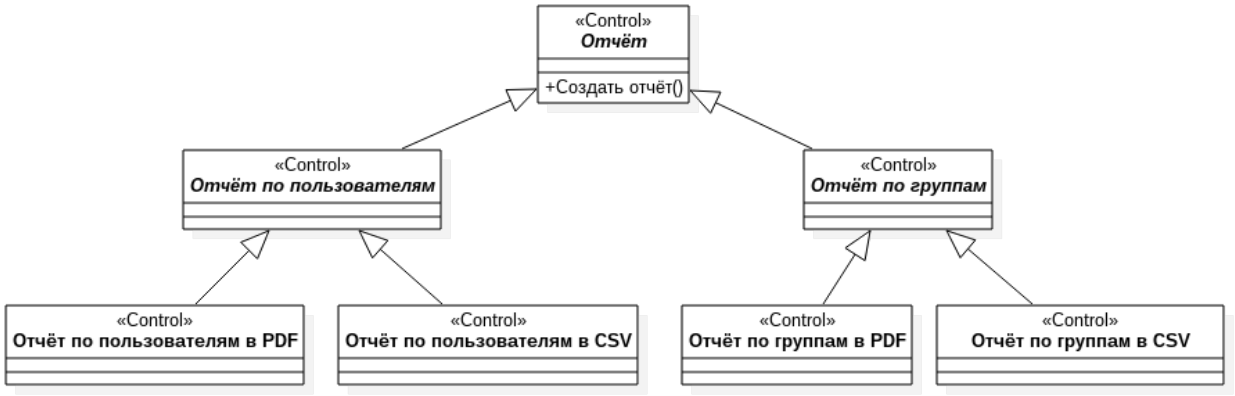

Рисунок 4.5 — Диаграмма управляющих классов

4.3.2 Диаграмма последовательностей для классов анализа

Диаграмма последовательностей для прецедента «Экспорт отчёта по пользователям в PDF» показана на рис. 4.6, для прецедента «Экспорт отчёта по группам в CSV» на рис. 4.7. Диаграммы последовательностей для остальных прецедентов, реализуемых в рамках итерации, аналогичны.

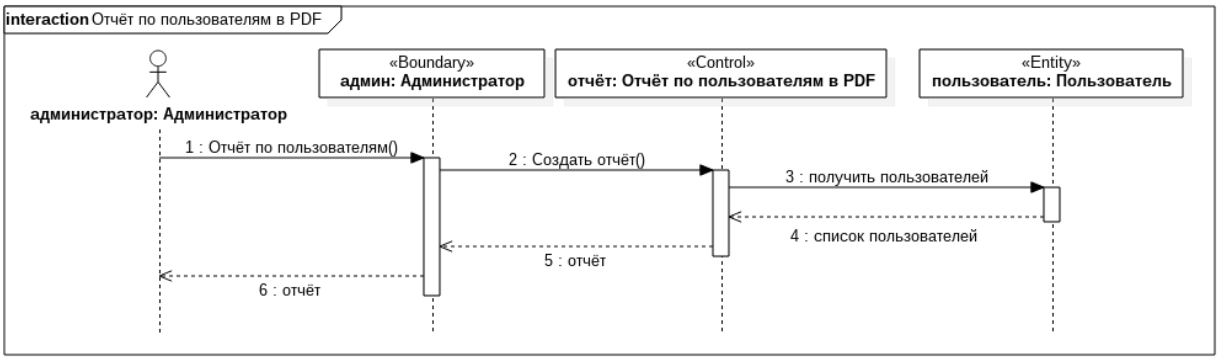

Рисунок 4.6 — Диаграмма последовательностей для прецедента «Экспорт отчёта по пользователям в PDF»

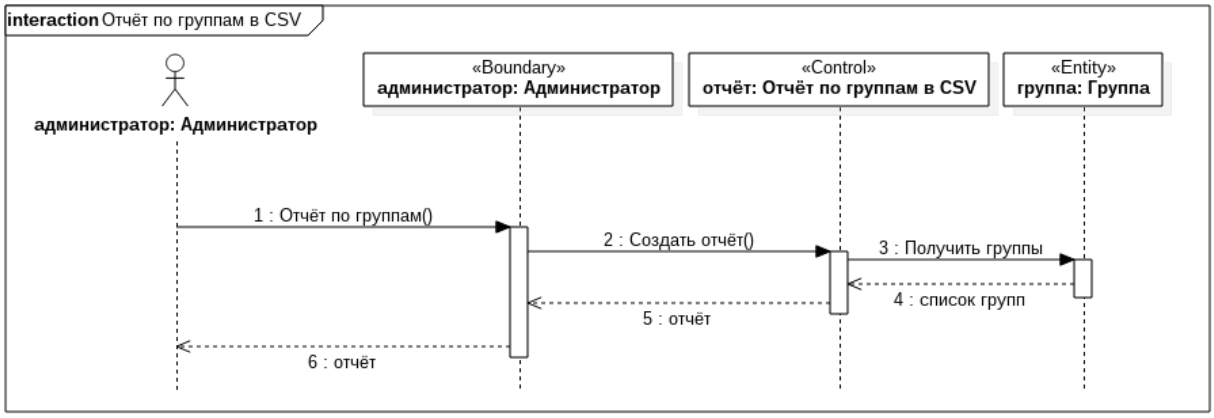

Рисунок 4.7 — Диаграмма последовательностей для прецедента «Экспорт отчёта по группам в CSV»

4.3.3 Диаграмма пакетов

В рамках итерации спроектирован новый пакет — Отчёт. Диаграмма этого пакет показана на рис. 4.8. В пакетах «Группа», «Пользователь» и «Актёр» добавлены новые методы. Диаграмма новых и изменённых пакетов на рис. 4.9.

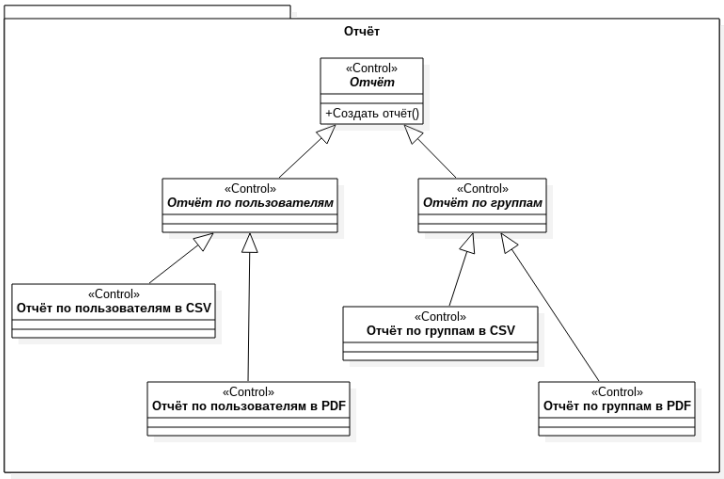

Рисунок 4.8 — Диаграмма пакета «Отчёт»

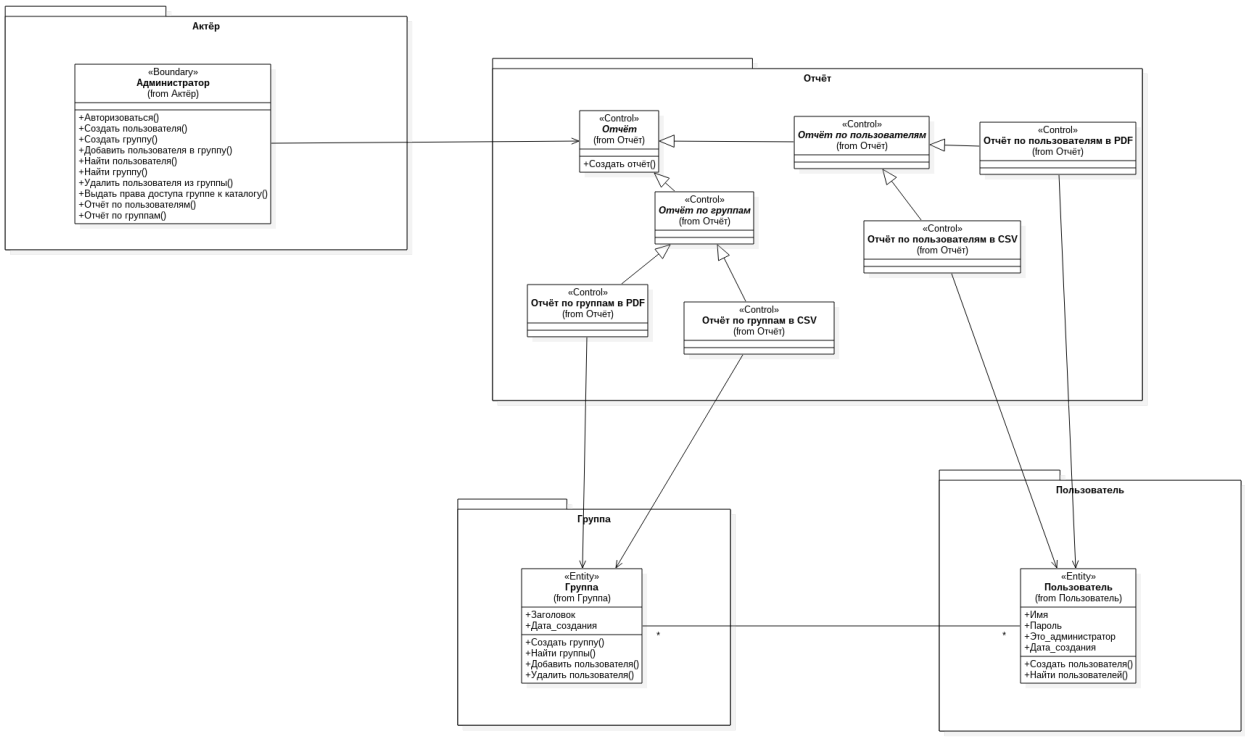

Рисунок 4.9 — Диаграмма пакетов

4.4 Модель проектирования для новых прецедентов

4.4.1 Диаграммы подсистем

Пакет анализа «Отчёт» трассируется в подсистему «П/С Отчёт» на рис. 4.10. Подсистема «П/С Отчёт» представляет собой компонент «Отчёт» (рис. 4.11) с интерфейсом IОтчёт.

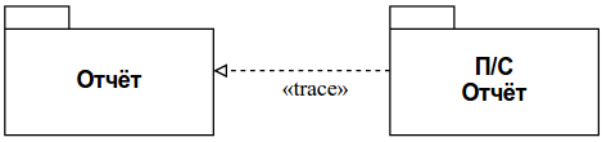

Рисунок 4.10 — Трассировка пакета «отчёт» в подсистему

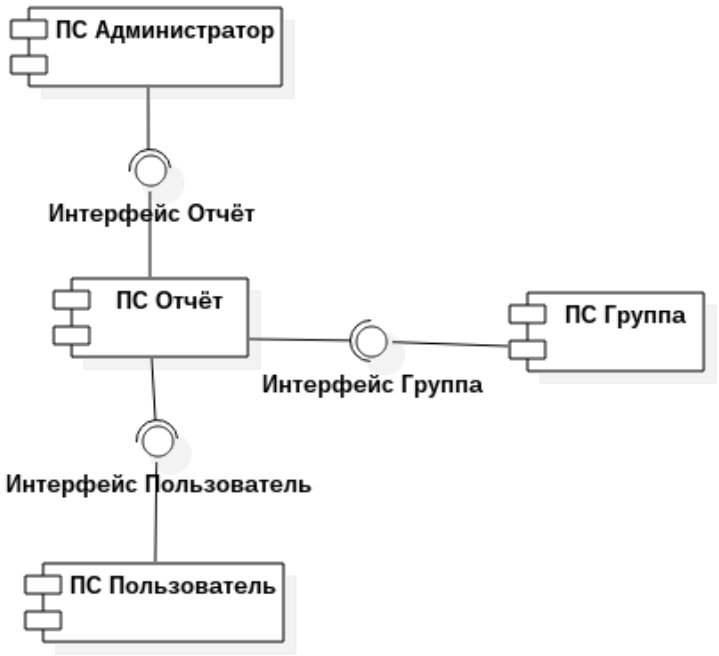

Рисунок 4.11 — Диаграмма подсистем

4.4.2 Диаграмма классов

Диаграмма используемых классов для реализации новых прецедентов изображена на рис. 4.12. Классы пакета Report — на рис. 4.13. Трассировка классов анализа производилась с полным соответствием имён классов и методов, реализуемых ими.

Классы пакета Report спроектированы с учётом паттерна «Приспособленец». Это достигается за счёт класса ReportFactory, который в зависимости от передаваемого типа отчёта создаёт новый экземпляр требуемого класса, либо возвращает уже существующий из внутреннего хранилища, если он был создан ранее. Данный паттерн позволяет уменьшить затраты на создание объектов.

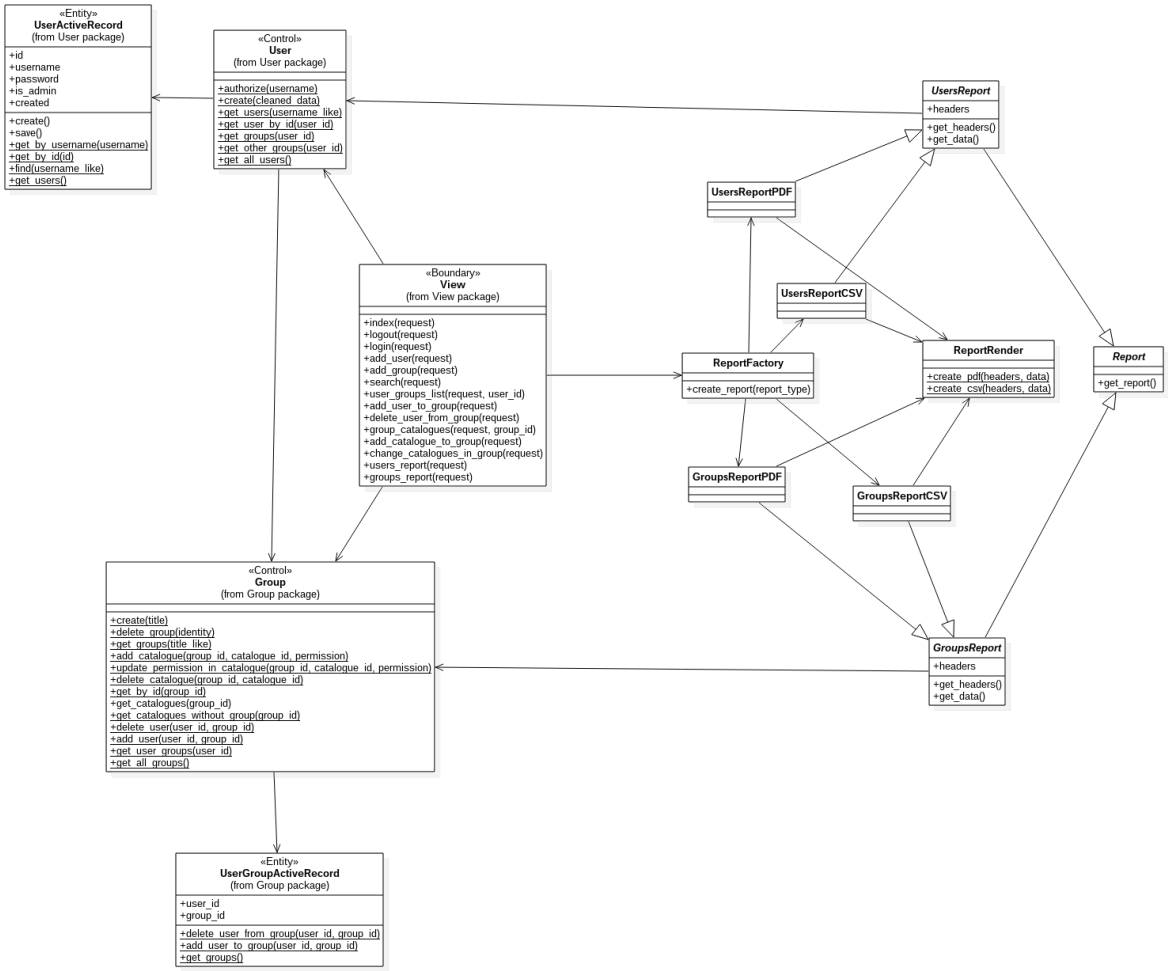

Рисунок 4.12 — Диаграмма используемых классов для новых прецедентов

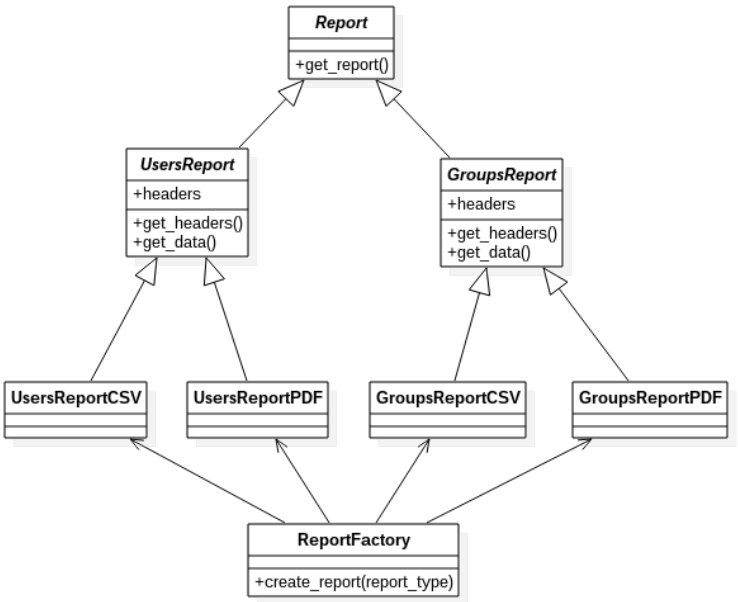

Рисунок 4.13 — Диаграмма классов для паттерна «приспособленец»

4.4.3 Диаграмма пакетов

В новый пакет Report входят следующие классы: Report, UsersReport, UsersReportCSV, UsersReportPDF, GroupsReport, GroupsReportCSV, GroupsReportPDF. Описание их назначение дано в разделе 4.5, диаграмма пакетов на рис. 4.14

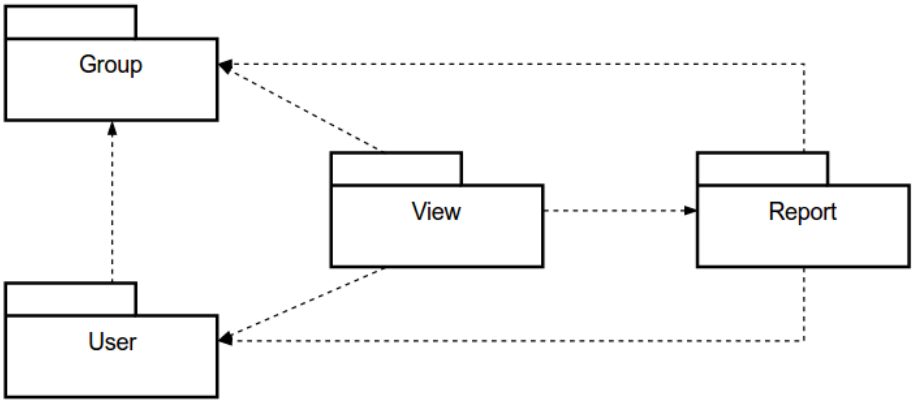

Рисунок 4.14 — Диаграмма пакетов

4.4.3 Диаграмма последовательностей уровня классов

На рис. 4.15 показана диаграмма последовательностей уровня классов для прецедента «Экспорт отчёта по пользователям в формате CSV». В альтернативном блоке «report exists» проверяется наличие ранее созданного экземпляра класса UsersReportsCSV. Если он существует, то экземпляр возвращается из внутреннего состояния класса ReportFactory. В противном случае создаётся новый объект, ссылка на который помещается в кэш класса ReportFactory. При следующих запросах объект пересоздаваться не будет. В этом заключается структурный паттерн приспособленец.

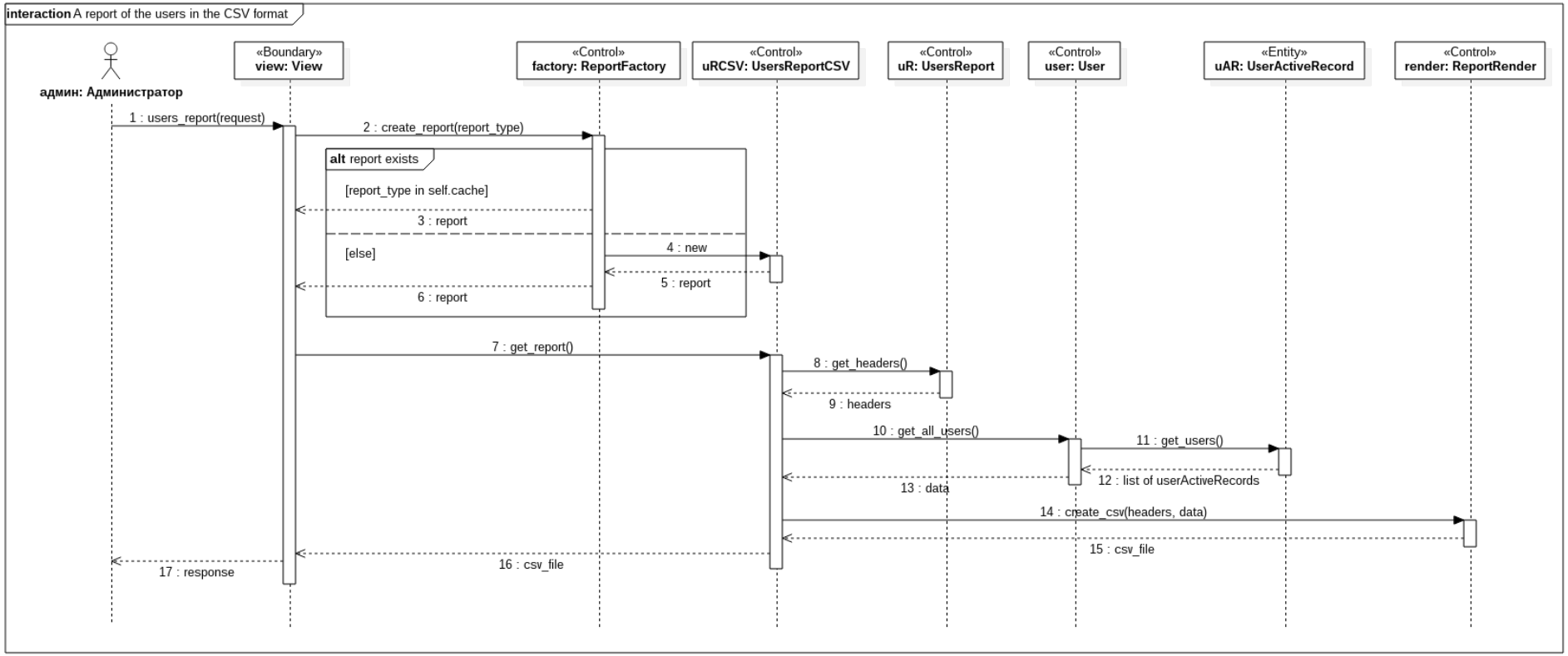

Рисунок 4.15 — Диаграмма последовательностей в терминах классов для прецедента «Экспорт отчёта по пользователям в формате CSV»

4.4.4 Диаграммы последовательностей уровня подсистем

Диаграмма последовательностей уровня подсистем для экспорта отчёта по группам показана на рис. 4.16, для экспорта отчёта по пользователям — на рис. 4.17. Тип отчёта (csv либо pdf) передаётся в requst в качестве get-параметра.

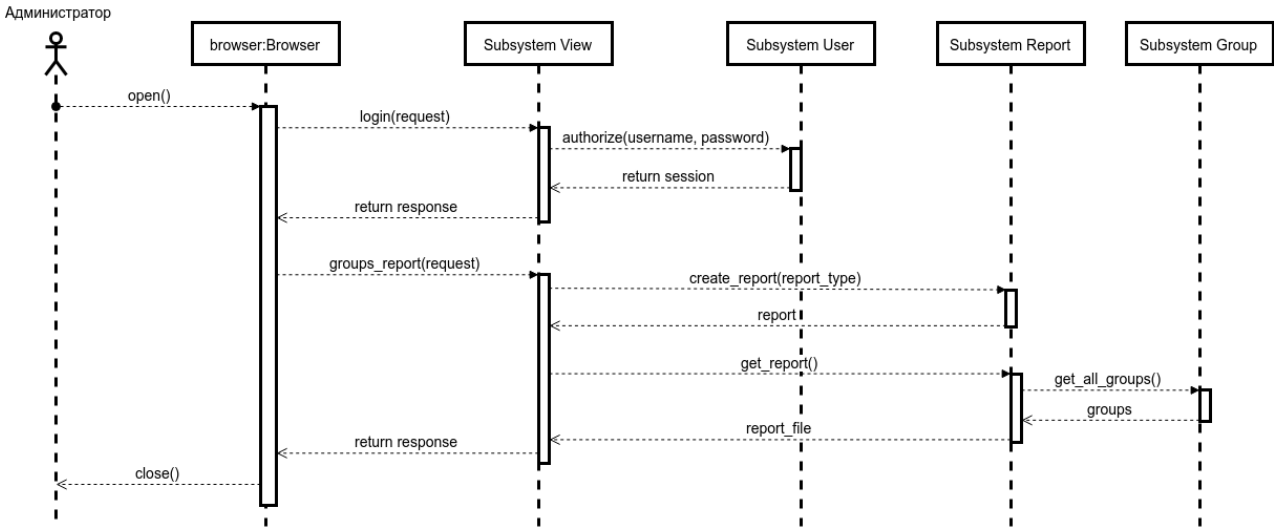

Рисунок 4.16 — Диаграмма последовательностей для экспорта отчётов по группам

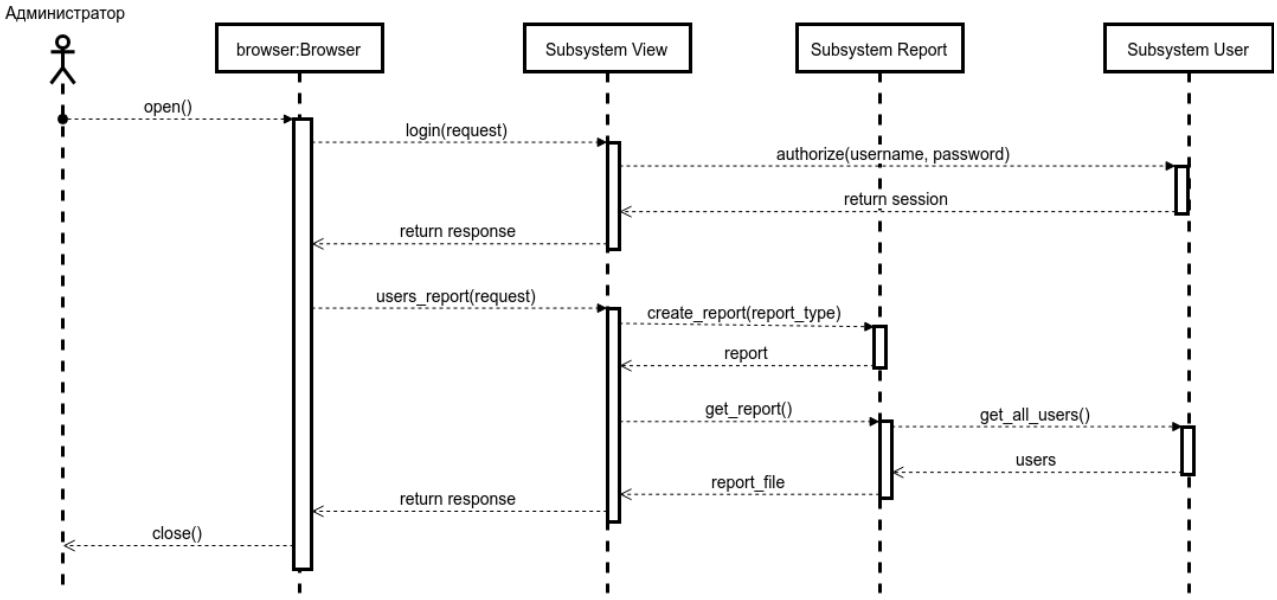

Рисунок 4.17 — Диаграмма последовательностей для экспорта отчёта по пользователям

4.5 Модель реализации для новых прецедентов

В ходе итерации была реализована новая подсистема отчётов — Report. В подсистему входят следующие классы:

- Report — класс с абстрактным методом get_report(), который должны переопределить наследники; описывает объект отчёта;
- UsersReport — абстрактный класс, содержащий методы get_headers() и get_data(); взаимодействует с подсистемой пользователей с целью извлечения информации о пользователях для формирования отчёта;
- UsersReportCSV — класс, реализующий интерфейс get_report(); формирует отчёт по пользователям в виде CSV-файла;
- UsersReportPDF — класс, реализующий интерфейс get_report(); формирует отчёт по пользователям в виде PDF-файла;
- GroupsReport — абстрактный класс, содержащий методы get_headers() и get_data(); взаимодействует с подсистемой групп с целью извлечения информации о группах для формирования отчёта;
- GroupsReportPDF — класс, реализующий интерфейс get_report(); формирует отчёт по группам в виде PDF-файла;
- GroupsReportCSV — класс, реализующий интерфейс get_report(); формирует отчёт по группам в виде CSV-файла;
- ReportRender — вспомогательный класс для генерации pdf и csv-файлов.

4.6 Описание тестовых вариантов

Предусловие для всех тестовых вариантов:

1) Открыть в браузере страницу файлового сервера;
2) Авторизоваться используя логин и пароль администратора.

Тестовые варианты описаны в таблице 4.1.

Таблица 4.1.

№	Тестируемый прецедент	Действия	Ожидаемый результат	Тест пройден
1	Экспорт отчёта по	Нажать на кнопку	На PC будет загружен файл	Да

	пользователям в PDF	«Отчёт по пользователям в PDF»	с отчётом по пользователям системы в формате PDF	
2	Экспорт отчёта по пользователям в CSV	Нажать на кнопку «Отчёт по пользователям в CSV»	На PC будет загружен файл с отчётом по пользователям системы в формате CSV	Да
3	Экспорт отчёта по группам в PDF	Нажать на кнопку «Отчёт по группам в PDF»	На PC будет загружен файл с отчётом по группам системы в формате PDF	Да
4	Экспорт отчёта по группам в CSV	Нажать на кнопку «Отчёт по группам в CSV»	На PC будет загружен файл с отчётом по группам системы в формате CSV	Да

4.7 Диаграммы классов с учётом реализованных паттерном (по этапу)

Диаграмма классов, входящих в подсистемы (пакеты) User, Group, View, Catalogue и Report показаны на рис. 4.18-4.22.

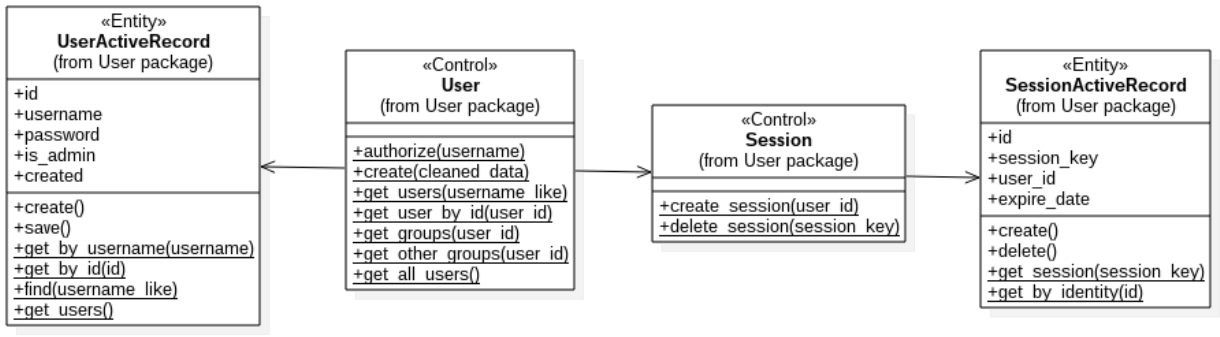

Рисунок 4.18 — Диаграмма классов подсистемы User

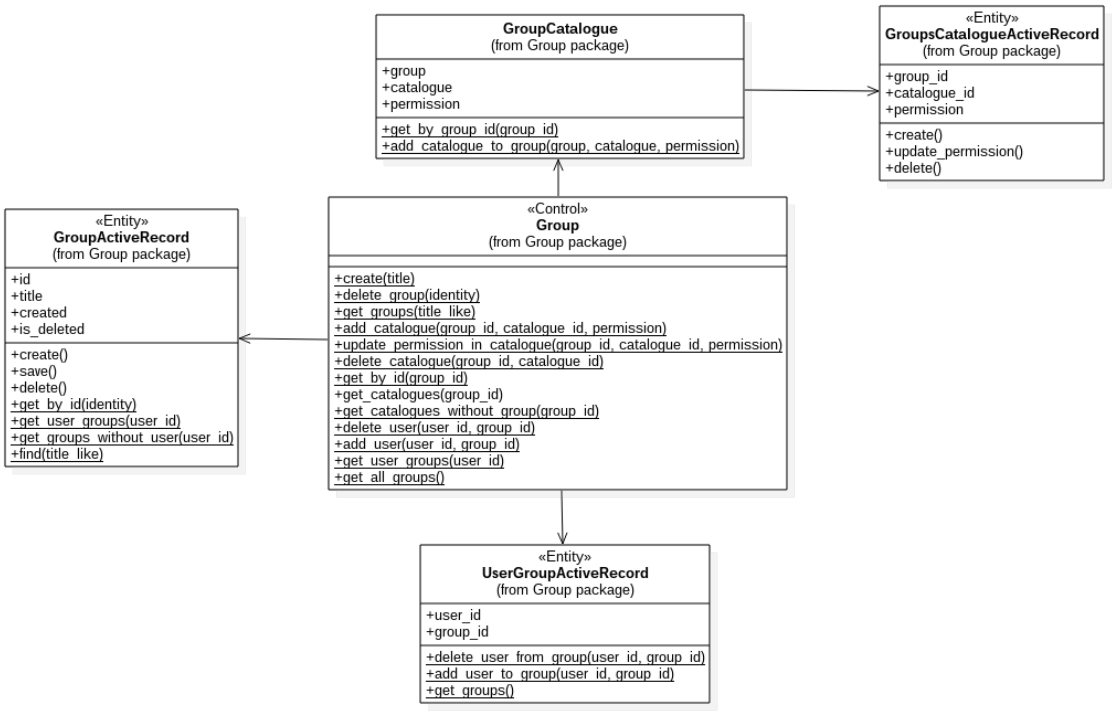

Рисунок 4.19 — Диаграмма классов подсистемы Group

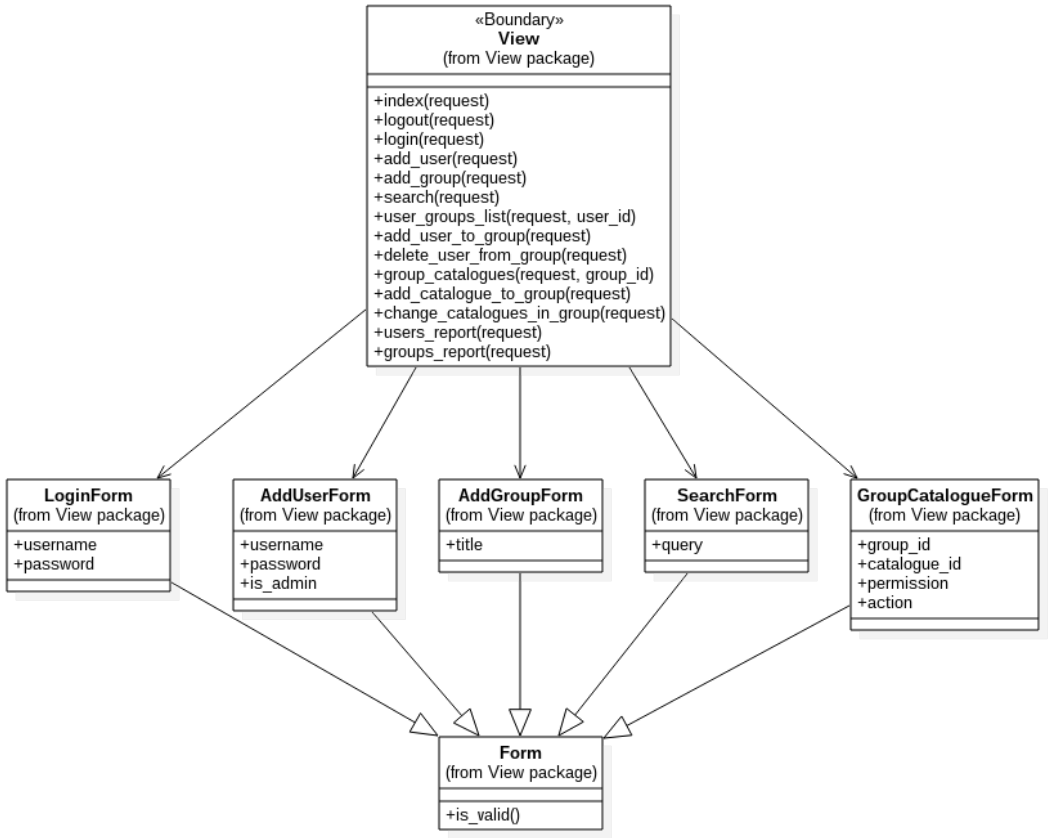

Рисунок 4.20 — Диаграмма классов подсистемы View

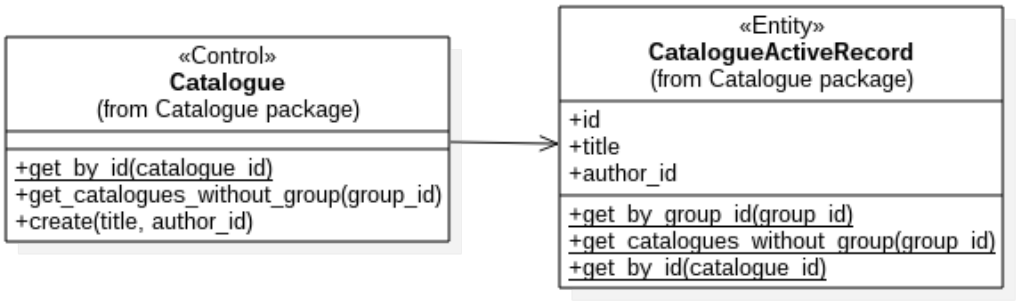

Рисунок 4.21 — Диаграмма классов подсистемы Catalogue

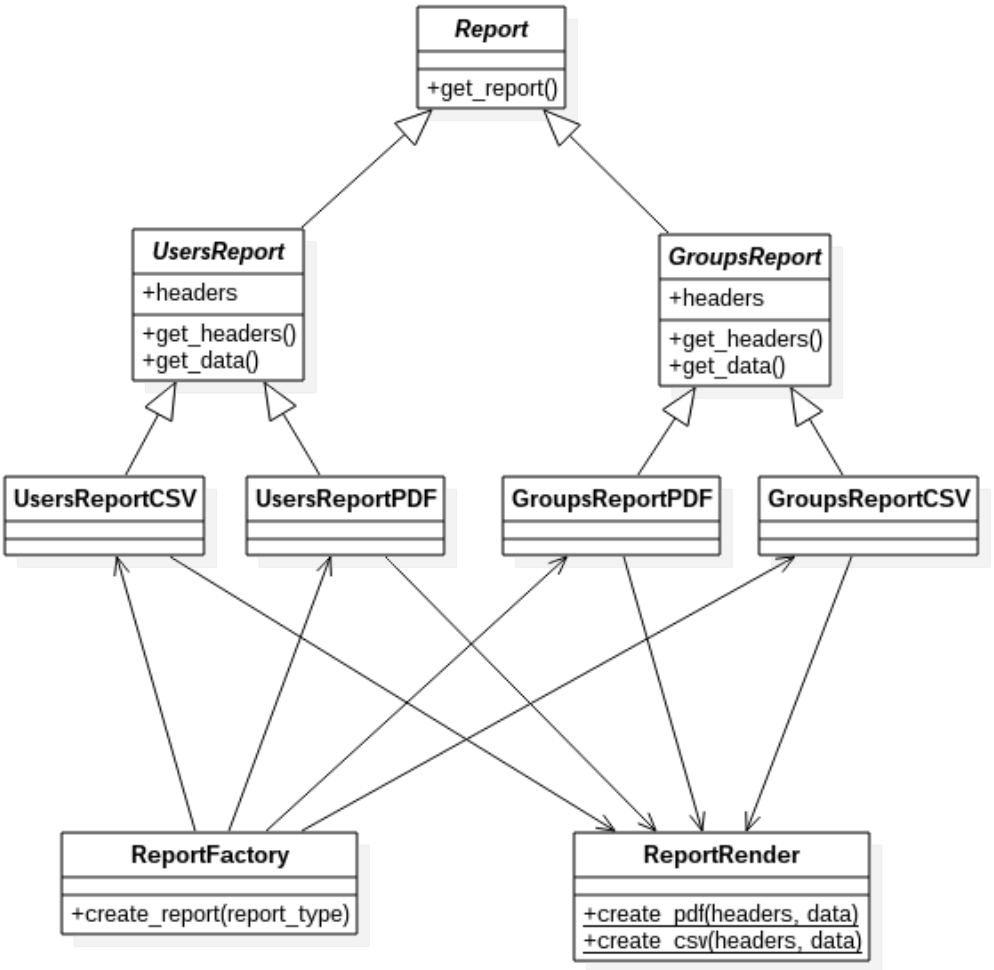

Рисунок 4.22 — Диаграмма классов подсистемы Report

4.8 Уточнённая диаграмма компонентов (по этапу)

Уточнённая диаграмма компонентов показана на рис. 4.23. На ней новая подсистема Report взаимодействует с подсистемами Group через интерфейс IGroup для извлечения информации о группах, с подсистемой User через интерфейс IUser для извлечения информации о пользователя. Подсистема View обращается к подсистеме Report через интерфейс IReport для получения отчётов.

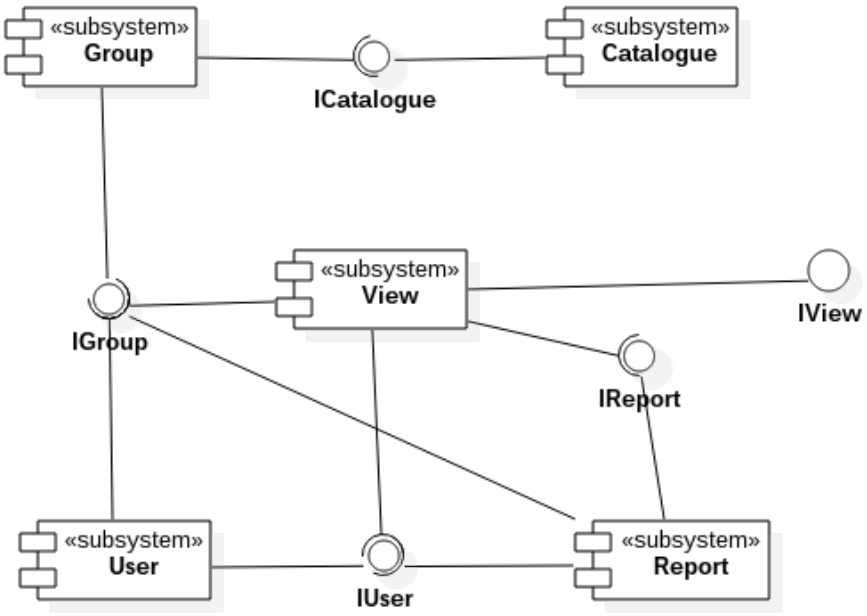

Рисунок 4.23 — Уточнённая диаграмма компонентов

4.9 Уточнённая диаграмма развёртывания (по этапу)

На диаграмме развёртывания на рис. 4.24 показано физическое расположение компонентов по узлам. Все разработанные компоненты располагаются на сервере приложения.

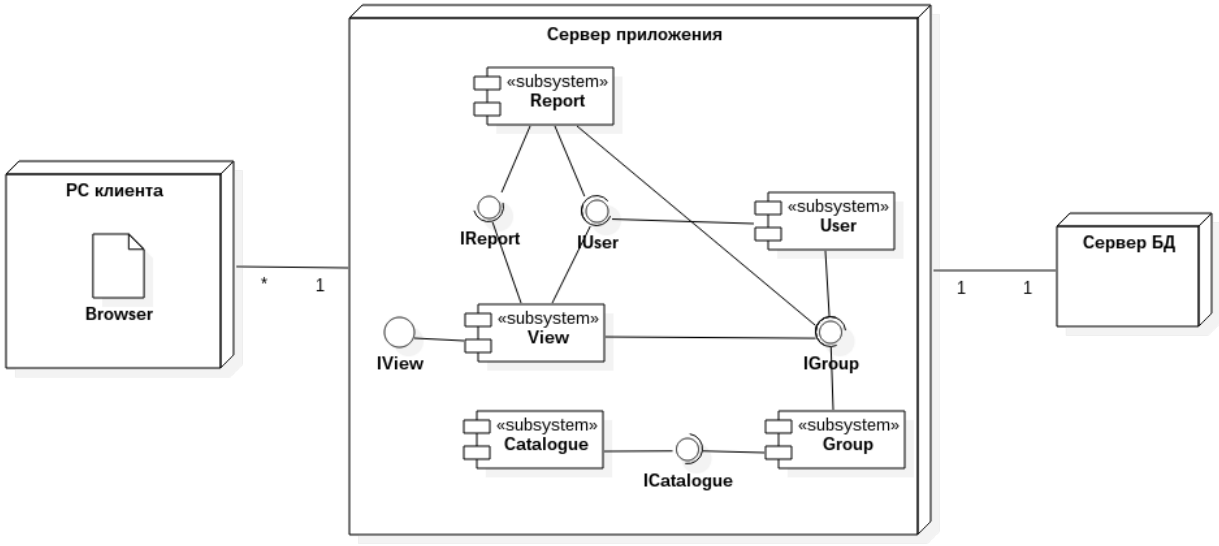

Рисунок 4.24 — Диаграмма развёртывания

4.10 Перечень и последовательность проведения тестов (по этапу)

Тестирование системы производилось с помощью тест-кейсов, описанных в таблице 4.2.

Предусловие для тест-кейсов:

1) Открыть в браузере страницу файлового сервера;
2) Авторизоваться в системе используя логин и пароль администратора.

Таблица 4.2. Перечень тестов

№	Тестируемый прецедент	Действия	Ожидаемый результат	Тест пройден
1	Создать пользователя	Выбрать в навигационном меню пункт «Добавить пользователя». Ввести имя нового пользователя, его пароль, выбрать значение параметра «это администратор». Нажать кнопку «Создать».	На экране отобразится сообщение «Пользователь <username> создан. Его id = <id>. *Примечание:* вместо <username> система выдаст имя введённого пользователя. <id> — идентификатор пользователя в системе.	Да
2	Создать группу	Выбрать в навигационном меню пункт «Добавить группу». Ввести заголовок новой группы. Нажать кнопку «Создать».	На экране отобразится сообщение «Группа "<заголовок>" была создана. Её id = <id>». Примечание: вместо <заголовок> система выдаст	Да

			ранее введённый заголовок группы. <id> — идентификатор группы в системе.	
3	Найти пользователя	Выбрать в навигационном меню пункт «Поиск». Ввести имя пользователя, который зарегистрирован в системе. Нажать на кнопку поиска.	Система отобразит список пользователей, имя которых удовлетворяет введённому имени пользователя по точному совпадению либо по его префиксу.	Да
4	Найти группу	Выбрать в навигационном меню пункт «Поиск». Ввести заголовок группы, существующей в системе. Нажать на кнопку поиска.	Система отобразит список групп, заголовок которых удовлетворяет введённому заголовку группы по точному совпадению либо по его префиксу.	Да
5	Добавить пользователя в группу	Выбрать в навигационном меню пункт «Поиск». Ввести имя пользователя, который зарегистрирован в системе. Нажать на кнопку поиска. Выбрать из списка требуемого пользователя. В открывшейся странице пользователя выбрать из списка групп нужную и нажать кнопку «Добавить в группу».	В группах пользователя появится выбранная группа.	Да
6	Удалить пользователя из группы	Выбрать в навигационном меню пункт «Поиск». Ввести имя пользователя, который зарегистрирован в системе. Нажать на кнопку поиска. Выбрать из списка требуемого пользователя. В открывшейся странице пользователя из раздела «Группы пользователя» выбрать группу, из которой требуется удалить пользователя, нажатием кнопки «Удалить из группы».	В группах пользователя исчезнет выбранная группа	Да
7	Найти каталоги, доступные группе	Выбрать в навигационном меню пункт «Поиск». Ввести заголовок группы, существующей в системе. Нажать на кнопку поиска. Выбрать из списка требуемую	В открывшейся странице в разделе «Каталоги, доступные группе» будут отображены каталоги и права доступа для заданной группы.	Да

		группу.		
8	Выдать права доступа группе к каталогу	Выбрать в навигационном меню пункт «Поиск». Ввести заголовок группы, существующей в системе. Нажать на кнопку поиска. Выбрать из списка требуемую группу. В открывшемся окне из раздела «Остальные каталоги» определить каталог, который требуется добавить к группе. Из выпадающего списка соответствующего каталога выбрать права доступ. Нажать кнопку «Добавить	В разделе «Каталоги, доступные группе» появится добавленный каталог с выбранными правами доступа.	Да

5 5 Этап внедрения (переход)

5.1 Перечень программ и рекомендации по установке

Для работы системы требуются следующие программы:

— Интерпретатор языка Python 3.5.2 либо выше;

— СУБД MySQL 5.6 либо выше;

Интерпретатор языка Python3 может быть установлен по инструкции с официального сайта https://www.python.org/; инструкция по установке СУБД MySQL располагается также на официальном сайте https://www.mysql.com/downloads/.

После установки языка разработки и СУБД необходимо установить зависимости (библиотеки) с помощью менеджера пакетов pip из файла requirements.txt. Менеджер пакетов pip поставляется вместе с языком python. Установка пакетов осуществляется с помощью следующей команды:

```
pip3 install reqirements.txt
```

5.2 Перечень документации для пользователей и заказчиков

Для заказчиков при сдаче программного продукта предоставляются следующие документы:

— Техническое задание;

— Программа и методика испытаний автоматизированной системы;

— Акт приёмки автоматизированной системы в опытную эксплуатацию.

Для пользователей программного продукта предоставляются следующие документы:

— Описание системы;

— Руководство оператора БД;

— Руководство системного администратора;

— Руководство программиста;

— Руководство администратора;

— Руководство пользователя;

— Руководство по эксплуатации автоматизированной системы и решению внештатных ситуаций.

5.3 Рекомендации по внедрению

Особые рекомендации по внедрению к данному продукту не применяются. Список необходимых программ и пакетов представлен в разделе 5.1.

6 Вывод

В ходе выполнения курсовой работы были достигнуты следующие цели:

— Изучены теоретические принципы унифицированного процесса разработки СОИУ и составляющих его этапов на примере разработки файлового сервера;

— Получены практические навыки применения шаблонов при проектировании и разработки СОИУ;

— Освоены CASE-средства для разработки СОИУ.

7 Список литературы

1. Унифицированный процесс разработки программного обеспечения: учебное пособие / Виноградова М.В., Белоусова В.И. – М.: МГТУ им.Н.Э. Баумана. – 2015 г. http://ebooks.bmstu.ru/catalog/193/book1303.html

2. Фаулер М. Архитектура корпоративных приложений. - М.:Изд.дом Вильямс. - 2008 г.

3. Орлов С. Технологии разработки программного обеспечения. — Спб.: Питер, 2002. — 464 с.