ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Глоссарий

Латентность географического положения — увеличение времени отклика приложения, обуславливаемое географическим положением элементов системы или пользователя.

Медиана времени отклика — максимальное время предоставления данных пользователю для каждого из как минимум половины пользователей.

Публикация — документ, доступный для массового использования (ГОСТ 7.0-99).

СОА — Сервис-ориентированная архитектура — модульный подход к разработке ПО, базирующийся на обеспечении удалённого по стандартизированным протоколам использования распределённых, слабо связанных легко заменяемых компонентов (сервисов) со стандартизированными интерфейсами.

Тег — идентификатор для категоризации, описания или поиска.

Узел системы — сервер, соединённый с другими устройствами как часть компьютерной сети.

UUID4 — формат представления уникальных идентификаторов в СОА.

1 Введение

1.1 Наименование программы

Наименование программы — «Распределённая система ведения онлайн-дневников (блогов)» (далее — Система).

1.2 Краткое описание предметной области

В современном мире одну из лидирующих позиций занимает информационное пространство — это публичная площадка в сети Интернет, где человек излагает свои мысли. Множество людей наблюдает за подобными публичными площадками и людьми, которые их ведут, в различных социальных сетях.

Один из наиболее интересных видов информационного пространства — блог. Термин «блог» произошёл от английского weblog («logging the web» — записывать события в сеть). Впервые его использовал американский программист Йори Баргер в 1997 году для обозначения сетевого дневника.

Блог — это личный дневник, который размещается в интернете, имеет яркую индивидуальность, оригинальное содержание и свою аудиторию. В отличие от реальных дневников, которые читают только сами авторы, записи интернет-дневников принято выкладывать на всеобщее обозрение.

Людей, ведущих блог, называют блогерами. Блогер — это любой человек, который ведёт собственный электронный дневник и является его администратором.

1.3 Существующие аналоги

Среди аналогов разрабатываемой Системы можно отметить «LiveJournal», «Лайфхакер» и «Blogger». К недостаткам можно отнести то, что данные порталы являются зарубежными, что при текущих обстоятельствах ставит под вопрос стабильность работы на территории РФ.

Отличием данной разработки является сфокусированность на рускоязычную аудиторию. К преимуществам можно отнести возможность предоставления доступа к истории прочитанных публикаций, а также стабильность работы на территории России.

Описание системы

Система должна представлять собой публичную площадку в сети Интернет для обмена информацией посредством ведения онлайн-дневников и обсуждения прочитанного в комментариях к публикациям. Пользователь волен публиковать собственные записи как автор или подписываться на других пользователей, а также на категории (теги) для формирования своей ленты публикаций на прочтение, оценивать публикации и комментарии, просматривать самые популярные публикации за всё время.

2 Основания для разработки

Разработка ведётся в рамках выполнения лабораторных работ по курсу «Методология программной инженерии» на основании задания на курсовой проект по дисциплине «Распределённые системы обработки информации» на кафедре «Программное обеспечение ЭВМ и информационные технологии» факультета «Информатика, искусственный интеллект и системы управления» федерального государственного бюджетного образовательного учреждения высшего образования «Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)».

3 Назначение разработки

Главное назначение разрабатываемой Системы — предоставление пользователю возможности ведения собственного онлайн-дневника; просмотра публикаций других пользователей; формирования личного круга интересов посредством подписок на определённых авторов и теги; обсуждения публикаций в комментариях к ним; просмотра истории прочитанных публикаций. У пользователей должна быть возможность оценивать публикации и комментарии, сортировать их по дате и рейтингу. Система должна позволять фильтровать публикации по определённым тегам, а также формировать топ-лист самых

популярных публикаций за всё время.

4 Требования к программе

4.1 Общие требования к системе

— Разрабатываемое ПО должно обеспечивать функционирование Системы в режиме 24/7/365 со среднеговодым временем доступности не менее 99,9 %. Допустимое время, в течение которого Система недоступна, за год должно составлять менее $24 \cdot 365 \cdot 0{,}001 = 8{,}76$ часа.
— Время восстановления Системы после сбоя не должно превышать 15 минут.
— Каждый узел должен автоматически восстанавливаться после сбоя.
— Обеспечить безопасность работоспособности Системы за счёт отказоустойчивости узлов.

4.2 Требования к функциональным характеристикам

— Медиана времени отклика Системы на запросы пользователя на получение информации не должна превышать 3 секунд без учёта латентности географического расположения узла.
— Медиана времени отклика Системы на запросы, добавляющие или изменяющие информацию в Системе, не должна превышать 5 секунд без учёта латентности географического расположения узла.
— Система должна обеспечивать возможность запуска в современных браузерах: не менее 94,01 % пользователей Интернета должны иметь возможность пользоваться порталом без какой-либо деградации функционала.

4.3 Функциональные требования

Система должна:

а) обеспечивать
   1) регистрацию и авторизацию пользователей с валидацией вводимых данных как через интерфейс приложения;
   2) аутентификацию пользователей;
   3) разделение пользователей на три роли:
      — Гость (неавторизованный Пользователь),
      — Пользователь,
      — Администратор;

      причём Пользователю доступны все функции Гостя, Администратору — все функции Пользователя;

б) предоставлять Гостю следующие функции:
   1) получение списка публикаций
      — по автору,
      — по тегу,
      — всех авторов;
   2) просмотр полного текста публикаций и комментариев к ней;
   3) сортировка публикаций/комментариев по дате/рейтингу;

в) предоставлять Пользователю следующие функции:
   1) добавление публикаций в свой онлайн-дневник;
   2) редактирование публикаций из своего онлайн-дневника;
   3) удаление публикаций из своего онлайн-дневника;
   4) управление подписками на других авторов;
   5) управление подписками на теги;
   6) получение списка публикаций
      — по авторам и тегам, на которые подписан Пользователь;
      — просмотренных;
   7) добавление комментария к публикации;
   8) редактирование своего комментария;

9) удаление своего комментария;
10) оценивать «плюсом» или «минусом» публикации/комментарии;

г) предоставлять Администратору следующие функции:
- блокирование части публикации, комментария или пользователя;
- отображение статистики просмотров публикаций (см. 4.3.2 пункт г).

4.3.1 Требования к организации входных данных

Входные данные Системы:

а) пользователь:
- ФИО — строка, не более 256 символов;
- адрес электронный почты — строка, не более 256 символов;
- пароль — строка, не менее 8 и не более 256 символов.

б) Публикация:
- ФИО автора — строка, не более 256 символов;
- заголовок — строка, не более 256 символов;
- тело — строка, не более 65536 символов;
- теги — список из не более 32 строк, каждая из которых состоит из не более 32 символов;
- рейтинг — целое число.

в) Комментарий:
- ФИО автора — строка, не более 256 символов;
- текст — строка, не более 4096 символов.

г) Тег:
- название категории — строка, не более 32 символов.

4.3.2 Требования к организации выходных данных

Выходными данными Системы являются веб-страницы. В зависимости от запроса Пользователя они могут содержать следующие сведения:

а) Публикация:
   — ФИО автора — строка, не более 256 символов;
   — заголовок — строка, не более 256 символов;
   — тело — строка, не более 65536 символов;
   — теги — список из не более 32 строк, каждая из которых состоит из не более 32 символов;
   — рейтинг — целое число;
   — дата публикации (генерируется автоматически) — строка, соответствующая формату стандартного вывода POSIX-утилиты date;
   — идентификатор (генерируется автоматически) — UUID4.

б) Комментарий:
   — ФИО автора — строка, не более 256 символов;
   — текст — строка, не более 4096 символов.
   — дата (генерируется автоматически) — строка, соответствующая формату стандартного вывода POSIX-утилиты date;
   — идентификатор (генерируется автоматически) — UUID4.

в) Тег:
   — название категории — строка, не более 32 символов.

г) Статистика просмотров публикации:
   — всё из пункта а)
   — таблица с атрибутами:
      — ФИО Пользователя — строка, не более 256 символов;
      — адрес электронной почты Пользователя — строка, не более 256 символов.
      — дата просмотра (генерируется автоматически) — строка, соответствующая формату стандартного вывода POSIX-утилиты date.

4.4 Требования к программной реализации

а) Требуется использовать СОА для реализации Системы.

б) Каждый сервис реализует свою функциональность и взаимодействует с другими сервисами по протоколу HTTP (придерживаться нотации RESTful), либо через очередь.

в) Каждый сервис при необходимости имеет доступ к связанной с ним базе данных, но не должен иметь доступ к базам данных других сервисов.

г) Предусмотреть ситуацию недоступности систем, обработку таймаутов и ошибок сервисов. В случае ошибки/недоступности некритичного функционала выполнять деградацию функциональности.

д) Необходимо предусмотреть авторизацию пользователей через интерфейс приложения.

е) Для запросов, выполняющих обновление данных на нескольких узлах распределенной системы, в случае недоступности одной из систем, необходимо выполнять полный откат транзакции.

ж) Приложение должно поддерживать возможность горизонтального и вертикального масштабирования за счёт увеличения количества функционирующих узлов и совершенствования технологий реализации компонентов и всей архитектуры системы.

4.5 Топология системы

Топология разрабатываемой системы изображена на рисунке 1.

Разрабатываемая система состоит из фронтенда и 5 подсистем:

— Сервис публикаций;
— Сервис подписок;
— Сервис статистики;
— Сервис регистрации и авторизации;
— Сервис-координатор.

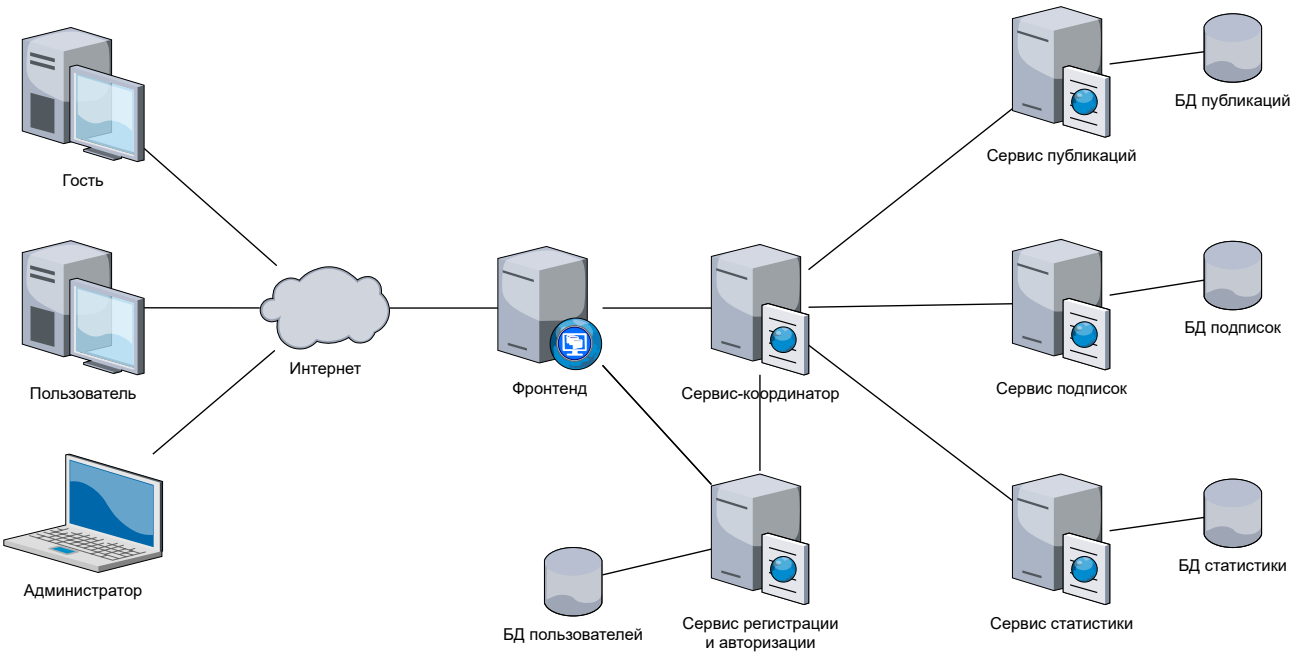

Рисунок 1 — Топология системы

4.5.1 Общие требования к подсистемам

а) Фронтенд – серверное приложение, при разработке которого следует учесть следующие нюансы:
   1) Фронтенд должен принимать запросы по протоколу HTTP и формировать ответ пользователю в формате HTML-страниц (использование CSS обязательно);
   2) Фронтенд является посредником между пользователями, передавая их запросы последовательно на сервис агрегации запросов;
   3) Запросы от фронтенда могут быть только к сервису-координатору либо сервису регистрации и авторизации для получения токена.

б) К реализации бэкендов должны быть предъявлены следующие требования:
   1) Приём и возврат данных должен происходить в формате JSON по протоколу HTTP;
   2) Все запросы, кроме авторизации пользователя, проходят через сервис-координатор.
   3) Если результаты работы сервиса необходимо сохранять в базе

данных, то доступ к ней должен осуществляться по протоколу HTTP. Доступ к базе данных может осуществляться только из подсистем, работающих напрямую с данными её таблиц.

4.5.2 Функциональные требования к сервисам

а) Сервис публикаций — отвечает за добавление публикаций и хранение информации о них. В базе данных, ассоциированной с сервисом, должны храниться сущности:
   - Публикация, с обязательными полями
     - идентификатор;
     - автор;
     - заголовок;
     - тело публикации;
     - дата добавления;
     - рейтинг;
   - Комментарий, с обязательными полями:
     - идентификатор;
     - автор;
     - текст комментария;
     - дата добавления;
     - рейтинг;
   - Тег, с обязательным полем
     - название категории.

   Сервис должен реализовывать следующий функционал:
   1) добавление/изменение/удаление публикации;
   2) добавление/изменение/удаление комментария;
   3) добавление/изменение/удаление тега (для Администратора);
   4) получение публикаций по автору/тегу (или всех), с сортировкой по дате или рейтингу;
   5) получение отдельной публикации и комментариев к ней.

б) Сервис подписок — осуществляет формирование персонального информационного окружения пользователя. Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:
   - идентификатор пользователя;
   - идентификатор автора или тега, на который подписывается читатель.

   Сервис должен реализовывать следующий функционал:
   1) добавление/удаление подписки;
   2) получение идентификаторов авторов или тегов, на которые подписан пользователь;
   3) получение всех подписок пользователя.

в) Сервис статистики должен реализовывать следующий функционал:
   - получение статистики просмотров публикации;

г) Сервис регистрации и авторизации. Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:
   - ФИО;
   - адрес электронной почты;
   - пароль;

   Сервис должен реализовывать следующий функционал:
   1) проверка существования пользователя;
   2) регистрация пользователя;
   3) аутентификация пользователя;
   4) удаление пользователя.

д) Сервис-координатор должен реализовать диспетчеризацию запросов.

4.6 Требования к надёжности

Надёжное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведён ниже:

- организация бесперебойного питания технических средств;

— использование лицензионного программного обеспечения;
— регулярным выполнением ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов.

В ситуации недоступности систем, выводится соответствующее сообщение об ошибке. В случае ошибки/недоступности некритичного функционала, выполняется деградация функциональности.

5 Требования к программной документации

Исполнитель должен подготовить и передать Заказчику следующие документы:

— руководство администратора Системы;
— руководство пользователя Системы.

КОНСТРУКТОРСКИЙ РАЗДЕЛ

Концептуальный дизайн

Концептуальный дизайн системы содержит наиболее общие схемы описания функционала приложения с точки зрения пользователей. Одной из таких схем является IDEF0-модель и графические модели, входящие в неё. На рисунке 2 отображена контекстная диаграмма верхнего уровня, которая обеспечивает наиболее общее или абстрактное описание работы системы. Данный вид диаграммы позволяет формализовать описание запросов пользователя и ответов системы на данные запросы, отобразив систему в виде «чёрного» ящика.

Рисунок 2 — Концептуальная модель системы в нотации IDEF0

Для уточнения деталей работы системы применяется декомпозиция функций, отображённых на диаграмме верхнего уровня, при помощи создания дочерних диаграмм. В качестве примера на рисунке 3 изображена дочерняя диаграмма, которая определяет последовательность выполнения операций в системе при обработке запроса пользователя на получение информационного

контента.

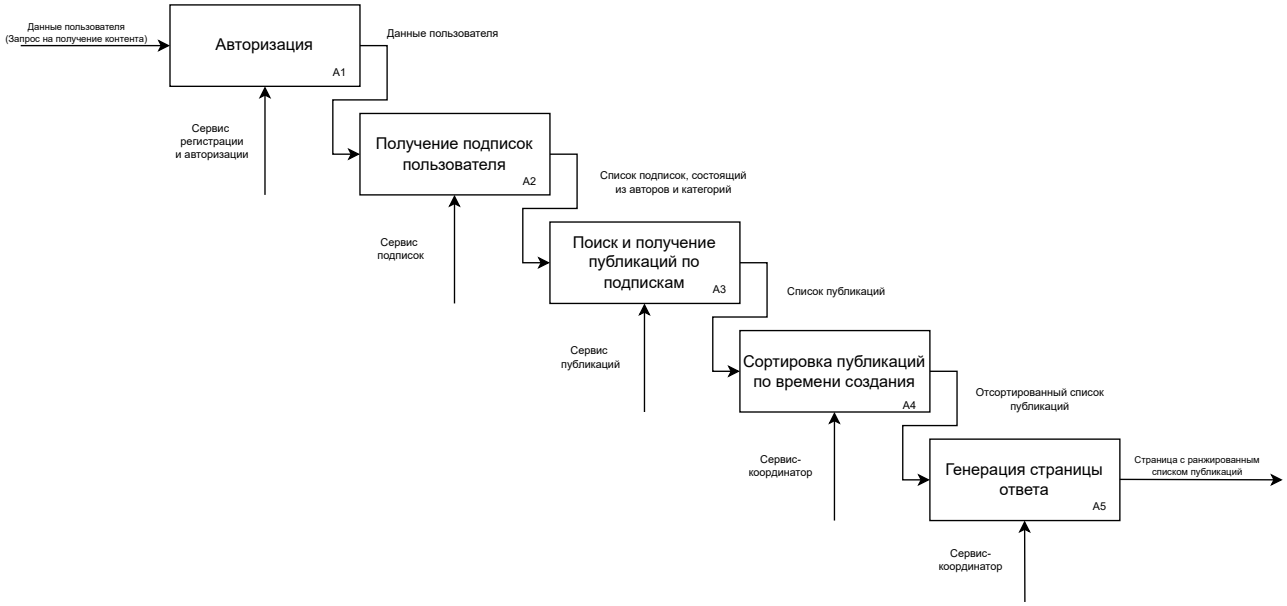

Рисунок 3 — Детализированная концептуальная модель системы в нотации IDEF0

Детализированная диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский блок, но описывает ее более подробно. Поэтому такая диаграмма может быть создана для любого из запросов, отображённых на диаграмме верхнего уровня. Каждый из блоков детализированной диаграммы может быть в свою очередь также описан при помощи дочерней диаграммы.

Сценарии функционирования системы

Сценарии функционирования или использования системы описывают конкретную последовательность действий, иллюстрирующую поведение пользователя при работе с приложением. Далее приведены подробные сценарии основных возможных действий пользователя.

Регистрация пользователя

а) Пользователь нажимает на кнопку «Войти» в интерфейсе приложения.
б) Система перенаправляет пользователя на страницу авторизации, которая содержит поля для заполнения его данных.
в) Пользователь вводит данные в форму и для завершения регистрации нажимает на кнопку «Регистрация», тем самым подтверждая верность своих данных, а также согласие на их обработку и хранение.
г) Если пользователь с введённым для регистрации именем уже существует, то пользователь перенаправляется на страницу ошибки. При успешной регистрации система перенаправляет пользователя на страницу своего профиля.

Авторизация пользователя

а) Пользователь нажимает на кнопку «Войти» в интерфейсе приложения.
б) Система перенаправляет пользователя на страницу авторизации, которая содержит поля для заполнения его логина и пароля.
в) Пользователь завершает работу с формой авторизации нажатием кнопки «Войти».
г) При обнаружении ошибки в данных, система перенаправляет пользователя на страницу ошибки; при совпадении данных с записью в базе данных аккаунтов пользователь получает доступ к системе.

Получение списка подписок

а) Авторизованный пользователь нажимает на кнопку «Подписки».
б) Система перенаправляет пользователя на страницу, содержащую список пользователей и категорий, на которые он подписан.

Просмотр ранжированного списка публикаций на основе подписок пользователя

а) Авторизованный пользователь нажимает на кнопку «Лента».
б) Система перенаправляет пользователя на страницу, которая содержит список публикаций, ранжированных по дате создания.

Получение списка просмотренных публикаций

а) Авторизованный пользователь нажимает на кнопку «История просмотров».
б) Система перенаправляет пользователя на страницу, содержащую список просмотренных пользователем статей, ранжированных по дате просмотра.

Получение статистики

а) Администратор нажимает на кнопку «Панель администратора».
б) Система перенаправляет администратора на страницу просмотра статистики.

Диаграммы прецедентов

Графически сценарии функционирования системы можно представить при помощи диаграмм прецедентов. Они позволяют схематично отобразить типичные сценарии взаимодействия между клиентами и приложением. В системе выделены 3 основных роли: Гость, Пользователь и Администратор, диаграммы прецедентов для этих ролей изображены на рисунках 4, 5 и 6.

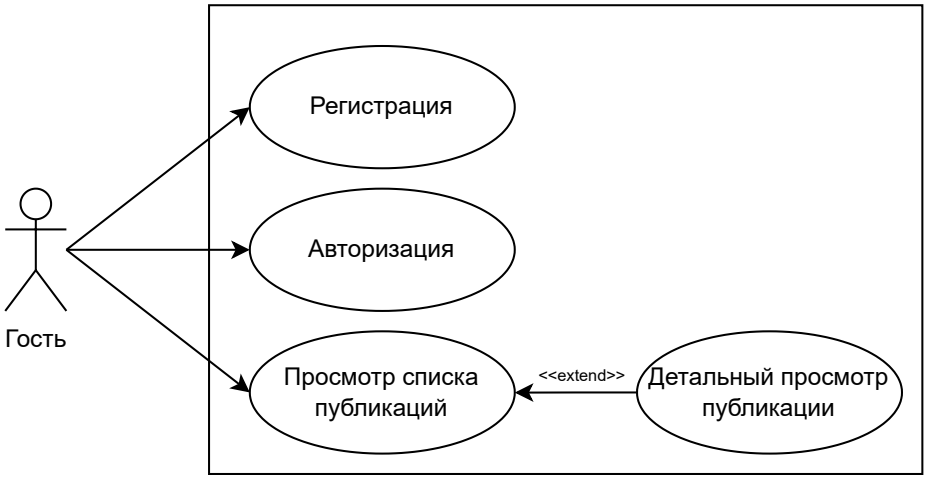

Рисунок 4 — Диаграмма прецедентов с точки зрения Гостя

Рисунок 5 — Диаграмма прецедентов с точки зрения Пользователя

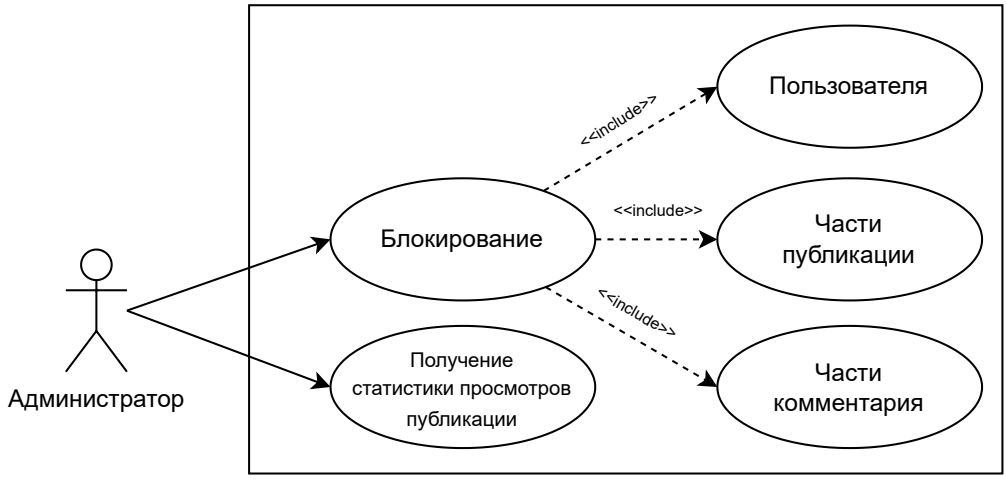

Рисунок 6 — Диаграмма прецедентов с точки зрения Администратора

Спецификации сценариев

Приведённые сценарии могут иметь как основной поток выполнения, который выполняется чаще всего, так и альтернативные потоки, описывающие выполнение запроса при отклонении от основного хода сценария. Все возможные ходы выполнения сценария описываются при помощи спецификаций. Примеры спецификаций для описанных выше сценариев приведены в данном разделе.

Таблица 1 — Спецификация сценария «Регистрация»

Действие пользователя	**Отклик системы**	
	Нормальный ход сценария	**Альтернативный ход сценария**
Пользователь нажимает кнопку «Войти»	Открывается страница для ввода данных	Открывается страница для ввода данных
Пользователь вводит данные в поля и нажимает кнопку «Регистрация»	Открывается страница с профилем созданного пользователя	Открывается страница с сообщением об ошибке, что пользователь с такими данными существует

Таблица 2 — Спецификация сценария «Авторизация»

Действие пользователя	**Отклик системы**	
	Нормальный ход сценария	**Альтернативный ход сценария**
Пользователь нажимает кнопку «Войти»	Открывается страница для ввода данных	Открывается страница для ввода данных
Пользователь вводит данные в поля и нажимает кнопку «Войти»	Открывается страница профиля пользователя	Открывается страница с сообщением об ошибке о неверно введённых данных

Таблица 3 — Спецификация сценария «Просмотр своих подписок»

Действие пользователя	**Отклик системы**	
	Нормальный ход сценария	**Альтернативный ход сценария**
Пользователь нажимает кнопку «Подписки»	Открывается страница со списком пользователей и категорий, на которые подписан пользователь	Открывается страница с сообщением об отсутствии подписок ИЛИ Открывается страница с сообщением, что сервис подписок недоступен

Таблица 4 — Спецификация сценария «Просмотр своей ленты информационного контента»

Действие пользователя	**Отклик системы**	
	Нормальный ход сценария	**Альтернативный ход сценария**
Пользователь нажимает кнопку «Лента»	Открывается страница со списком публикаций авторов и по категориям, на которые подписан пользователь	Открывается страница с сообщением, что у пользователя нет подписок ИЛИ Открывается страница ошибки с сообщением, что сервис публикаций недоступен

Таблица 5 — Спецификация сценария «История просмотренных публикаций»

Действие пользователя	**Отклик системы**	
	Нормальный ход сценария	**Альтернативный ход сценария**
Пользователь нажимает кнопку «История просмотров»	Открывается страница со списком просмотренных публикаций	Открывается страница с сообщением, что история пуста

Таблица 6 — Спецификация сценария «Получение статистики»

Действие пользователя	**Отклик системы**	
	Нормальный ход сценария	**Альтернативный ход сценария**
Пользователь авторизуется с правами Администратора	Открывается страница с профилем Администратора	Открывается страница с профилем Администратора
Пользователь нажимает кнопку «Панель администратора	Открывается страница, содержащая статистику просмотров публикаций	Открывается страница ошибки с сообщением, что сервис статистики недоступен

Логический дизайн

В процессе создания концептуального дизайна системы были отражены основные сценарии взаимодействия пользователя и системы. В разделе логического дизайна представлена организация элементов системы и их взаимодействие между собой. На основе функциональных требований к выделенным подсистемам, а также объектов, о которых необходимо хранить данные в системе, была разработана схема данных приложения. Результат её проектирования отображён на условной ER-диаграмме, представленной на рисунке 7. На данной схеме прямоугольниками обозначены ключевые сущности, а ромбами — связи между ними. Участие сущности в отношении с другой сущностью

отмечается линией, соединяющей их. Число, располагающееся около линии, означает тип связи между соединёнными сущностями.

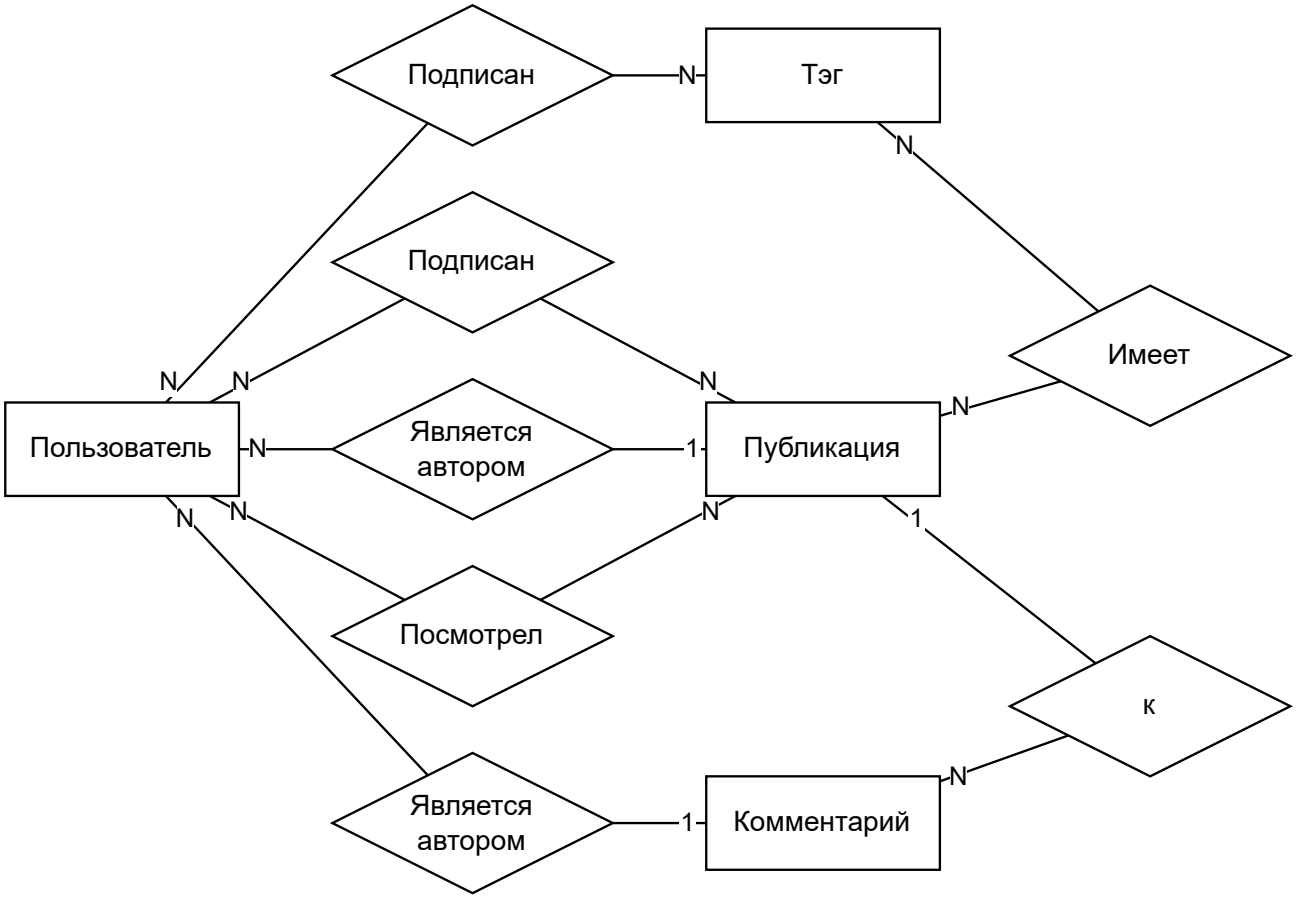

Рисунок 7 — ER-диаграмма данных системы

На следующей стадии проектирования, добавив в схему данных атрибуты сущностей, получаем схему базы данных, которая изображена на рисунке 8.

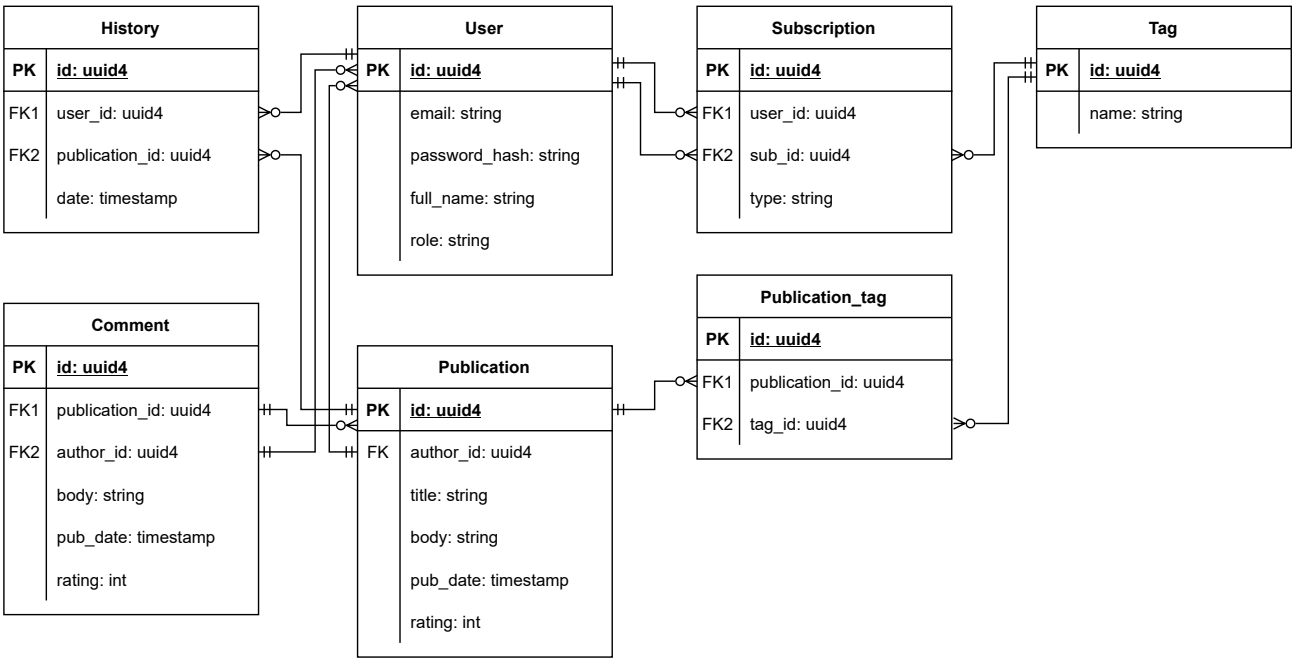

Рисунок 8 — Схема базы данных системы

Далее приводятся спецификации таблиц базы данных, приведённых на рисунке 8.

Таблица 7 — Спецификация таблицы User

Имя атрибута	**Тип атрибута**	**Описание атрибута**
id	UUID4	Идентификатор пользователя
email	string	Электронная почта
password_hash	string	Хеш пароля
full_name	string	ФИО
role	string	Роль (Пользователь или Администратор)

Таблица 8 — Спецификация таблицы Tag

Имя атрибута	**Тип атрибута**	**Описание атрибута**
id	UUID4	Идентификатор тега
name	string	Категория

Таблица 9 — Спецификация таблицы Publication

Имя атрибута	**Тип атрибута**	**Описание атрибута**
id	UUID4	Идентификатор публикации
author_id	UUID4	Идентификатор автора
title	string	Заголовок
body	string	Тело
pub_date	timestamp	Дата публикации
rating	int	Рейтинг

Таблица 10 — Спецификация таблицы Comment

Имя атрибута	**Тип атрибута**	**Описание атрибута**
id	UUID4	Идентификатор комментария
author_id	UUID4	Идентификатор автора
publication_id	UUID4	Идентификатор публикации
body	string	Тело
pub_date	timestamp	Дата публикации комментария
rating	int	Рейтинг

Таблица 11 — Спецификация таблицы Subscription

Имя атрибута	**Тип атрибута**	**Описание атрибута**
id	UUID4	Идентификатор сущности
user_id	UUID4	Идентификатор пользователя
sub_id	UUID4	Идентификатор объекта подписки
type	string	Тип подписки (Тег или Автор)

Таблица 12 — Спецификация таблицы History

Имя атрибута	**Тип атрибута**	**Описание атрибута**
id	UUID4	Идентификатор сущности
user_id	UUID4	Идентификатор пользователя
publication_id	UUID4	Идентификатор публикации
date	string	Дата просмотра

Таблица 13 — Спецификация таблицы Publication_tag

Имя атрибута	**Тип атрибута**	**Описание атрибута**
id	UUID4	Идентификатор сущности
publication_id	UUID4	Идентификатор публикации
tag_id	UUID4	Идентификатор тега

Диаграммы последовательности действий

Для описания поведения компонентов системы на единой оси времени используются диаграммы последовательности действий, при помощи которых можно описать последовательность действий для каждого прецедента, необходимую для достижения цели. Например, на рисунке 9 изображён процесс получения списка публикаций пользователем на основе его подписок.

Рисунок 9 — Диаграмма последовательности действий при запросе пользователем публикаций на основе его подписок

Сервис-координатор отправляет запрос сервису подписок на получение списка авторов и тегов, на которые подписан пользователь. После получения данных сервис-координатор обращается к сервису публикаций, чтобы получить список публикаций по переданным авторам и тегам. После окончания данной процедуры главный сервис ранжирует список по времени публикации, формирует веб-страницу и возвращает её пользователю.

Диаграмма потоков данных

Рассматриваемая система предполагает распределённое хранение данных. Все данные системы предполагают хранение в единой базе данных, хранилищами данных являются таблицы. Диаграмма потоков данных, представленная на рисунке 10, отображает модель информационной системы с точки зрения хранения, передачи и обработки данных во время обработки запроса пользователя на получение новостей.

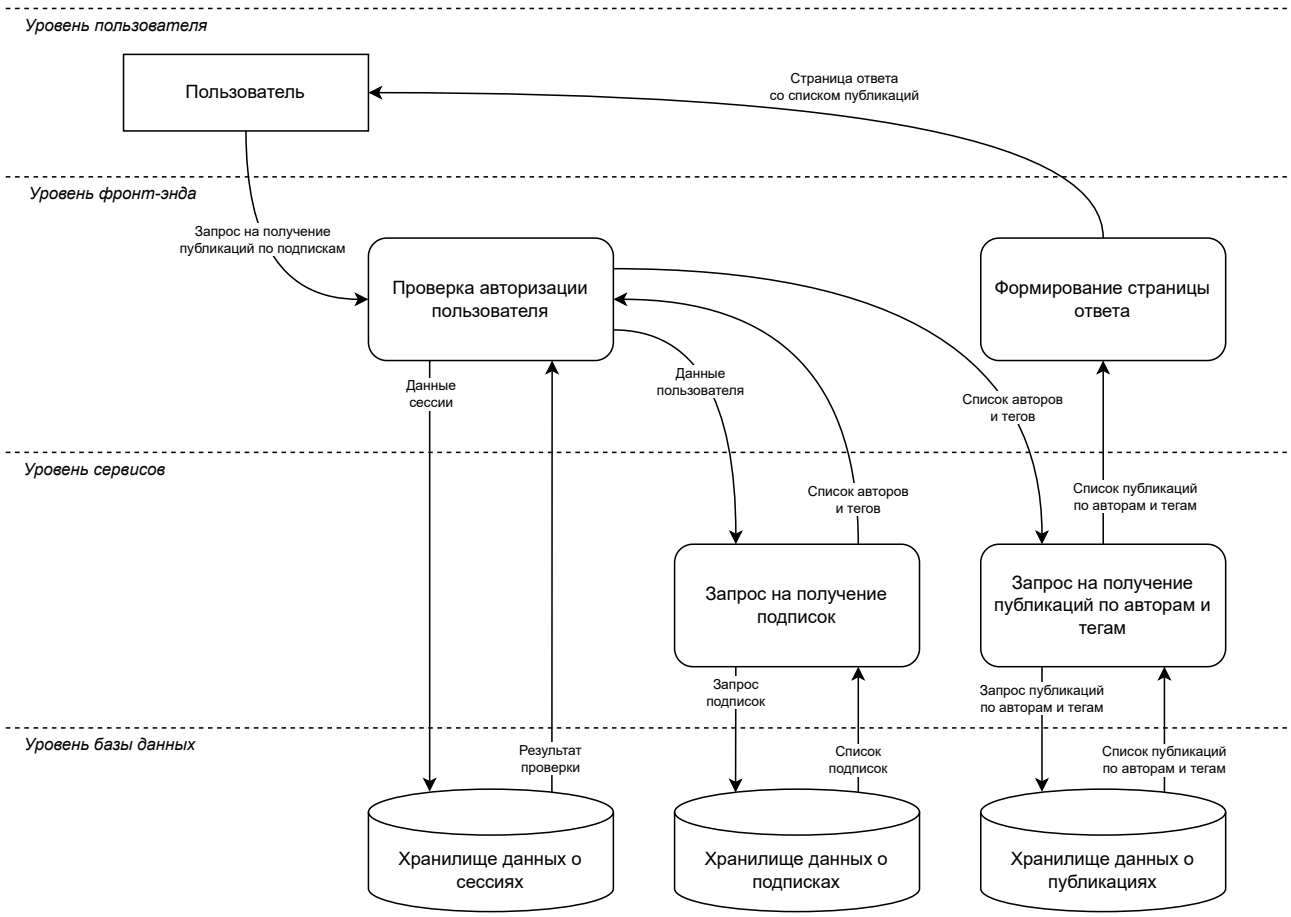

Рисунок 10 — Диаграмма потоков данных при запросе пользователем публикаций на основе его подписок

Архитектура системы

Основополагающей идеей построения программной архитектуры является идея снижения сложности системы путём абстракции и разграничения полномочий. В данном проекте каждая функциональная область реализована посредством собственного микросервиса. Этот подход позволяет бороться со сложностью современных систем. Архитектура системы призвана показать способ развёртывания системы во внешних средах. На рисунке 11 представлена архитектура системы, которая показывает размещение элементов системы на физических носителях и способах их взаимодействия, то есть, указаны протоколы, по которым происходит информационный обмен.

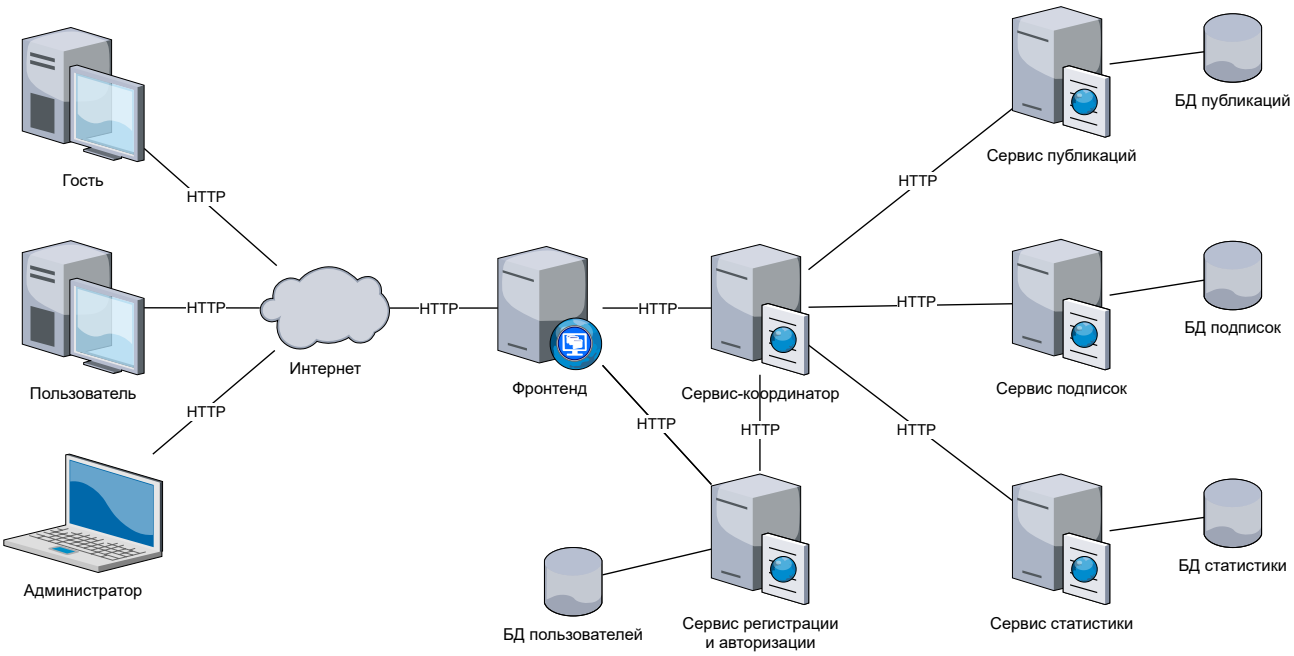

Рисунок 11 — Архитектура разрабатываемой системы

ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

Выбор языка и архитектурного стиля взаимодействия компонентов системы

Для реализации системы использовалась сервис-ориентированная архитектура (СОА). Архитектурным стилем взаимодействия между сервисами выбран REST (Representational State Transfer — «передача состояния представления»). В качестве языка реализации серверной части приложения выбран Python, так как он является одним из ведущих языков в разработке информационных систем на сегодняшний день, а также обладает всеми необходимыми особенностями для реализации СОА и REST.

Для системы выбран фреймворк Django. Он обладает возможностью запуска на разных платформах, использует только необходимые библиотеки и быстро развивается.

Фреймворк использует модель MVC (Model-View-Controller — «Модель-Представление-Контроллер»). Он содержит три основных компонента:

— Модель (model): описывает используемые в приложении данные, а также логику, которая связана непосредственно с данными. Как правило, объекты моделей хранятся в базе данных. Модель не должна содержать логику взаимодействия с пользователем и не должна определять механизм обработки запроса. Кроме того, модель не должна содержать логику отображения данных в представлении.

— Представление (view): отвечает за визуальную часть или пользовательский интерфейс, зачастую (как и в этой работе) это html-страница, через которую пользователь взаимодействует с приложением. Также представление может содержать логику, связанную с отображением данных. Представление не должно содержать логику обработки запроса пользователя или управления данными.

— Контроллер (controller): представляет центральный компонент MVC, который обеспечивает связь между пользователем и приложением, представлением и моделью. Контроллер содержит логику обработки запроса

пользователя. Контроллер получает вводимые пользователем данные и обрабатывает их. В зависимости от результатов обработки, отправляет пользователю определённый вывод, например, в виде представления, наполненного данными моделей.

Данная архитектура позволяет чётко разделять ответственность между компонентами, отделять бизнес-логику от её визуализации. Например, для добавления поддержки мобильных устройств достаточно добавить только соответствующее представление, не меняя модель и контроллер.

Выбор СУБД

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных. Так как, согласно ТЗ, в базе данных не требуется хранить сложные объекты, например, файлы, для проекта подходит реляционная база данных. Поэтому для хранения данных была выбрана СУБД Postgre SQL. PostgreSQL — реляционная система управления базами данных. Она является не-коммерческим ПО с открытым исходным кодом. Для работы с этой СУБД существуют библиотеки для таких распространённых языков программирования как Python, Ruby, Perl, PHP, C, C++, Java, C#, Go. Она работает под управлением многих операционных систем: Linux, MacOS, Windows, FreeBSD, Solaris и многих других. PostgreSQL поддерживает распределенные транзакции, что позволяет использовать его в проекте для хранения финансовых данных. По сравнению с MySQL система PostgreSQL лучше работает с репликацией, так как в ней существует WAL журнал (средство восстановления системы в случае сбоя) физической модификации страниц. PostgreSQL осуществляет асинхронную репликацию типа «ведущий – ведомый», также возможна репликация формата «мастер-мастер».

6 Обеспечение масштабируемости

Как было отмечено выше, СОА позволяет масштабировать систему горизонтально с использованием сервисов-балансировщиков. Совместное использование СОА и REST подходов, предоставляет возможность с лёгкостью добавлять и удалять сервисы, динамически распределяя нагрузку между существующими. Данную функциональность поддерживает, например, веб-сервер nginx.