МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

БАКАЛАВРА

на тему

МИКРОСЕРВИС УЧЕТА ЗАГРУЗКИ РАБОТНИКОВ ПРЕДПРИЯТИЯ

Студент _____ 31.05.2022 г. Горынина А.В.
(подпись, дата)

Руководитель _____ 31.05.2022 г. Синев М.П.
(подпись, дата)

Нормоконтролёр _____ 31.05.2022 г. Бычков А.С.
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 02.06.2022 г. № 13)

Заведующий кафедрой _____ М.А. Митрохин
(подпись)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 07.06.2022 № 5)

Секретарь ГЭК _____ Е.И. Гурин
(подпись)

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«26» апреля 2022 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛВАРА

1. Студент Горынина Анастасия Валерьевна, гр.18ВВ2 факультета вычислительной техники.

2.Тема работы: «Микросервис учета загрузки работников предприятия»

Тема утверждена приказом ПГУ №357/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы к.т.н., доцент каф. ВТ ПГУ Синев М.П.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать микросервис, реализующий функционал агрегирования и хранения данных о занятости сотрудников предприятия. Микросервис должен обеспечивать работу с базой данных. Система должна отвечать требованиям спецификации REST.

5. Перечень подлежащих разработке вопросов
Требуется произвести анализ предметной области, языки программирования в области разработки back-end части веб-приложений, проанализировать преимущества и недостатки существующих баз данных и спроектировать собственную базу данных, выбрать среду разработки, изучить сетевое взаимодействие по протоколу HTTP, проанализировать спецификацию REST. В рамках обеспечения безопасности жизнедеятельности необходимо идентифицировать травмирующие и вредные факторы, возникновение которых потенциально возможно при работе с ЭВМ. Нужно определить риск возникновения травмоопасного воздействия и снизить его до допустимого уровня. Необходимо привести расчет общего искусственного освещения помещения, где используются ПЭВМ.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы, %	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	18	26.04.2022-02.04.2022	
2.	Выбор стека технологий	10	02.05.2022-05.05.2022	
3.	Разработка приложения	24	05.05.2022-14.05.2022	
4.	Тестирование и отладка	24	14.05.2022-23.05.2022	
5.	Анализ условий работы с точки зрения безопасности жизнедеятельности	9	23.05.2022-26.05.2022	
6.	Оформление пояснительной записки	12	26.05.2022-30.05.2022	
7.	Нормоконтроль	3	31.05.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы _____ 26.04.2022 г. Синев М.П.
(подпись, дата)

Задание к исполнению принял студент _____ 26.04.2022 г. Горынина А.В.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 06.06.2022 г. Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Бакалаврская работа содержит 153 листа, 23 рисунка, 13 таблиц, 20 источников, 6 приложений.

БАЗА ДАННЫХ, МИКРОСЕРВИС, СУБД, JAVA, POSTGRESQL, REST, SPRING FRAMEWORK.

Цель работы – разработать микросервис учета загрузки работников предприятия.

Объектом разработки является приложение, обеспечивающее выполнение следующих функций: создание, удаление, изменение, хранение пользователей, контрактов, активностей. Хранение информации о пользователях, контрактах, активностях должно осуществляться в базе данных. Также необходимо реализовать *API (Application Programming Interface)* для взаимодействия с остальными микросервисами.

В результате проведенной работы было разработано приложение, удовлетворяющее поставленной задаче, а именно: разработана структура БД, создан *REST API*, приложение реализует необходимый функционал. Приложение успешно протестировано и готово к использованию.

Содержание

Введение 5
1 Микросервис учета загрузки работников предприятия 6
1.1 Анализ предметной области 6
1.2 Выбор стека технологий 12
1.2.1 Описание микросервисной архитектуры 13
1.2.2 Выбор типа СУБД 14
1.2.3 Анализ РСУБД 16
1.2.4 Анализ языков программирования 20
1.2.5 Анализ фреймворков 23
1.2.6 Анализ сред разработки 27
1.2.7 Описание архитектуры *REST API* 30
1.3 Реализация 32
1.3.1 Структура базы данных 33
1.3.2 Описание *REST API* 38
1.3.3 Описание классов 40
1.3.4 Программная реализация 42
1.4 Описание результата 43
2 Безопасность жизнедеятельности 54
2.1 Классификация и категорирование помещения, где производится работа 54
2.2 Санитарно-гигиеническая характеристика производства 56
2.3 Электробезопасность 59
2.4 Пожарная безопасность 60
2.5 Расчет общего искусственного освещения помещения, где работают на ПЭВМ 62
Заключение 65
Список использованных источников 66
Приложение А – *UML*-диаграммы 68

Приложение Б – Модели базы данных 72
Приложение В – *SQL*-файл базы данных 75
Приложение Г – Описание *REST API* для работы с активностями 83
Приложение Д – Листинг программы 107
Приложение Е – Презентация 143

Введение

В современных условиях для обеспечения жизнеспособности предприятия необходимо рациональное использование всех доступных ресурсов предприятия. Данная проблема частично решаема в рамках приложений для учета загрузки работников предприятия.

Таким образом, актуальность выбранной темы обусловлена потребностью в распределении и организации времени сотрудников для более эффективной работы офиса в целом.

Объект исследования – микросервис учета загрузки работников предприятия.

Цель работы – разработка модульного программного обеспечения, позволяющего отслеживать временную занятость сотрудников офиса.

1 Микросервис учета загрузки работников предприятия

1.1 Анализ предметной области

Сегодня существует множество предприятий самых разных направлений, однако их объединяет необходимость получать прибыль в условиях конкуренции.

Рынок заставляет ускоряться: автоматизировать процессы, планировать гибче, использовать ресурсы максимально эффективно – как трудовые, так и материальные; анализировать каждый этап создания продукта и строить планы развития производства, учитывая множество факторов.

В данный момент работа на усредненном предприятии построена так – подписывается контракт, некий руководитель назначает задачи на проекте, их приоритет и исполнителей. Одной из проблем производства в такой схеме долгое время остается работа с кадрами: отсутствие дисциплины, низкий уровень исполнительности и ответственности ввиду недостатка контроля, или же наоборот, выгорание сотрудников из-за чрезмерной нагрузки.

Эти проблемы решает табель учета рабочего времени. Идея, конечно, не нова – изначально отработанные часы сотрудников отмечались табельщиком или вахтером в бумажных табелях. По окончании месяца табель сдавался в расчетный отдел, где на основании указанных в нем данных сотрудникам начислялась зарплата.

В настоящее время бумажные табели используются все реже. Вместе с тем, основная парадигма сохранена и для цифровых табелей – программ тайм-трекеров. Чаще всего встречаются два типа табелей: классические, табличные табели, которые полностью повторяют организацию традиционного бумажного и представляют собой таблицу со строками-задачами и колонками-днями, и журнальные или хронологические табели, что похожи на журнал событий и представляют собой ленту задач-строк.

Стандартный табель оптимален, если у пользователя на период, заявленный в табеле, в работе не более 10-15 задач. В таком случае ему лишь один раз нужно выбрать задачу, над которой он работает, а далее по дням указывать трудозатраты.

Журнальный подвид востребован, когда уже не хватает стандартного табеля. Например, сотрудник выполняет более 100 разнообразных задач за неделю, при этом каждую задачу нужно учесть отдельно, поскольку этого требует система автоматической генерации счета клиенту с детализацией работ. В таком случае проще каждую работу заводить отдельной строкой, что и позволяет сделать журнальный табель учета времени.

Отслеживание рабочего времени может снизить расходы предприятия несколькими способами.

Расчет показателя утилизации персонала позволяет более эффективно начислять заработную плату. За счет повышения доли оплачиваемого или признаваемого полезным времени сотрудников работодатель получает снижение издержек и себестоимости конечного продукта. Для руководителей учет рабочего времени также дает сведения о загрузке сотрудников, выполняемых типах работ, что позволяет лучше управлять персоналом, не допуская «выгорания» или простоя.

Использование систем тайм-трекеров на проектах повышает прозрачность расходов с целью обеспечения возможности их снижения, также дает возможность автоматизировать выставление счетов с детализацией работ клиенту.

Учет рабочего времени с формированием отчетности по трудозатратам за период позволяет эффективнее управлять организационной деятельностью: достигать плановые значения себестоимости или же, напротив, быстро корректировать курс развития проекта в случае непредвиденных на этапе

планирования затрат; оперативнее распределять и планировать трудовые ресурсы в зависимости от промежуточных итогов работы.

Важно понимать, что системы учета рабочего времени не предназначены для управления задачами и процессами в смысле постановки задачи, контроля сроков исполнения. Поэтому использование систем управления задачами и учета рабочего времени в связке – это частая практика.

Для выбора направления собственной реализации стоит рассмотреть представленные на рынке системы учета времени.

А2Б – онлайн-система управления бизнесом. Система обеспечивает возможность создания стратегии развития бизнеса, постановки и контроля исполнения задач, управления проектами с контролем затрат. В системе возможен сбор, ранжирование и запуск идей и рационализаторских предложений по улучшению компании, предоставляется корпоративный портал, чат. Также система включает возможности *CRM*-системы: ведение базы клиентов и управление продажами, электронный документооборот. Эффективность бизнеса увеличивается за счет сокращения времени на управление и контроль продаж, проектов, документов, задач и бизнес-процессов компании [1].

Рассматриваемая А2Б позиционируется как система контроля полного цикла производства продукта, однако для некоторых категорий разработок ее функционал может быть избыточным: далеко не всегда необходима полноценная *CRM*-система.

Также возникает неудобство, когда заказчик настаивает на использовании собственной или просто иной системы учета рабочего времени: необходимость заполнять таблицы А2Б все равно остается, поскольку эти данные лягут в основу дальнейшего анализа и автоматического планирования, а значит сотруднику придется заполнять несколько разных систем.

OfficeTime – это кроссплатформенный тайм-трекер в основном для работы в офлайне, поэтому ПО хорошо подходит для проектов, где важна секретность. Имеет приложения для работы под ОС *macOS, Windows* и *iOS*. Во время работы отображается таймер, можно категоризировать отслеживаемое время по клиенту или проекту. Функция отслеживания простоев дает возможность определить время, в течение которого сотрудник не работал. К сессиям можно добавлять дополнительные данные, включая разновидность выполняемой работы и почасовую ставку. Если время, потраченное на определенную задачу, не отслеживалось, то задачу можно добавить вручную. В *OfficeTime* присутствует функция отслеживания расходов, составления отчетов и базовых счетов [2].

Сейчас многие компании придерживаются политики «меньше программного обеспечения, которое нужно дополнительно устанавливать и конфигурировать – меньше ответственности компании в плане технического обеспечения сотрудника». Поэтому во многом предпочтение отдается веб-сервисам и облачным технологиям.

Тайм-трекер *Harvest* больше подходит для удаленных команд. В программе реализована функция отслеживания расходов и некоторые другие бухгалтерские функции: подсчет затрат на клиентские проекты, управление счетами и платёжными извещениями, запись и удаление платежей, просмотр истории счетов, уведомления для платежей, сканирование чеков об оплате. Трекер использует таймер, но поддерживает и ручной ввод часов, реализует просмотр прогресса товарищей по команде в режиме реального времени, для улучшения отслеживания задач введена возможность их расширенного описания. Приложение работает под управлением операционных систем *macOS, Windows, iOS, Android*, имеет множество интеграций со сторонними инструментами, включая управление проектами, создание счетов, *CRM*, клиентская поддержка, инструменты разработчика [3].

Из-за обилия бухгалтерских функций тайм-трекер *Harvest* больше подходит для небольших независимых команд, но не для разработки на предприятии. Однако и для маленьких команд это программное обеспечение не самый лучший выбор: интерфейс конфигурирования отчетов в сравнении с аналогичными программами недостаточно гибок, создание нового проекта неоправданно усложнено с точки зрения интерфейса.

Система учета рабочего времени *Kickidler* – это расширенная версия тайм-трекера, способная выполнять задачи по контролю сотрудников. Программа включает дополнительно такие инструменты как: отчеты по рабочему времени, анализ эффективности сотрудников, динамика продуктивности сотрудников, кейлоггер. Функция «Контроль нарушений» позволяет фиксировать любые нарушения рабочего распорядка сотрудниками. Нарушения можно увидеть, как в реальном времени, используя онлайн-мониторинг компьютеров, так и в истории, используя отчеты по рабочему времени и записи с экранов мониторов: *Kickidler* фиксирует открытие нежелательных сайтов, запуск нежелательных программ или приложений, открытие нежелательных файлов. Функционал программы включает автоматические уведомления сотрудникам о нарушениях на рабочем месте, а также интерфейс, дающий возможность сотрудникам видеть аналитику по собственной продуктивности [4].

Рассматриваемое программное обеспечение позиционируется на рынке как повышающее мотивацию и дисциплину коллектива в целях поддержки постоянно высокой продуктивности команд. Однако же на практике подобные вмешательства в работу сотрудника скорее нервируют, отвлекают от основной деятельности.

Подводя итоги, вышеперечисленные системы тайм-трекинга довольно сильно нагружены дополнительным функционалом, не имеющим веса в рамках предприятия АО «Ай-теко ВЦ», также в рассмотренных аналогах учет

времени вызывает у сотрудника по большей части отрицательный опыт, в связи с чем от данных систем постепенно отказываются.

Собственная система предприятия АО «Ай-теко ВЦ», достаточно проста чтобы быть комфортной пользователю, однако реализует необходимый предприятию функционал: учет рабочего времени, формирование отчетности о затраченных на проект временных ресурсах. Система состоит из нескольких взаимодействующих микросервисов и *front-end* приложения. Структурная схема системы приведена на рисунке 1.

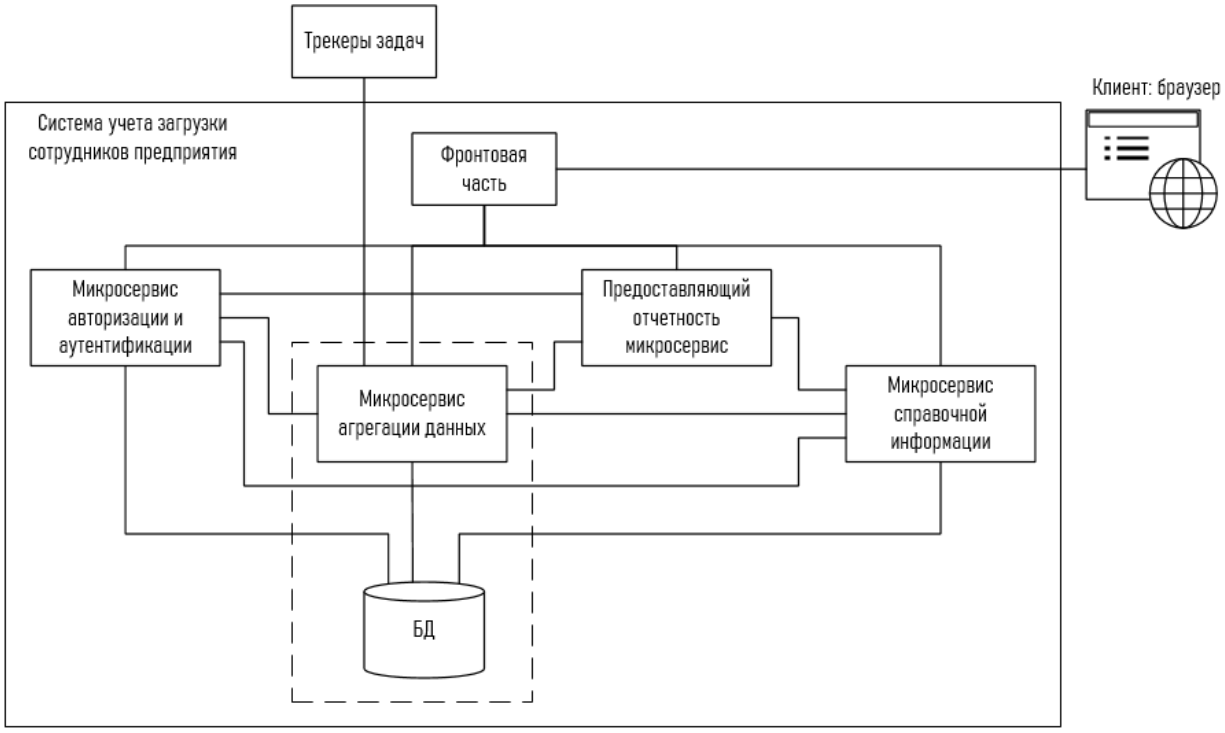


Рисунок 1 – Структурная схема системы

Предприятие ведет заказную разработку и в случае, когда заказчик настаивает на использовании собственной системы отслеживания состояния задач проекта, задачи различных проектов оказываются подконтрольны разным системам. В таких условиях итоговая загрузка сотрудника становится неочевидна. Актуальность разработки для предприятия состоит в централизованном хранении загрузки сотрудников.

Разрабатываемое в рамках данной работы приложение должно быть реализовано как отдельный независимый микросервис, реализующий функционал заполнения занятости и впоследствии включенный в основную систему компании. Микросервис должен обеспечивать работу с базой данных, которая, в свою очередь, должна быть спроектирована с учетом потребностей: должен быть предусмотрен механизм хранения большого количества записей и реализован алгоритм быстрой выборки записей из базы для дальнейшего построения отчетов в основной системе. Реализуемая в данной работе часть представлена на диаграмме А.1 блоками, окруженными пунктирной линией. Интеграция с часто используемыми трекерами задач: *Jira*, *Redmine, YouTrack* – в данной работе не производится, однако предусматривается в дальнейшем.

Для разработки ставятся задачи: сравнить часто используемые языки программирования в области разработки *back-end* части веб-приложений, проанализировать преимущества и недостатки существующих баз данных и спроектировать собственную базу данных, изучить сетевое взаимодействие по протоколу *HTTP (Hypertext Transfer Protocol),* взаимодействие между клиент-сервером по текстовому формату обмена данными *JSON (JavaScript Object Notation),* проанализировать спецификацию *REST (Representational State Transfer).*

1.2 Выбор стека технологий

Архитектура приложения ясна – постановка задачи подразумевает микросервисную архитектуру разрабатываемого приложения, однако от выбора стека технологий – языка программирования, фреймворка, среды разработки, системы управления базами данных – напрямую зависит время, затрачиваемое на создание минимально жизнеспособного продукта, время полного цикла разработки, удобство проектирования базы данных, написания кода, тестирования и отладки, поддержки приложения.

1.2.1 Описание микросервисной архитектуры

Микросервисная архитектура – это вариант сервис-ориентированной архитектуры программного обеспечения, направленный на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей [5]. Каждый микросервис может независимо развиваться и масштабироваться, модуль работает со собственными ресурсами. Отдельные микросервисы могут располагаться на разных серверных узлах. Сравнение монолитной и микросервисной архитектур представлено на рисунке 2.

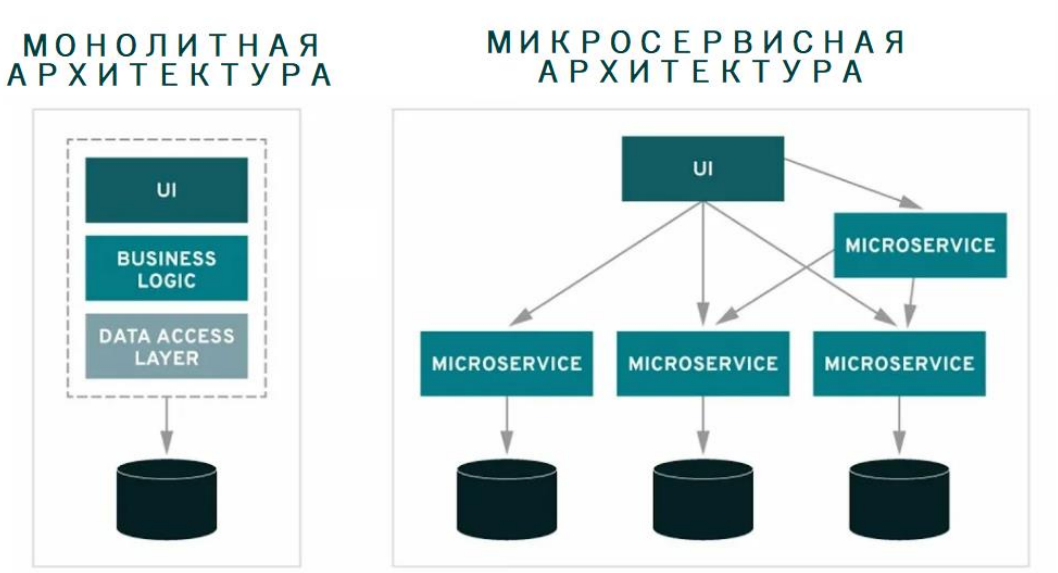


Рисунок 2 – Сравнение монолитной и микросервисной архитектур

Преимущества использования микросервисов:

– независимость развёртывания, обновления и масштабирования каждого из микросервисов;

– при сбое одного микросервиса, приложение не сбоит целиком;

– микросервисная архитектура позволяет размещать разные модули на разных серверах;

– модули могут быть реализованы с использованием различных языков программирования, фреймворков, связующего программного обеспечения, выполняться в различных средах контейнеризации,

виртуализации, под управлением различных операционных систем на различных аппаратных платформах;

– децентрализованное управление;

– децентрализованное управление данными.

Как и у всякой технологии и подхода к разработке у микросервисов есть и ряд недостатков:

– сложность разработки выше в сравнении с приложением монолитной архитектуры;

– сложность управления, модули используют *API* и физически располагаются на разных серверных узлах;

– необходимо четкое разграничение функционала модулей, микросервис по возможности выполняет только одну достаточно элементарную функцию;

– увеличение сетевых задержек.

Разрабатываемое в рамках данной работы приложение реализуется как отдельный независимый микросервис, реализующий функционал заполнения занятости и впоследствии включенный в основную систему компании.

1.2.2 Выбор типа СУБД

Сегодня существует два типа систем управления базами данных (СУБД): реляционные, где используется язык *SQL (structured query language)* для запросов, и нереляционные (*NoSQL*). Для выбора вида СУБД, подходящего в данной разработке, нужно выделить их характерные отличия друг от друга.

Разница в структуре и типе хранимых данных. В реляционных базах данных (БД) должна быть однозначно определена структура хранения данных, в то время как в *NoSQL* БД нет таких ограничений: допускается хранение любого типа данных. Для микросервиса подходит структура данных с однозначно определенными связями между таблицами.

Хранение и доступ к сложным структурам данных. В этом вопросе реляционные базы данных выигрывают в производительности по сравнению с *NoSQL* БД, так как принципы построения реляционных БД предполагают работу со сложными структурами данных. Для микросервиса больше подходит реляционная БД.

Различия в структуре запросов. В реляционных БД данные могут быть получены только посредством языка *SQL*, так как реляционные БД строятся по общим принципам. В нереляционных СУБД нет четкого определения, как должен выглядеть запрос и поэтому для каждой *NoSQL* БД реализуется свой способ работы с данными. Для микросервиса требуется построение множества запросов с выборками по условию, и использование нереляционной СУБД будет сопряжено с проблемами с созданием запросов, поэтому выбор реляционной СУБД здесь очевиден.

Масштабируемость. Оба вида СУБД хорошо масштабируются по вертикали, однако по горизонтали лучше масштабируются нереляционные СУБД. Для микросервиса больше подходит вариант с лучшим горизонтальным масштабированием.

Надежность. Реляционные базы данных являются более надежными в сравнении с *NoSQL*. Поэтому реляционные БД более предпочтительны для выбора в качестве решения для микросервиса.

Поиск информации. *NoSQL* базы данных появились заметно позже реляционных, отчасти поэтому интернет-сообщество более обширно у реляционного типа СУБД, а значит и найти готовые решения для возникающих проблем не составит труда. Однако сказывается еще и то, что реляционные БД строятся по общим принципам, а нереляционные БД часто предлагают уникальные решения в плане организации хранения. В этом вопросе выбор в пользу реляционных БД.

Анализ возможностей реляционных и нереляционных СУБД показал, что для разработки микросервиса учета загрузки сотрудников больше подходит реляционная СУБД (РСУБД).

1.2.3 Анализ РСУБД

Для выбора наиболее подходящей реляционной СУБД необходимо проанализировать представленные на рынке реляционные СУБД. Для рассмотрения были выбраны 4 популярных РСУБД: *Oracle Database, MySQL, PostgreSQL, MS SQL Server*.

1.2.3.1 ОРСУБД *Oracle Database* – объектно-реляционная система управления базами данных, разработанная компанией *Oracle*. В 2013 году была представлена версия 12c (суффикс «*c*» обозначает «*cloud*» – облако), основное новшество которой – поддержка подключаемых баз данных (*pluggable database*), обеспечивающая свойства мультиарендности и живой миграции баз данных. Эта версия поддерживает подключение баз данных через облачные сервисы, что позволяет размещать базу данных на нескольких серверах и хранить огромный объем данных. Сегодня доступна версия 21c, она включает более 200 инноваций, в том числе поддержку неизменяемых блокчейн-таблиц, встроенную в СУБД поддержку языка программирования *JavaScript*, поддержку типа данных *JSON* в бинарном формате, который встроен в БД, автоматизированное машинное обучение (*AutoML*) для машинного обучения в базе данных и поддержку энергонезависимой памяти, а также усовершенствования в технологии «*in-Memory*», производительности обработки геоинформации, шардинге, технологии подключаемых баз данных и безопасности [6].

Достоинства ОРСУБД *Oracle Database*:

– надежность;
– хорошая документация;
– функционал.

Недостатки ОРСУБД *Oracle Database*:

– высокая стоимость продукта для юридических лиц;

– для системы необходимы аппаратные ресурсы, обладающие достаточной мощностью и производительностью.

К сожалению, высокая стоимость, $118.40 за хранилище в 1 ТБ в месяц, не позволяет использовать данную ОРСУБД, бесплатная версия позволяет развертывать относительно небольшие по объему базы данных.

1.2.3.2 СУБД *MySQL* – свободная реляционная система управления базами данных. Разработку и поддержку *MySQL* осуществляет корпорация *Oracle*. *MySQL* имеет двойное лицензирование. СУБД может распространяться в соответствии с условиями лицензии *GNU GPL* (универсальная общедоступная лицензия *GNU*), однако по условиям *GPL*, если какая-либо программа использует библиотеки (или включает в себя другой *GPL*-код) *MySQL*, то она тоже должна распространяться по лицензии *GPL*. Это может расходиться с планами разработчиков, не желающих открывать исходные коды своих программ. Для таких случаев предусмотрена коммерческая лицензия ценой в $2000 (*Standart edition*) и $5000 (*Enterprise edition*) в год за один сервер. Разработчики создают функциональность по заказу лицензионных пользователей – именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации. Гибкость СУБД *MySQL* обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа *MyISAM*, поддерживающие полнотекстовый поиск, так и таблицы *InnoDB*, поддерживающие транзакции на уровне отдельных записей. Благодаря открытой архитектуре и *GPL*-лицензированию, в СУБД постоянно появляются новые типы таблиц.

Достоинства СУБД *MySQL*:

– бесплатная при условии последующего распространения кода в соответствии с условиями лицензии *GNU GPL*;

– хорошая документация;

– безопасность;

– достойный функционал бесплатной версии;

– возможность взаимодействия с другими базами данных, к примеру, *DB2* и *Oracle*.

Недостатки СУБД *MySQL*:

– проблемы с надежностью;

– для использования без необходимости соблюдения условий лицензии *GNU GPL* необходима платная версия;

– некоторые ограничения функционала бесплатной версии;

– нет встроенной поддержки *XML* и *OLAP*.

По итогу рассматриваемая система используется в основном для мелких и средних проектов и не подходит для крупных.

1.2.3.3 СУБД *Microsoft SQL Server* – СУБД, разработанная корпорацией *Microsoft*. Основным языком для написания запросов является *Transact-SQL*. Данная РСУБД используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия. Последняя версия продукта – *SQL Server* 2019. В этой версии добавились инструменты для анализа данных с помощью встроенной платформы *Apache Spark*, при этом есть возможность выполнять запросы по реляционным и нереляционным базам данных, а также структурированным и неструктурированным данным. При использовании СУБД *Microsoft SQL Server* активно используется интеграция с продуктами *Microsoft*, например, с платформой *Visual Studio* [7].

Достоинства СУБД *MS SQL Server*:

– высокая производительность, отказоустойчивость;

– поддержка многопользовательской среды;

– расширенные функции резервирования данных;

– работа с удаленным подключением;

– возможность регулировки и отслеживания уровня производительности в целях уменьшения загрузки;

– возможность визуализации на мобильных устройствах.

Недостатки СУБД *MS SQL Server*:

– очень высокая стоимость продукта для юридических лиц;

– возможны проблемы в работе служб интеграции импорта файлов;

– высокая ресурсоемкость.

При всех упомянутых достоинствах, у данной РСУБД есть очень существенный недостаток, который делает ее недоступной для использования – ее высокая цена для юридических лиц, а именно $13748 за копию. Бесплатная версия не обладает достаточными возможностями для полной реализации базы данных микросервиса.

1.2.3.4 ОРСУБД *PostgreSQL* – свободная объектно-реляционная система управления базами данных с открытым исходным кодом, разработчиком которой является сообщество *PostgreSQL*. В СУБД помимо хранимых процедур (набора *SQL*-инструкций, который компилируется один раз и хранится на сервере) есть возможность написания блоков кода, исполняемых на сервере – функций. Функции могут быть написаны с использованием одного из следующих языков: встроенный процедурный язык *pgSQL*, скриптовые языки — *Lua*, *Perl*, *PHP*, *Python*, *Ruby*, *sh*, *Tcl*, *Scheme*, *v8* (*Javascript*); классические языки — *C*, *C++*, *Java* (через модуль *Java*); статистический язык *R* (через модуль *R*). *PostgreSQL* допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса [8].

Достоинства ОРСУБД *PostgreSQL*:

– высокопроизводительные и надёжные механизмы транзакций и репликации;

– расширяемая система встроенных языков программирования;

– встроенная поддержка слабоструктурированных данных в формате *JSON* с возможностью их индексации;

– расширяемость – возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных;

– бесплатная для личного и коммерческого использования.

Недостатки ОРСУБД *PostgreSQL*:

– трудности с обновлением.

1.2.3.5 Анализ сильных и слабых сторон ОРСУБД PostgreSQL показал, что ОРСУБД реализует весь необходимый для разработки микросервиса функционал, соответствует требованиям к надежности ПО. Аргументами в пользу выбора PostgreSQL стало и то, что продукт бесплатно распространяется, имеет большое сообщество. Версия *PostgreSQL* – 11.

1.2.4 Анализ языков программирования

При реализации любого программного продукта одним из самых важных этапов является выбор языка программирования – от выбора напрямую зависит скорость разработки. В настоящее время наиболее популярными языками программирования для разработки веб-приложений являются: *Go, Python, JavaScript, Java*. Такие языки как *C++, PHP* и *Ruby* в последнее время теряют популярность в веб-разработке.

1.2.4.1 Язык программирования *Go* (также известный как *Golang*) — компилируемый многопоточный язык программирования, разработанный компанией *Google*. Язык разрабатывался для проектирования высокопроизводительных программ и для решения проблем большинства языков. *Golang* проектировался с упором на решение проблем медленной сборки, небезопасного и ресурсоемкого параллелизма программ, неконтролируемых зависимостей, проблемы межъязыкового взаимодействия. В языке есть все необходимое для веб-разработки и не только – количество

сторонних библиотек постоянно растет. Кроме того, есть возможность использовать библиотеки *C* и *C++*. Язык имеет сильное сообщество и постоянно развивается.

Язык не предоставляет целый ряд средств, доступных в других популярных языках, что объясняется сознательным выбором разработчиков, считающих, что такие средства либо затрудняют эффективную компиляцию, либо провоцируют программиста на создание неэффективного кода [9]. В реализацию языка не попали средства обобщённого программирования (*generics*), алгебраические типы данных, перегрузка функций, переопределение операторов, отсутствуют перечислимые типов в привычном понимании. Также данный язык не является объектно-ориентированным в полном смысле этого понятия, так как в язык сознательно не была включена концепция наследования – наследование заменяется механизмом встраивания. Существуют интерфейсы, которые не нужно явно имплементировать, достаточно реализовать методы интерфейса. Однако решение оказалось спорным: отсутствие явного указания на реализацию интерфейса типом затрудняет понимание кода, его модификацию и рефакторинг, а компилятор не может автоматически проверить тип на соответствие реализуемым интерфейсам.

1.2.4.2 Язык программирования *Python* – высокоуровневый язык программирования, который ориентирован на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык гибок и обладает широким функционалом, а код высокой степенью читаемости. Сильными сторонами *Python* являются его модульность и возможность интегрироваться с другими языками программирования, в том числе в составе сложных комплексных приложений и систем [10].

Однако у данного языка есть существенный недостаток – он обладает низким быстродействием и поэтому чаще используется как язык второго плана.

1.2.4.3 Язык программирования *JavaScript* – мультипарадигменный язык программирования, не принадлежащий какой-либо организации или компании. Наиболее часто используется в качестве языка сценариев веб-страниц, однако может быть использован для разработки *back-end* части веб-приложений. *JavaScript* поддерживает объектно-ориентированный и функциональный стили программирования. Так как язык является интерпретируемым, то производительность и безопасность зависит от конкретной реализации интерпретатора [11].

1.2.4.4 Язык программирования *Java* – строго типизированный объектно-ориентированный язык программирования общего назначения. В настоящее время разработкой занимается компания *Oracle*. Основной особенностью данного языка является то, что программы на *Java* транслируются в байт-код, который выполняется на виртуальной машине *Java*. Виртуальная машина обрабатывает байтовый код и передает инструкции оборудованию как интерпретатор, что обеспечивает переносимость программы на платформы, для которых существует реализация *Java*-машины. Другой важной особенностью технологии *Java* является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной [12]. Любые операции, которые превышают установленные полномочия программы, вызывают немедленное прерывание.

1.2.4.5 В качестве языка программирования был выбран язык *Java*. В отличие от программ, написанных на *Python,* программы на *Java* обладают большей производительностью, по сравнению с языком *Golang Java* давно используется программистами, имеет большое сообщество, хорошую

документацию, множество фреймворков и готовых решений. *Golang* не был выбран еще и потому, что многие решения создателей языка пока являются довольно спорными. *JavaScript* проигрывает *Java* в безопасности. Для разработки выбрана *Java* 11.

1.2.5 Анализ фреймворков

Фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. Разработку с использованием фреймворка можно назвать каркасным подходом к построению программ, где любая конфигурация программы строится из двух частей: постоянная часть – каркас, не меняющийся от конфигурации к конфигурации и несущий в себе гнёзда, в которых размещается вторая, переменная часть, сменные модули – точки расширения.

С учетом выбранного на предыдущем этапе языка программирования для рассмотрения были подобраны 3 фреймворка – *Spring Framework, Blade Framework, Dropwizard Framework.*

1.2.5.1 Фреймворк *Spring* является модульным фреймворком, что позволяет использовать только те части, которые нужны для решения конкретной задачи, без необходимости вводить остальные. Ядро каркаса *Spring Framework* основано на принципе инверсии управления (*Inversion of Control - IoC*) [13], когда создание зависимостей между компонентами и управление ими осуществляется внешним образом. В объектно-ориентрованном подходе под зависимостью понимают класс, без которого не может функционировать код другого класса.

В контексте внедрения зависимостей *Spring* действует в большей степени подобно контейнеру, чем каркасу, предоставляя экземпляры классов приложения со всеми необходимыми зависимостями, но делает это гораздо менее навязчиво. Применение каркаса *Spring* для внедрения зависимостей

опирается на соблюдение в его классах соглашений об именовании, принятых для компонентов *JavaBeans.*

Преимущества использования внедрения зависимостей:

– возможность значительного сокращения объема кода, который должен быть написан для связывания вместе компонентов приложения;

– упрощенное конфигурирование приложений;

– возможность управлять общими зависимостями в одном хранилище;

– улучшенная тестируемость – когда классы проектируются для внедрения зависимостей, становится возможной простая замена зависимостей;

– стимулирование качественных проектных решений для приложений.

Внедрение зависимостей дает немало преимуществ, хотя оно и не лишено недостатков. В частности, внедрение зависимостей способно затруднить тем, кто не особенно хорошо ориентируется в прикладном коде, возможность выяснить, какая именно реализация отдельной зависимости привязана к конкретным объектам. Но, как правило, подобное затруднение возникает только из-за отсутствия опыта внедрения зависимостей.

Spring Boot является расширением *Spring*, используемым для создания приложений на основе микросервисов. *Spring Boot* самостоятельно определяет набор настроенных *bean*-компонентов по умолчанию, которые разработчик может переопределить при необходимости. Более того, фреймворк выбирает пакеты для установки с учетом необходимых зависимостей. *Spring Boot* позволяет разработчикам настраивать и запускать автономные приложения *Spring* производственного уровня без их развертывания на веб-сервере. Встроенный сервер – часть приложения. Это означает, что не нужно предварительно устанавливать его в среде развертывания. Таким образом, разработчики *Spring Boot* сразу же начинают создавать свои приложения, уделяя больше внимания бизнес-логике, поскольку большая часть работы выполняется самим фреймворком.

Основные преимущества *Spring Boot*:

– быстрая и легкая разработка приложений на основе *Spring*;

– автоконфигурация;

– встроенные серверы (*Tomcat*, *Jetty* и *Undertow*);

– *HTTP end-points*, позволяющие вводить внутренние функции приложения, такие как показатели, состояние здоровья и другие;

– не требует *XML*-конфигурации, поддерживает как свойства, так и *YAML*;

– множество плагинов, облегчающих работу разработчиков со встроенными базами данных и базами данных в памяти;

– легкий доступ к базам данных и службам очередей;

– полная документация и обширное сообщество;

– интеграция с экосистемой Spring.

Недостатки *Spring Boot*:

– *Spring Boot* создает множество неиспользуемых зависимостей, что приводит к большому размеру файла развертывания;

– сложный и трудоемкий процесс преобразования устаревшего или существующего проекта *Spring* в приложения *Spring Boot*;

– не подходит для масштабных проектов – по мнению многих разработчиков, *Spring Boot* не подходит для создания монолитных приложений.

1.2.5.2 Фреймворк *Blade* – это фреймворк, который является частью более широкой платформы под названием *Lavarel*. Фреймворк используется для создания веб-приложений, позволяя строить *RESTful* веб-сервисы. Механизм *Blade* удобен для пользователя, когда речь идет о доступе к функциям платформы *Lavarel*. Фреймворк имеет базовый *Auth* функционал, нативную возможность формировать *json*-ответы, поддержку большого количества расширений и достаточно высокую производительность [14].

Имеет хорошую документацию, развивающееся сообщество. *Blade* очень легковесен – его исходный код занимает не больше 500 Кб, поэтому его выгодно использовать в маленьких проектах вместо тяжеловесного *Spring*.

1.2.5.3 Фреймворк *Dropwizard* поставляется с базовым пакетом компонентов или библиотек, необходимых для разработки веб-сервисов *RESTful*, поэтому не нужно настраивать каждый из них отдельно. *Dropwizard* имеет готовую поддержку сложной конфигурации, метрик приложений, логгирования, операционных инструментов и многого другого, что позволяет команде поставлять веб-сервис производственного качества в кратчайшие сроки. Имеет некоторую поддержку других типов сервисов, но в первую очередь предназначен для высокопроизводительного уровня *HTTP/REST* [15]. *Dropwizard* использует библиотеку *HTTP Jetty* для внедрения настроенного *HTTP*-сервера непосредственно в проект. Поддерживает конфигурирование свойствами и *YAML*. Фреймворк имеет хорошую документацию, обширное сообщество.

1.2.5.4 Для разработки был выбран фреймворк *Spring*. *Spring* работает скорее как *IoC*-контейнер, не накладывая особых ограничений на структуру приложения – считается, что он менее навязчив, нежели остальные фреймворки. *Spring Boot* принимает довольно самоуверенный взгляд на платформу *Spring* и сторонние библиотеки, поэтому время сборки, развертывания приложения оказывается меньше, чем при разработке с использованием фреймворка *Blade*. *Blade* легковеснее *Spring,* однако в данном случае разрабатывается приложение для внутреннего корпоративного использования с возможностью расширения функционала впоследствии, следовательно, легковесность не так важна при выборе фреймворка.

Dropwizard и *Spring Boot* занимают одну нишу на рынке, однако *Spring* имеет большее количество типов поддерживаемых сервисов – *REST, JMS*, расширенный протокол очереди сообщений, веб-сервисы на основе *SOAP* и

услуги *Contract First*. Только некоторые из этих сервисов возможно реализовать на фрейворке *Dropwizard* – с использованием сторонних бибилиотек. Также аргументом в пользу *Spring* и *Spring Boot* стала встроенная инъекция зависимостей *Spring*. В итоговом продукте используется *Spring* 5. Также используется ставшая де-факто стандартом *ORM Hibernate*.

1.2.6 Анализ сред разработки

Интегрированная среда разработки (англ. *Integrated Development Environment, IDE*) – комплекс программных средств, используемый программистами для разработки программного обеспечения. Среда разработки включает в себя: текстовый редактор, компилятор и/или интерпретатор, средства автоматизации сборки, отладчик. Иногда содержит также средства для интеграции с системами управления версиями и инструменты для упрощения конструирования графического интерфейса пользователя. Выбор среды разработки во многом определяет удобство программиста при написании кода, тестировании и отладке приложения, а значит и скорость создания конечного продукта. С учетом выбора фреймворка и языка программирования, для рассмотрения были подобраны три среды разработки – *Java NetBeans, MyEclipse* или *IntelliJ IDEA*.

1.2.6.1 Среда разработки *Java NetBeans* – бесплатная *IDE*, развиваемая *Oracle* и сообществом *NetBeans,* работает на машинах под управлением ОС *Windows, Mac OS X, Linux* и *Solaris,* пакет *portable* можно запустить на любых системах, для которых существует *Java*-машина.

Редактор *NetBeans* поддерживает языки, обнаруживает ошибки во время печати, имеет всплывающие подсказки и автодополнение кода. *NetBeans* обладает полным спектром инструментов рефакторинга, включая средство автоматизированного рефакторинга *Inspect & Transform. NetBeans* имеет встроенную поддержку *Maven* и *Ant,* а также плагина для *Gradle*. *NetBeans* также содержит привлекательное для разработчика графическое отображение

для зависимостей *Maven*. Встроенный профайлер *NetBeans* позволяет контролировать использование процессора и памяти, и обладает отличными инструментами для поиска утечек памяти. *NetBeans* поддерживает многие фреймаорки, в их числе и *Spring*. Присутствует интеграция с системами контроля версий *CVS* и *Git*, редактор *SQL* и инструменты для работы с базами данных.

1.2.6.2 Среда разработки *IntelliJ IDEA* с точки зрения возможностей и цены поставляется в двух вариантах: бесплатного варианта *Community edition*, и платного *Ultimate edition* с расширенной функциональностью с ценой в $598.80 за пользователя в первый год или же $59.88 ежемесячно. В последующие годы цена для бизнеса падает, для стартапов, фрилансеров она существенно ниже, а для студентов, учителей, *Java*-чемпионов и *open source*-разработчиков она и вовсе бесплатна.

Community edition предназначена для *JVM*- и *Android*-разработки. Бесплатная версия поддерживает *Java*, *Kotlin*, *Groovy* и *Scala*, присутствуют пакетные менеджеры *Maven, Gradle* и *SBT*, есть инструменты для работы с системами контроля версий *Git, SVN, Mercurial* и *CVS*.

Ultimate edition приспособлена для веб- и *enterprise*-разработки. Эта версия *IDE* работает дополнительно с такими системами контроля версий как *Perforce, ClearCase* и *TFS*, есть поддержка *Java EE, Spring, GWT, Vaadin, Play*, *Grails* и ряда других фреймворков. Введены также редактор *SQL* и инструментов для работы с базами данных. *IntelliJ IDEA* также имеет множество бесплатных плагинов, например, для работы с *UML*-диаграммами. Также *IDEA* оснащена средой выполнения тестов, инструментами покрытия (*coverage tools*) и встроенным терминальным окном. *IntelliJ IDEA* не имеет встроенного профайлера, однако для закрытия этой потребности можно установить плагин, к примеру, *YourKit*.

IntelliJ IDEA демонстрирует глубокое понимание кода, предлагая «умное автодополнение». Это означает, что среда разработки показывает список наиболее релевантных символов, применимых в данном контексте. Список символов зависит не только от контекста как такового, «общепринятого», но от стиля программирования конкретного разработчика. Во всех случаях автодополнения, *IDEA* пытается угадать тип символа во время выполнения, уточнить свой выбор и даже применить приведение типов если необходимо.

1.2.6.3 Среда разработки *Eclipse* – полностью бесплатная кроссплатформенная среда разработки: *IDE* работает на *Linux*, *Mac OS X*, *Solaris* и *Windows*. *Eclipse* интегрируется с системами контроля версий CVS и Git, состав расширяется плагинами. Распространенные паттерны кода можно генерировать из шаблонов кода. *Eclipse* поддерживает расширенный набор действий по рефакторингу – 23 операции. *Eclipse* поддерживает отладку как локально, так и удаленно, при условии, что используется *JVM*, которая поддерживает удаленную отладку.

Eclipse работает довольно медленно, поскольку упирается корнями в довольно старые версии *JVM*. Даже сегодня она кажется медлительной, особенно при одновременном использовании множества плагинов. Часть расходов ресурсов *Eclipse* можно отнести на счёт её встроенного инкрементного компилятора, который запускается всякий раз при загрузке файла или обновлении кода – он контролирует ввод текста, предупреждая ошибки.

В настоящее время для *Eclipse* написано более 1700 плагинов, плагины поддерживают более 100 языков программирования и почти 200 фреймворков для разработки приложений, в том числе Spring, также около 30 серверов приложений. У *Eclipse* – самая развернутая экосистема плагинов среди всех *IDE*, а также наибольшая вероятность сломать сборку *IDE* из-за установки

несовместимого набора плагинов. Также присутствует проблема запоздалого обновления документации.

1.2.6.4 Среда разработки *IntelliJ IDEA* подкупает своим глубоким пониманием кода, умной эргономикой, обширным функционалом, множеством плагинов – набор плагинов больше, чем у *NetBeans*, их поведение более предсказуемо нежели у *Eclipse*. Для студентов *IntelliJ IDEA Ultimate* полностью бесплатна, так что выбор остановился именно на этой среде разработки, конкретно *IntelliJ IDEA Ultimate* 2018.3.

1.2.7 Описание архитектуры *REST API*

REST означает *REpresentational State Transfer* – «передача состояния представления». Это архитектурный подход к проектированию взаимодействий компонентов распределенного веб-приложения. Принципы *REST* сформулировал Рой Филдинг в своей диссертации в 2000 году, но не существует официально принятого стандарта или спецификации *REST*. *API*-интерфейс может считаться *RESTful* только в том случае, если соблюдены все требования. Существует шесть ограничений для проектирования *REST*-приложений.

1.2.7.1 Модель клиент-сервер. Система должна быть поделена на клиентов и серверов, в основе данного ограничения лежит принцип разграничения потребностей. Это позволяет упростить серверную часть приложения, что в будущем упрощает ее масштабируемость. При этом у клиентского интерфейса повышается переносимость кода на другие платформы.

1.2.7.2 Кэширование. Клиенты могут выполнять кэширование ответов сервера. В этом случае ответы сервера должны быть обозначены как кэшируемые, если поступила новая информация, или некэшируемые, если поступившие данные уже передавались ранее. Грамотно построенное кэширование способно уменьшить взаимодействие между клиентом и

сервером, что увеличивает производительность и масштабируемость распределенной веб-системы в целом.

1.2.7.3 Отсутствие состояния. Взаимодействие между клиентом и сервером строится по следующему условию: в моменты времени между запросами никакая информация о клиенте не должна храниться на сервере, а каждый запрос от клиента должен быть составлен таким образом, чтобы на сервер поступила вся необходимая информация о транзакции для ее завершения. При этом информация о состоянии сессии сохраняется у клиента. Во время обработки запросов от клиента считается, что клиент находится в переходном состоянии. Отсутствие необходимости хранить информацию о состояниях высвобождает ресурсы сервера, которые могут быть направлены на обработку одновременно большего количества клиентов.

1.2.7.4 Единообразие интерфейса. Для эффективного взаимодействия компонентов и кэширования в сети необходим унифицированный интерфейс, позволяющий независимо развиваться отдельным сервисам.

В свою очередь к унифицированным интерфейсам предъявляются следующие четыре ограничительных требования:

– идентификация ресурсов, ресурс – это информация, которая демонстрируется внешнему миру через приложение и имеет идентификатор, не меняющийся при изменении состояния ресурса. Идентификатором в *REST* является *URI*.

– манипуляция ресурсами через представление, ресурсы могут быть представлены различными способами: *JSON, HTML, XML*. Представление является описанием текущего состояния ресурса и используется для выполнения операций над ресурсом. Взаимодействие клиентов с ресурсами происходит посредством представлений.

– «самоописываемые» сообщения, каждое сообщение содержит достаточно информации, чтобы понять, каким образом его обрабатывать.

– гипермедиа, как средство изменения состояния приложения, клиенты изменяют состояние системы только через действия, которые динамически определены в гипермедиа на сервере. Исключая простые точки входа в приложение, клиент не может предположить, что доступна какая-то операция над каким-то ресурсом, если не получил информацию об этом в предыдущих запросах к серверу.

1.2.7.5 Многослойная система. В архитектуре *REST* возможно разделение системы на иерархию слоев, но с условием: отдельный компонент системы может видеть только компоненты следующего уровня. Использование промежуточных серверов способно повысить масштабируемость за счёт балансировки нагрузки и распределённого кэширования. Также это позволяет использовать политику безопасности для обеспечения конфиденциальности данных.

1.2.7.6 Код по требованию. Является необязательным требованием. В архитектуре *REST* данное условие позволяет расширить функциональность клиента за счёт загрузки кода с сервера в виде апплетов или скриптов.

1.3 Реализация

В данной работе приведены результаты разработки микросервиса учета загрузки сотрудников предприятия. Функциональное назначение данного микросервиса – агрегация и хранение активностей пользователей. На этапе проектирования сервиса было выделено три пользовательских роли: сотрудник, руководитель проекта, администратор. Диаграмма вариантов использования представлена в Приложении А.

Действия с какой-либо сущностью подразумевают создание, изменение или удаление сущности. Действия с группами сущностей включают создание или удаление групп сущностей.

1.3.1 Структура базы данных

При разработке сервиса была использована СУБД *PostgreSQL*. Далее представлено описание структуры разработанной базы данных. Концептуальная, логическая и физическая модели базы данных приведены в Приложении Б. Таблицы 1-12 содержат описание атрибутов таблиц базы данных.

Таблица 1 – Структура таблицы «Должность (*pl_position*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор должности	*INTEGER*	*PK*
name	Наименование должности	*VARCHAR*(255)	*NOT NULL, UNIQUE*

Таблица 2 – Структура таблицы «Пользователь системы (*pl_user*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор пользователя	*INTEGER*	*PK*
email	*E-mail* пользователя	*VARCHAR*(255)	*NOT NULL, UNIQUE*
password_hash	Хэш пароля	*VARCHAR*(255)	*NOT NULL*
full_name	Полное имя пользователя	*VARCHAR*(512)	*NOT NULL*
role_id	Роль пользователя	*INTEGER*	*NOT NULL*
rate	Ставка	*NUMERIC*(3, 2)	*NOT NULL*
type_activity	Вид деятельности	*VARCHAR*(512)	-
created_at	Дата создания пользователя	*TIMESTAMP*	*NOT NULL*
last_login	Дата последнего захода пользователя в систему	*TIMESTAMP*	-
is_enabled	Признак активности пользователя	*BOOLEAN*	*NOT NULL*
group_id	Группа пользователя	*INTEGER*	-
position_id	Должность	*INTEGER*	-
outsourcing	Аутсорсинг	*BOOLEAN*	*NOT NULL*

Название поля	Описание поля	Тип данных	Ограничения
deleted	Признак удаления	*BOOLEAN*	*NOT NULL*

Таблица 3 – Структура таблицы «Таблица *refresh*-токенов (*refresh_token*)»

Название поля	Описание поля	Тип данных	Ограничения
hash	Хэш токена *SHA*256	*VARCHAR*(128)	*PK*
user_id	Пользователь	*INTEGER*	*NOT NULL*

Таблица 4 – Структура таблицы «Группа пользователя (структурная единица организации) (*pl_user_group*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор группы	*INTEGER*	*PK*
name	Наименование группы	*VARCHAR*(255)	*NOT NULL, UNIQUE*

Таблица 5 – Структура таблицы «Роль пользователя в системе (*pl_user_role*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор роли	*INTEGER*	*PK*
name	Наименование роли	*VARCHAR*(255)	*NOT NULL, UNIQUE*

Таблица 6 – Структура таблицы «Нерабочий день для сотрудника (*pl_user_out_day*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор	*INTEGER*	*PK*
user_id	Сотрудник	*INTEGER*	*NOT NULL*
out_day	Нерабочий день	*DATE*	*NOT NULL*
out_type	Тип нерабочего дня (отпуск, больничный, отгул)	*INTEGER*	*NOT NULL*
duration_min	Длительность нерабочего дня	*INTEGER*	*NOT NULL*

Таблица 7 – Структура таблицы «Группа контракта (*pl_contract_group*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор группы	*INTEGER*	*PK*
name	Наименование группы	*VARCHAR*(255)	*NOT NULL, UNIQUE*

Таблица 8 – Структура таблицы «Производственный календарь (*pl_prod_calendar*)»

Название поля	Описание поля	Тип данных	Ограничения
year	Год	*INTEGER*	*PK*
month	Месяц	*INTEGER*	*PK*
days_out_mask	Маска выходных дней	*INTEGER*	*NOT NULL*
days_short_mask	Маска сокращенных дней	*INTEGER*	*NOT NULL*
work_total_min	Общее рабочее время в минутах	*INTEGER*	*NOT NULL*
is_enabled	Активность календаря	*BOOLEAN*	*NOT NULL*

Таблица 9 – Структура таблицы «Контракт (*pl_contract*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор контракта	*INTEGER*	*PK*
name	Наименование	*VARCHAR*(255)	*UNIQUE*
full_name	Полное наименование	*VARCHAR*(512)	*NOT NULL, UNIQUE*
user_rp_id	Руководитель проекта	*INTEGER*	*NOT NULL*
is_enabled	Признак активного контракта	*BOOLEAN*	*NOT NULL*
group_id	Группа контракта	*INTEGER*	-
docs	Проектные документы (ссылка)	*VARCHAR*(512)	-
deleted	Признак удаления	*BOOLEAN*	*NOT NULL*
created_at	Дата создания записи	*TIMESTAMP*	*NOT NULL*
created_by	Кем создана запись	*VARCHAR*(255)	-

Название поля	Описание поля	Тип данных	Ограничения
created_by_id	Кем создана запись, идентификатор пользователя	*INTEGER*	-
updated_at	Время изменения записи	*TIMESTAMP*	-
updated_by	Кем изменена запись	*VARCHAR*(255)	-
updated_by_id	Кем изменена запись, идентификатор пользователя	*INTEGER*	-

Таблица 10 – Структура таблицы «Отчет по активностям (*pl_sum_activity*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор отчета	*INTEGER*	*PK*
year	Год	*INTEGER*	*NOT NULL*
month	Месяц	*INTEGER*	*NOT NULL*
user_id	Пользователь	*INTEGER*	*NOT NULL*
contract_id	Контракт	*INTEGER*	*NOT NULL*
work_min	Количество отработанного времени в минутах	*INTEGER*	*NOT NULL*
work_need_min	Количество необходимого отработанного времени в минутах	*INTEGER*	*NOT NULL*
completed	Показатель завершенности отработанного времени	*NUMERIC* (5, 4)	*NOT NULL*

Таблица 11 – Структура таблицы «Активность (*pl_activity*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор активности	*BIGINT*	-
contract_id	Контракт	*INTEGER*	*NOT NULL*
duration_min	Время активности в минутах	*INTEGER*	*NOT NULL*
duration	Время активности	*VARCHAR*(255)	*NOT NULL*
date_of	Дата активности	*DATE*	*NOT NULL*
task_url	*URL* задачи в *agile* системе	*VARCHAR* (1024)	-
task_desc	Описание задачи	*VARCHAR*(255)	-

Название поля	Описание поля	Тип данных	Ограничения
user_id	Пользователь	*INTEGER*	*NOT NULL*
created_at	Дата создания записи	*TIMESTAMP*	*NOT NULL*
created_by	Кем создана запись	*VARCHAR*(255)	-
created_by_id	Кем создана запись, идентификатор пользователя	*INTEGER*	-
updated_at	Время изменения записи	*TIMESTAMP*	-
updated_by	Кем изменена запись	*VARCHAR*(255)	-
updated_by_id	Кем изменена запись, идентификатор пользователя	*INTEGER*	-
__version	Версия записи	*BIGINT*	*NOT NULL*

Таблица 12 – Структура таблицы «Пользователь – Контракт (*pl_user_pl_contract*)»

Название поля	Описание поля	Тип данных	Ограничения
id	Идентификатор	*SERIAL*	*PK*
user_id	Пользователь	*INTEGER*	*NOT NULL*
contract_id	Контракт	*INTEGER*	*NOT NULL*

В таблице «Активность» поле «*id*» не является первичным ключом поскольку таблица подвергается партицированию – разбиению на логические части по выбранным критериям. Партицированные или секционированные таблицы призваны уменьшить время поиска записи. В данной реализации к одной партиции относятся все активности за год – ключом партиции стала дата активности.

Агрегатная таблица «Отчет по активностям» хранит информацию о количестве отработанного времени пользователей на проектах, данные в ней обновляются при создании, изменении или удалении каждой активности. Впоследствии таблица будет использоваться формирующим отчетность

микросервисом. Агрегатная таблица обеспечивает быструю выборку массива данных при формировании отчетности.

Листинг *SQL*-файла, формирующего базу данных, приведен в Приложении В.

1.3.2 Описание *REST API*

Разработка *API* является одним из важнейших этапов создания микросервиса поскольку от полноты реализации функционала *API* зависит удобство реализации остальных микросервисов и производительность итогового продукта как совокупности взаимодействующих микросервисов. *API* соответствует архитектурному стилю *REST* и поддерживает следующие функции:

- создание активности;
- удаление активности;
- удаление активностей;
- создание активностей;
- обновление активности;
- поиск активностей;
- мой календарь со сводной информацией по активностям;
- мои активности за месяц сгруппированные по дням;
- мои активности за день;
- моя сводная информация за месяц;
- календарь со сводной информацией по активностям пользователя;
- активности пользователя за месяц сгруппированные по дням;
- активности пользователя за день;
- сводная информация пользователя за месяц.

В качестве примера – запрос на создание активности. Спецификация *REST API* требует того, чтобы при операциях создания использовался *HTTP* метод *POST*. Также представляются требования к формированию *URI* запроса,

например, для данного запроса *URI* имеет следующую структуру: название микросервиса/название сущности. Тело запроса состоит из полей, которые необходимы для создания активности. При успешном создании активности возвращается код 200 и объект, описывающий созданную активность (который включает в себя поля, переданные при создании, и новые поля, добавленные микросервисом или БД). В случае ошибки выдается код 400 и сообщение «*Bad Request*».

Запрос на создание активности:

Метод: *POST*

URI: /project-load/*activity*

Тело запроса:

```
        {
          "contractId": int,
          "dateOf": "string",
          "duration": "string",
          "durationMin": int,
          "task": {
            "desc": "string",
            "url": "string"
          },
          "userId": int
        }
```

Ответ: 200 OK

```
        {
          "id": int,
          "contract": null,
          "contractId": int,
          "duration": "string",
          "durationMin": int,
          "dateOf": [ date  ],
          "task": {
            "url": "string",
```

```
    "desc": "string"
  },
  "audit": {
    "createdAt": [ date ],
    "createdBy": "string",
    "createdById": int,
    "updatedAt": null,
    "updatedBy": null,
    "updatedById": null
  },
  "version": 0,
  "user": null,
  "userId": int
}
400 Bad Request
{ "errorMessage": "string" }
```

Полное описание *REST API* для работы с активностями приведено в Приложении Г.

1.3.3 Описание классов

В микросервисе для взаимодействия ранее описанных СУБД и *REST API* архитектура разделена на основные функциональные блоки:

– классы-контроллеры – выполняют запросы *REST API*;

– классы-сущности – отвечают за описание сущностей базы данных и связей между ними в программе. Классы-сущности облегчают работу с данными, полученными как результат использования методов классов-репозиториев;

– классы-представления – также хранят сущности, но не связи между ними, необходимы для вывода данных сущностей за пределы микросервиса;

– классы-репозитории – выполняют взаимодействие СУБД с сущностями. Данные передаются в эти классы из классов-сервисов;

– классы-сервисы – обеспечивают взаимодействие классов-контроллеров с классами-репозиториями посредством промежуточной обработки данных.

Далее на примере сущности «Активность» рассматривается взаимодействие всех типов классов. *UML*-диаграмма классов сущности «Активность» представлена в Приложении А.

Класс-сущность описывает структуру таблицы «Активность», а также описывается отношение сущности к остальным сущностям (сущности «Пользователь» и «Контакт»). Класс содержит методы, позволяющие получить или установить то или иное поле объекта (контракт, время активности, дата активности, описание задачи и другие).

Классы-представления не редкость при использовании объектно-реляционных отображений. В случае, когда *ORM* динамически загружает из базы данных объекты, связанные с текущим, может возникнуть ситуация «вечной загрузки». Чтобы этого избежать используются представления. Класс-представление «*ActivityView*» также описывает структуру таблицы «Активность», однако не хранит связей с другими сущностями. Класс содержит методы, позволяющие получить или установить поле объекта, наследуется от интерфейса *Serializable*.

Класс-контроллер «*ActivityController*» осуществляет обработку входящих запросов, валидацию входных данных, вызов методов класса-сервиса «*ActivityService*» и возврат результата.

Класс-репозиторий (интерфейс) «*ActivityRepo*» работает с СУБД и осуществляет операции создания, обновления, удаления и поиска активностей. При этом операции поиска представляют собой набор специфичных для каждого обрабатываемого контроллером запроса функций, которые ищут пользователя на основе ограничений, передаваемых при

запросе: например, поиск активностей пользователя на конкретный интервал времени.

Класс-сервис «*ActivityService*» предоставляет методы по обработке входных данных, описывающих активности, сохраняет сущности в БД, вызывая методы класса-репозитория, и, напротив, получает объекты из БД и готовит выходные данные.

1.3.4 Программная реализация

Как упоминалось в предыдущем разделе, для реализации данного микросервиса используется язык программирования *Java* и фреймворк *Spring*. Был реализован *API* для работы с активностями, контрактами, пользователями, их ролями и должностями, а также функции для работы с ними.

В Приложении А представлена *UML*-диаграмма последовательностей при обработке запроса «Создание активности».

С клиентской части в микросервис в объект класса «*ActivityController*» поступает запрос на создание новой активности. Класс-контроллер осуществляет проверку запроса, в частности тела запроса. Если запрос валиден, то класс-контроллер передает запрос в класс-сервис. Класс-сервис подготавливает запрос к взаимодействию с объектами класса-репозитория – входные данные преобразует в экземпляр класса-сущности. Класс-репозиторий отправляет к БД запрос на создание новой записи. Затем, когда данные получены из БД, класс-репозиторий преобразует данные в объекты программы и передает их в класс-сервис. Класс-сервис формирует ответ для класса-контроллера в виде экземпляра класса-представления, класс-контроллер выдает результат, который передается клиентской части. Пользователю виден результат использования функционала.

Исходный код на языке *Java* сущности «Активность», представления, контроллера, сервиса и репозитория приведен в Приложении Д.

Диаграмма развертывания реализованного микросервиса представлена в Приложении А.

1.4 Описание результата

В результате компиляции программы был получен исполняемый файл «*project-load.jar*». Для его запуска можно использовать команду «*Java –jar project-load.jar*». Результат запуска представлен на рисунке 3.

Для проверки работоспособности приложения используется программа *Insomnia*, поскольку она бесплатная и обладает широким функционалом по отладке *REST API*.

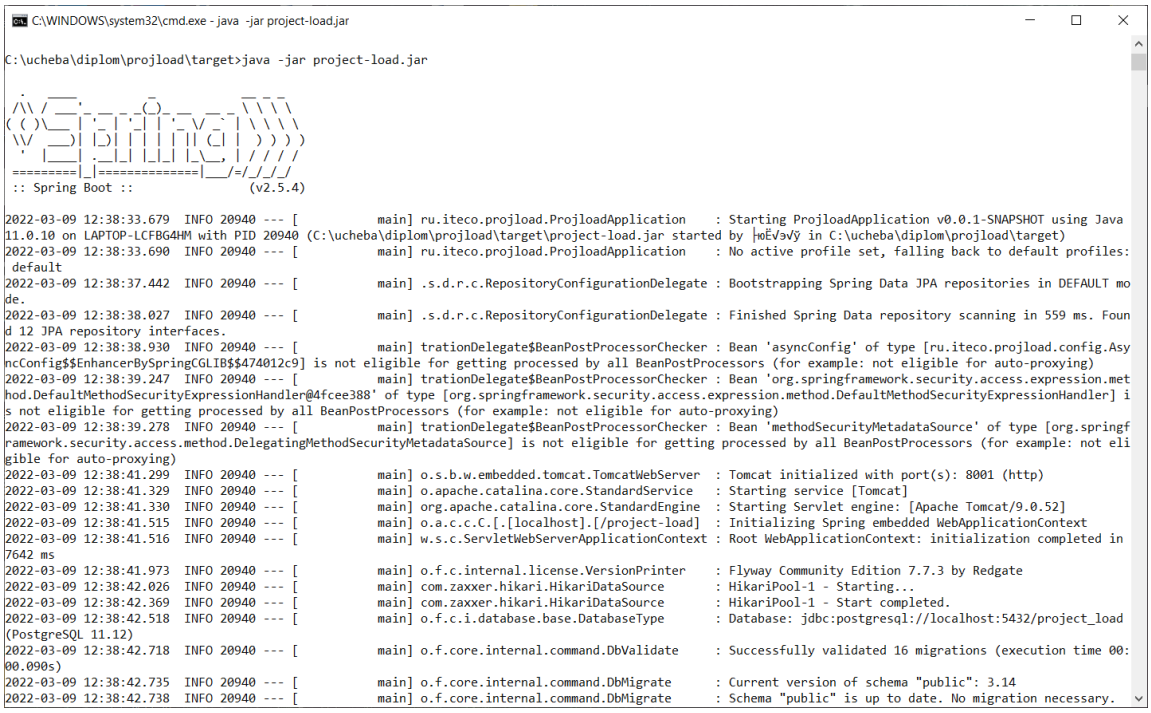


Рисунок 3 – Результат работы исполняемого файла «*project-load.jar*» сразу после запуска

Предварительно данными заполняются справочники «*pl_position*», «*pl_user_group*», «*pl_user_role*», «*pl_contract_group*». Затем вносится информация пользователя: при этом используется запрос «*POST /project-load/user/register*», в тело которого добавляется описание создаваемого

пользователя в формате *JSON*. При успешном выполнении запроса возвращается объект с описанием пользователя, в котором особое внимание стоит уделить полю «*id*». Значение этого поля необходимо указать в запросе создания активности в поле «*userId*». Результат выполнения запроса на создание пользователя представлен на рисунке 4.

Данные созданного пользователя можно проконтролировать *SQL*-запросом к базе данных «*SELECT * FROM pl_user*», что представлено на рисунках 5-6, или запросом «*GET /project-load/user/*9», где 9 – идентификатор новосозданного пользователя. Результат выполнения *GET*-запроса предложен на рисунке 7.

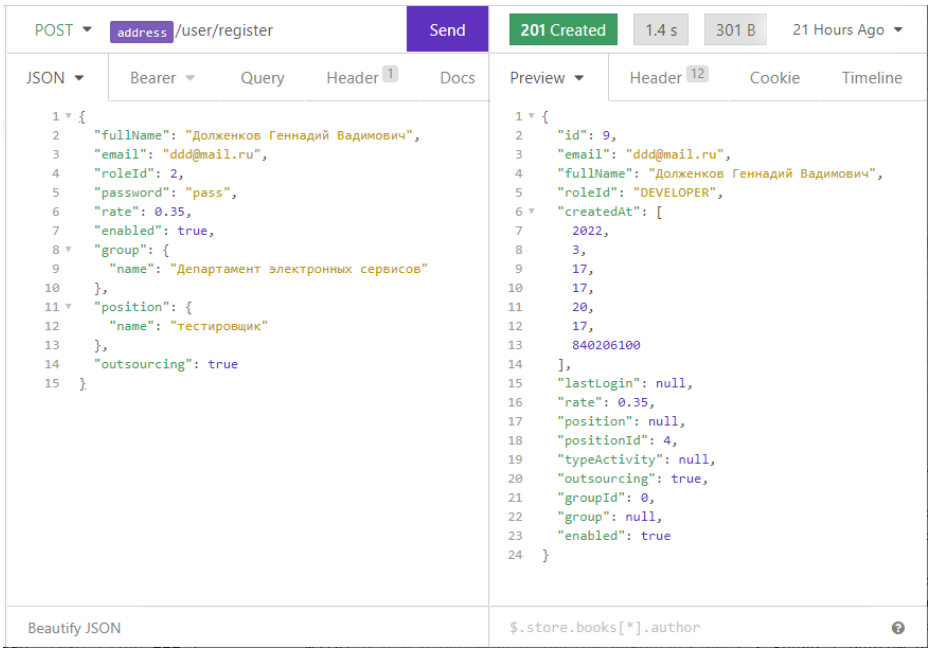


Рисунок 4 – Результат выполнения запроса на создание пользователя

6 rows Tx: Auto

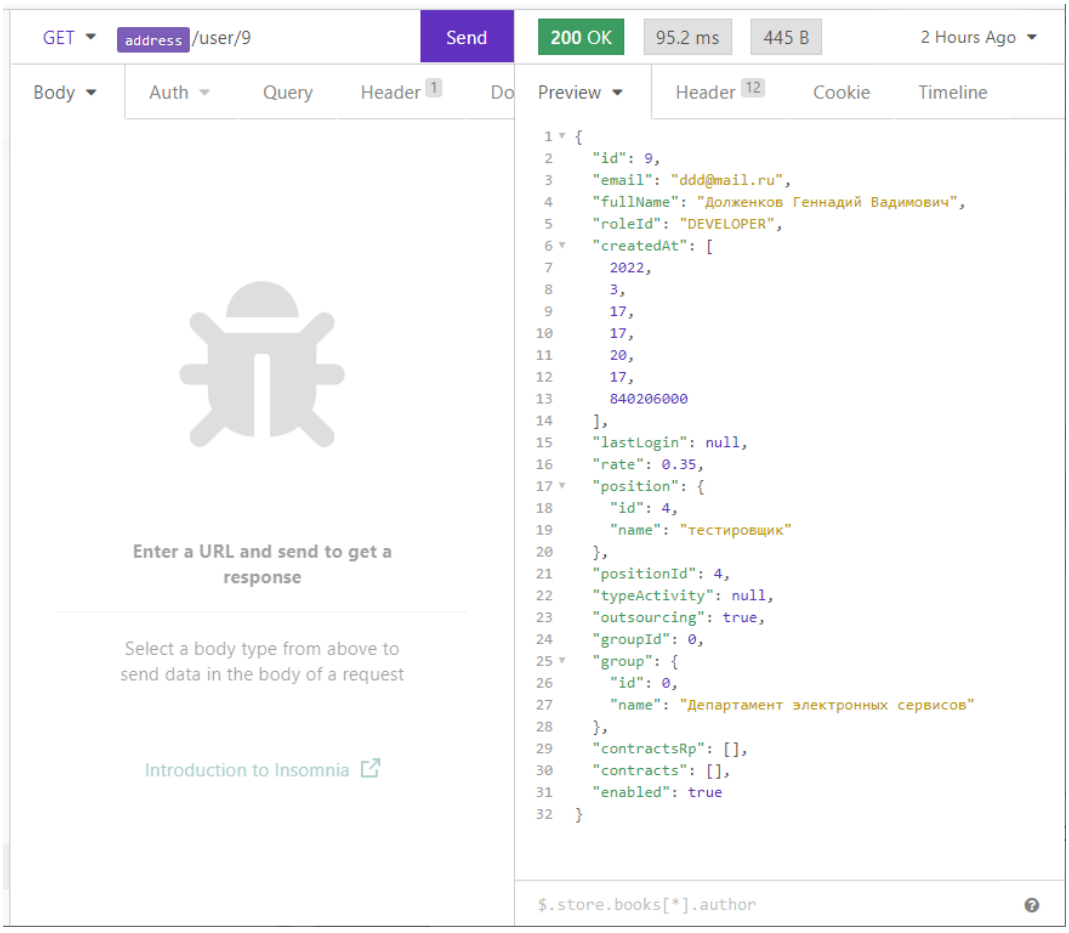
	…	email	password_hash	full_name	role_id	rate
1	1	admin	$2a$08$z6Q6600cvmme…	Иванов Иван Иванович	0	1.00
2	2	svetlana1995@mail.ru&…	$2a$08$PrsJIIgn/vtL…	Тутошкина Светлана Андреевна	2	1.00
3	3	hena1995@mail.ru&&015…	$2a$08$Fvlml9F7X6jd…	Алмазов Геннадий Сергеевич	2	1.00
4	7	shikanom@mail.ru&&c5e…	$2a$08$ZTfDiR/CI6rr…	Шиканов Василий Романович	2	0.35
5	8	asp@mail.ru	$2a$08$tNiuEWdbPCcN…	Аспарагус Виктория Вадимовна	2	0.35
6	9	ddd@mail.ru	$2a$08$jADwLvKCfEIR…	Долженков Геннадий Вадимович	2	0.35

Рисунок 5 – Результат выполнения запроса к базе данных, начало таблицы «*pl_user*»

Tab-se...d (TSV) DDL Tab-se...d (TSV) DDL View Query

type_activi...	created_at	last_log...	is_enabled	group_...	position_id	outsourcing	deleted
<null>	2022-03-03 13:03:54....	2022-03-15 ...	☑	<null>	<null>	☐	☐
Разработка ПО	2022-03-09 14:18:21....	<null>	☐	<null>	<null>	☐	☑
Разработка ПО	2022-03-09 14:41:34....	<null>	☐	<null>	<null>	☐	☑
<null>	2022-03-15 13:38:21....	<null>	☑	0	4	☑	☑
<null>	2022-03-15 14:28:30....	2022-03-15 ...	☐	0	4	☑	☐
<null>	2022-03-17 17:20:17....	<null>	☑	0	4	☑	☐

Рисунок 6 – Результат выполнения запроса к базе данных, окончание таблицы *«pl_user»*




```
{
  "id": 9,
  "email": "ddd@mail.ru",
  "fullName": "Долженков Геннадий Вадимович",
  "roleId": "DEVELOPER",
  "createdAt": [
    2022,
    3,
    17,
    17,
    20,
    17,
    840206000
  ],
  "lastLogin": null,
  "rate": 0.35,
  "position": {
    "id": 4,
    "name": "тестировщик"
  },
  "positionId": 4,
  "typeActivity": null,
  "outsourcing": true,
  "groupId": 0,
  "group": {
    "id": 0,
    "name": "Департамент электронных сервисов"
  },
  "contractsRp": [],
  "contracts": [],
  "enabled": true
}
```

Рисунок 7 – Результат выполнения *GET*-запроса «Получение пользователя»

Структура базы данных подразумевает, что активность пользователя не может существовать в отрыве от контракта, поэтому создается контракт запросом «*POST /project-load/contract*». В случае успешного создания возвращается объект с описанием созданного контракта так же в формате *JSON*. Помимо переданных входных данных итоговая запись содержит дополнительные поля аудита, необходимые для отслеживания действий над

записью. Значение поля «*id*» впоследствии указывается в запросе создания активности в поле «*contractId*». Результат выполнения запроса на создание контракта представлен на рисунке 8.

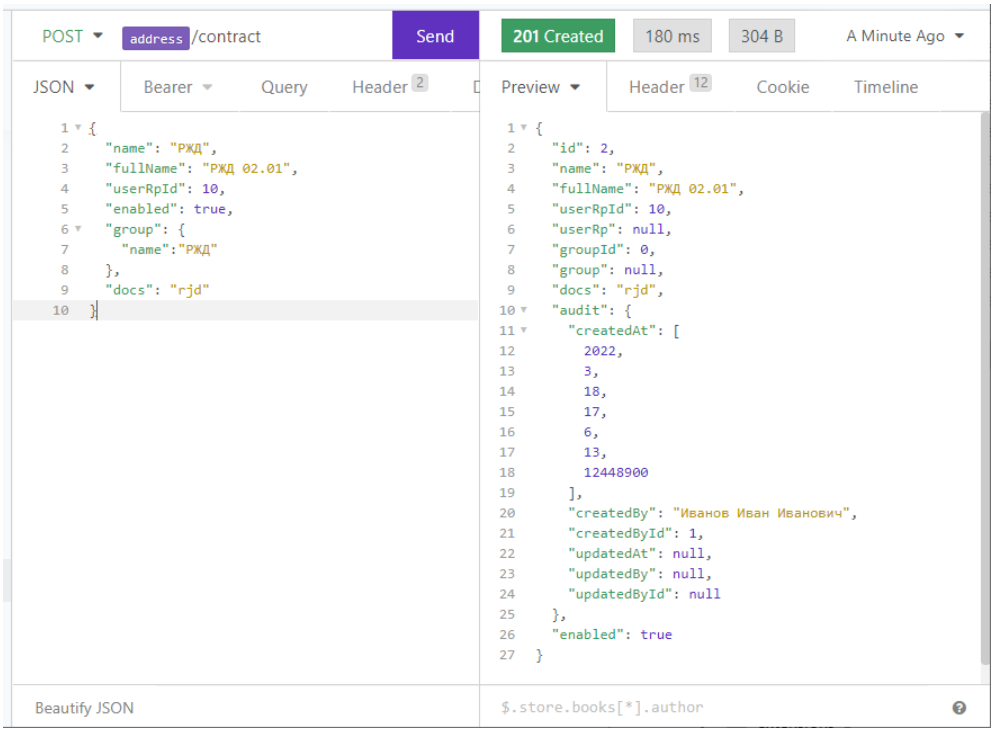


Рисунок 8 – Результат выполнения запроса на создание контракта

Проверить внесенные данные можно также двумя способами: запросом к БД «*SELECT * FROM pl_contract*» или запросом «*GET /project-load/contract/2*», где 2 – идентификатор нового контракта. Результаты двух запросов представлены на рисунках 9-11.

	id	name	full_name	user_rp_id	is_enabled	created_at	created_by
1	1	taest1	Профтех ДДР-001a	1	☑	2022-03-09 00:54:04.764950	Иванов Иван Иванович
2	2	РЖД	РЖД 02.01	10	☑	2022-03-18 17:06:13.012449	Иванов Иван Иванович

Рисунок 9 – Результат выполнения запроса к базе данных, начало таблицы «*pl_contract*»

created_by_id	updated_at	updated_by	updated_by_id	group_id	docs	deleted
1	<null>	<null>	<null>	<null>	dsa	☐
1	<null>	<null>	<null>	0	rjd	☐

Рисунок 10 – Результат выполнения запроса к базе данных, окончание таблицы «*pl_contract*»

Рисунок 11 – Результат выполнения *GET*-запроса «Получение контракта»

Далее запросом «*POST /project-load/activity*» создается активность. В тело запроса добавляется описание в формате *JSON* с данными новой активности, при этом в поля «*userId*» и «*contractId*» заполняются идентификаторами новосозданных сущностей. В случае успешного создания

возвращается объект с описанием созданной активности с дополнительными полями аудита также в формате *JSON*. Результат выполнения запроса на создание активности представлен на рисунке 12.

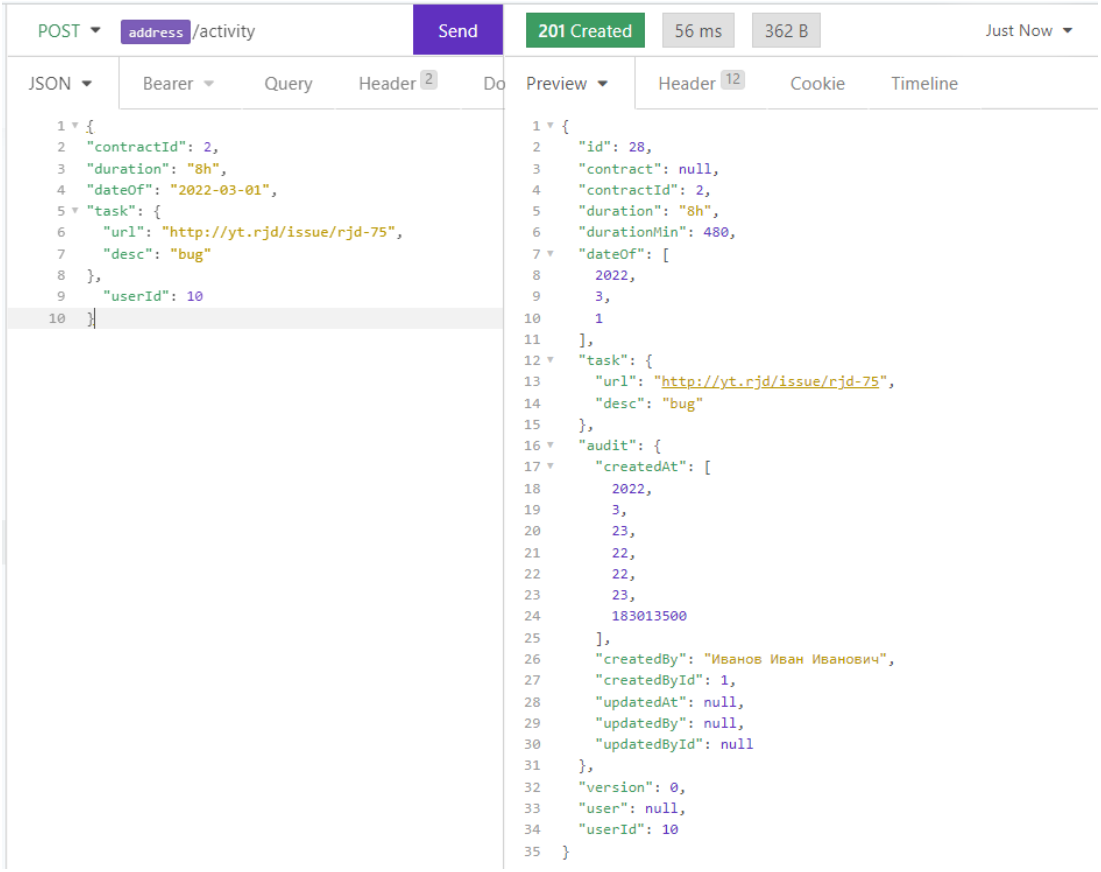


Рисунок 12 – Результат выполнения запроса на создание активности

Проверить внесенные данные можно запросом к БД «*SELECT * FROM pl_activity*». Результат запроса представлен на рисунках 13-14.

	id	contract_id	duration_min	duration	date_of	task_url	task_desc	user_id
13	13	2	480	8h	2022-04-06	http://yt.rjd	important	10
14	14	2	480	8h	2022-04-05	http://yt.rjd	important	10
15	15	2	480	8h	2022-04-04	http://yt.rjd	important	10
16	16	2	480	8h	2022-04-01	http://yt.rjd	important	10
17	17	1	480	8h	2022-04-29	http://www.google.ru	bug	10
18	18	1	480	8h	2022-04-28	http://www.google.ru	bug	10
19	19	1	480	8h	2022-04-27	http://www.google.ru	bug	10
20	20	1	480	8h	2022-04-26	http://www.google.ru	bug	10
21	21	1	480	8h	2022-04-25	http://www.google.ru	bug	10
22	22	1	480	8h	2022-04-22	http://www.google.ru	bug	10
23	23	1	480	8h	2022-04-21	http://www.google.ru	bug	10
24	24	1	480	8h	2022-04-20	http://www.google.ru	bug	10
25	25	1	480	8h	2022-04-19	http://www.google.ru	bug	10
26	26	1	480	8h	2022-04-18	http://www.google.ru	bug	10
27	28	2	480	8h	2022-03-01	http://yt.rjd/issue/rjd-75	bug	10

Рисунок 13 – Результат выполнения запроса к базе данных, начало таблицы «*pl_activity*»

created_at	created_by	created_by_id	updated_at	updated_by	updated_by_id	_version
2022-03-23 21:52:59.400025	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:52:59.401025	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:52:59.402024	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:52:59.402024	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.130828	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.132829	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.133830	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.134829	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.135828	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.136828	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.137828	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.138828	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.139830	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.140832	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 22:22:23.183014	Иванов Иван Иванович	1	<null>	<null>	<null>	0

Рисунок 14 – Результат выполнения запроса к базе данных, окончание таблицы «*pl_activity*»

Если пользователь несколько дней кряду работал над одной задачей, предусмотрена возможность заполнения активности на нескольких дней. В таком случае используется запрос «*POST /project-load/activity/bulk*», в тело запроса добавляется описание в формате *JSON* с данными группы активностей. Каждая активность из группы проходит валидацию. В случае успешного создания возвращаются объекты с описанием созданных активностей с дополнительными полями аудита также в формате *JSON*. Результат выполнения запроса представлен на рисунке 15.

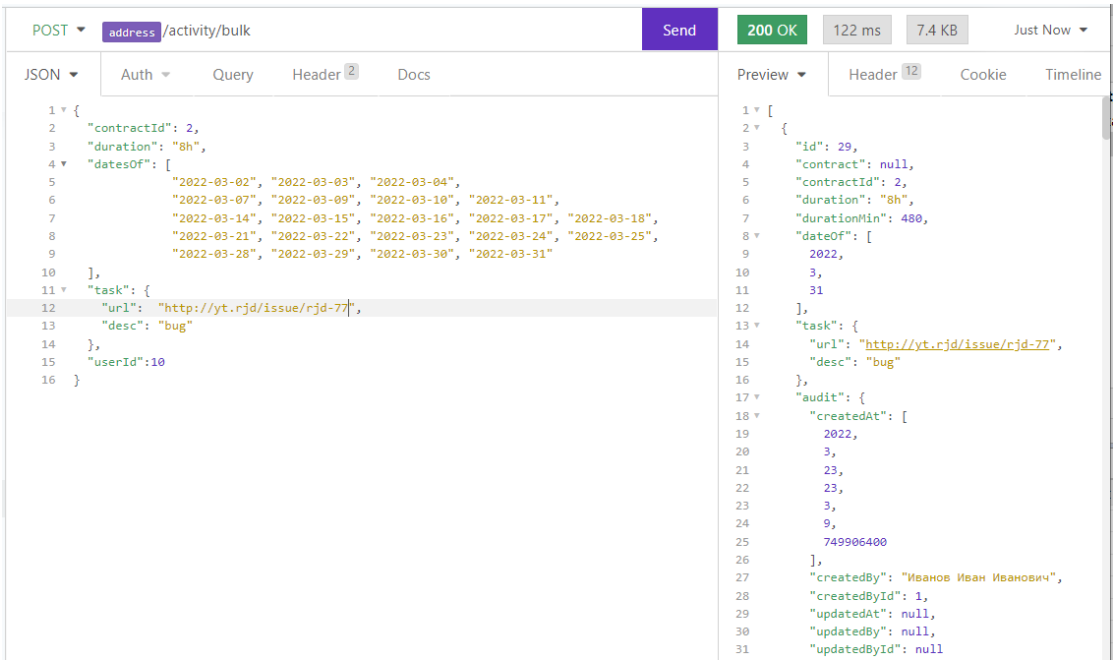


Рисунок 15 – Результат выполнения запроса на создание группы активностей

Проверить внесенные данные можно запросом к БД «*SELECT * FROM pl_activity*». Результат запроса представлен на рисунках 16-17.

	id	contract_id	duration_min	duration	date_of	task_url	task_desc	user_id
24	24	1	480	8h	2022-04-20	http://www.google.ru	bug	10
25	25	1	480	8h	2022-04-19	http://www.google.ru	bug	10
26	26	1	480	8h	2022-04-18	http://www.google.ru	bug	10
27	28	2	480	8h	2022-03-01	http://yt.rjd/issue/rjd-75	bug	10
28	29	2	480	8h	2022-03-31	http://yt.rjd/issue/rjd-77	bug	10
29	30	2	480	8h	2022-03-30	http://yt.rjd/issue/rjd-77	bug	10
30	31	2	480	8h	2022-03-29	http://yt.rjd/issue/rjd-77	bug	10
31	32	2	480	8h	2022-03-28	http://yt.rjd/issue/rjd-77	bug	10
32	33	2	480	8h	2022-03-25	http://yt.rjd/issue/rjd-77	bug	10
33	34	2	480	8h	2022-03-24	http://yt.rjd/issue/rjd-77	bug	10
34	35	2	480	8h	2022-03-23	http://yt.rjd/issue/rjd-77	bug	10
35	36	2	480	8h	2022-03-22	http://yt.rjd/issue/rjd-77	bug	10
36	37	2	480	8h	2022-03-21	http://yt.rjd/issue/rjd-77	bug	10
37	38	2	480	8h	2022-03-18	http://yt.rjd/issue/rjd-77	bug	10
38	39	2	480	8h	2022-03-17	http://yt.rjd/issue/rjd-77	bug	10
39	40	2	480	8h	2022-03-16	http://yt.rjd/issue/rjd-77	bug	10
40	41	2	480	8h	2022-03-15	http://yt.rjd/issue/rjd-77	bug	10
41	42	2	480	8h	2022-03-14	http://yt.rjd/issue/rjd-77	bug	10
42	43	2	480	8h	2022-03-11	http://yt.rjd/issue/rjd-77	bug	10
43	44	2	480	8h	2022-03-10	http://yt.rjd/issue/rjd-77	bug	10
44	45	2	480	8h	2022-03-09	http://yt.rjd/issue/rjd-77	bug	10
45	46	2	480	8h	2022-03-07	http://yt.rjd/issue/rjd-77	bug	10
46	47	2	480	8h	2022-03-04	http://yt.rjd/issue/rjd-77	bug	10
47	48	2	480	8h	2022-03-03	http://yt.rjd/issue/rjd-77	bug	10
48	49	2	480	8h	2022-03-02	http://yt.rjd/issue/rjd-77	bug	10

Рисунок 16 – Результат выполнения запроса к базе данных после добавления группы активностей, начало таблицы «*pl_activity*»

created_at	created_by	created_by_id	updated_at	updated_by	updated_by_id	__version
2022-03-23 21:59:03.138828	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.139830	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 21:59:03.140832	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 22:22:23.183014	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.749906	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.750874	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.751873	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.752874	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.753877	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.754873	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.755874	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.755874	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.756873	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.757873	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.758883	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.759785	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.760497	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.761027	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.762037	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.762037	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.763041	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.764048	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.764048	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.765039	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.766041	Иванов Иван Иванович	1	<null>	<null>	<null>	0

Рисунок 17 – Результат выполнения запроса к базе данных после добавления группы активностей, окончание таблицы «*pl_activity*»

Одновременно с внесением активностей изменяется и таблица «*pl_sum_activity*», ее записи получены запросом «*SELECT * FROM pl_sum_activity*». Таблица представлена на рисунке 18.

	id	year	month	user_id	contract_id	work_min	work_need_min	completed
1	2	2021	1	3	1	500	7200	0.0694
2	1	2021	1	1	1	620	7200	0.0861
3	3	2019	1	3	1	600	8160	0.0735
4	4	2022	4	10	2	5760	10080	0.5714
5	5	2022	4	10	1	4800	10080	0.4762
6	6	2022	3	10	2	10560	10500	1.0057

Рисунок 18 – Записи таблицы «*pl_sum_activity*»

В шестой записи хранится свернутая информация о работе пользователя Кириллова Романа Сергеевича (*id* = 10) на проекте «РЖД» (*id* = 2) за март 2022 года. Как видно на рисунке, значение в столбце «*work_min*» превышает «*work_need_min*» на 60 минут – пользователь имеет переработку в час в сокращенный день 7 марта. Изменить активность можно запросом «*PUT /project-load/activity/[activityId]*», в тело которого добавляется измененное описание активности в формате *JSON*. Вместо [*activityId*] подставляется идентификатор изменяемой активности, в примере *activityId* равен 46. В запросе передается новая длительность активности *duration* – 7 часов. Результат выполнения запроса на изменение активности представлен на рисунке 19.

Результат выполнения запроса «*SELECT * FROM pl_activity*» к базе данных после изменения активности приведен на рисунках 20-21. Кроме длительности активности изменились также поля аудита «*updated_at*», «*updated_by*», «*updated_by_id*» и «*_version*».

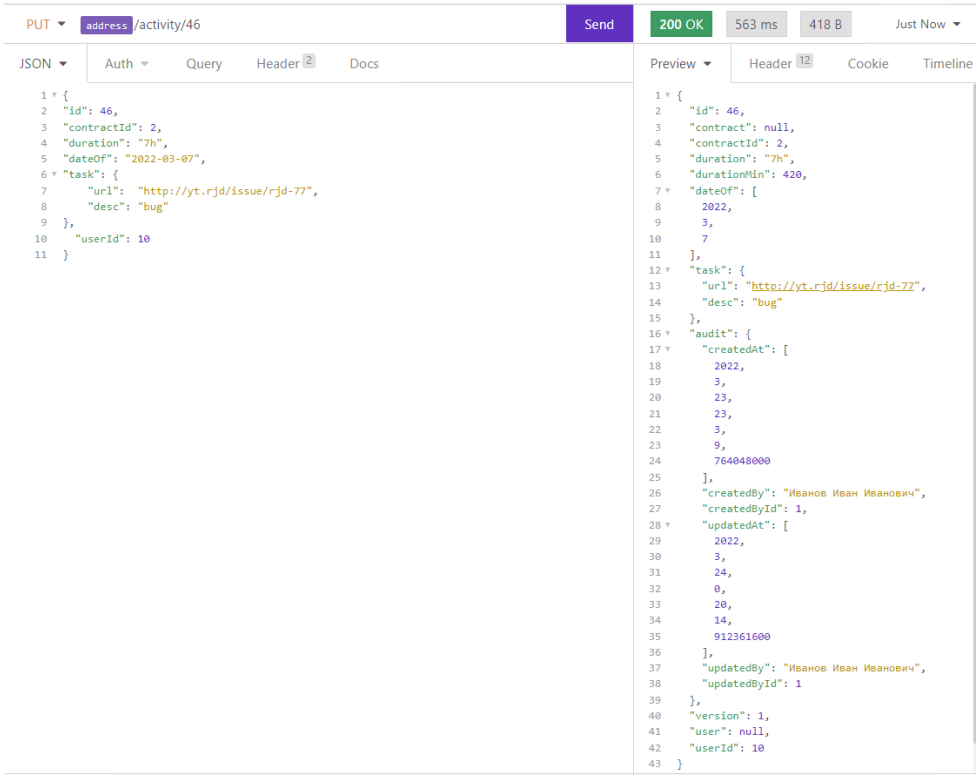


Рисунок 19 – Результат выполнения запроса на изменение активности

	id	contract_id	duration_min	duration	date_of	task_url	task_desc	user_id
33	34	2	480	8h	2022-03-24	http://yt.rjd/issue/rjd-77	bug	10
34	35	2	480	8h	2022-03-23	http://yt.rjd/issue/rjd-77	bug	10
35	36	2	480	8h	2022-03-22	http://yt.rjd/issue/rjd-77	bug	10
36	37	2	480	8h	2022-03-21	http://yt.rjd/issue/rjd-77	bug	10
37	38	2	480	8h	2022-03-18	http://yt.rjd/issue/rjd-77	bug	10
38	39	2	480	8h	2022-03-17	http://yt.rjd/issue/rjd-77	bug	10
39	40	2	480	8h	2022-03-16	http://yt.rjd/issue/rjd-77	bug	10
40	41	2	480	8h	2022-03-15	http://yt.rjd/issue/rjd-77	bug	10
41	42	2	480	8h	2022-03-14	http://yt.rjd/issue/rjd-77	bug	10
42	43	2	480	8h	2022-03-11	http://yt.rjd/issue/rjd-77	bug	10
43	44	2	480	8h	2022-03-10	http://yt.rjd/issue/rjd-77	bug	10
44	45	2	480	8h	2022-03-09	http://yt.rjd/issue/rjd-77	bug	10
45	47	2	480	8h	2022-03-04	http://yt.rjd/issue/rjd-77	bug	10
46	48	2	480	8h	2022-03-03	http://yt.rjd/issue/rjd-77	bug	10
47	49	2	480	8h	2022-03-02	http://yt.rjd/issue/rjd-77	bug	10
48	46	2	420	7h	2022-03-07	http://yt.rjd/issue/rjd-77	bug	10

Рисунок 20 – Результат выполнения запроса к базе данных после изменения активности, начало таблицы «*pl_activity*»

created_at	created_by	created_by_id	updated_at	updated_by	updated_by_id	_version
2022-03-23 23:03:09.754873	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.755874	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.755874	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.756873	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.757873	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.758883	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.759785	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.760497	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.761027	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.762037	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.762037	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.763041	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.764048	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.765039	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.766041	Иванов Иван Иванович	1	<null>	<null>	<null>	0
2022-03-23 23:03:09.764048	Иванов Иван Иванович	1	2022-03-24 00:20:14.912362	Иванов Иван Иванович	1	1

Рисунок 21 – Результат выполнения запроса к базе данных после изменения активности, окончание таблицы «*pl_activity*»

Изменение или удаление активностей также влияет на данные в таблице «*pl_sum_activity*», ее записи получены запросом «*SELECT * FROM pl_sum_activity*». Таблица представлена на рисунке 22.

	id	year	month	user_id	contract_id	work_min	work_need_min	completed
1	2	2021	1	3	1	500	7200	0.0694
2	1	2021	1	1	1	620	7200	0.0861
3	3	2019	1	3	1	600	8160	0.0735
4	4	2022	4	10	2	5760	10080	0.5714
5	5	2022	4	10	1	4800	10080	0.4762
6	6	2022	3	10	2	10500	10500	1.0000

Рисунок 22 – Записи таблицы «*pl_sum_activity*» после изменения активности

Информацию о деятельности пользователя за месяц можно получить запросом «*GET project-load/activity/user?info&year=[year]&month=[month]&userId=[userId]*», результаты запроса предложены на рисунке 23.

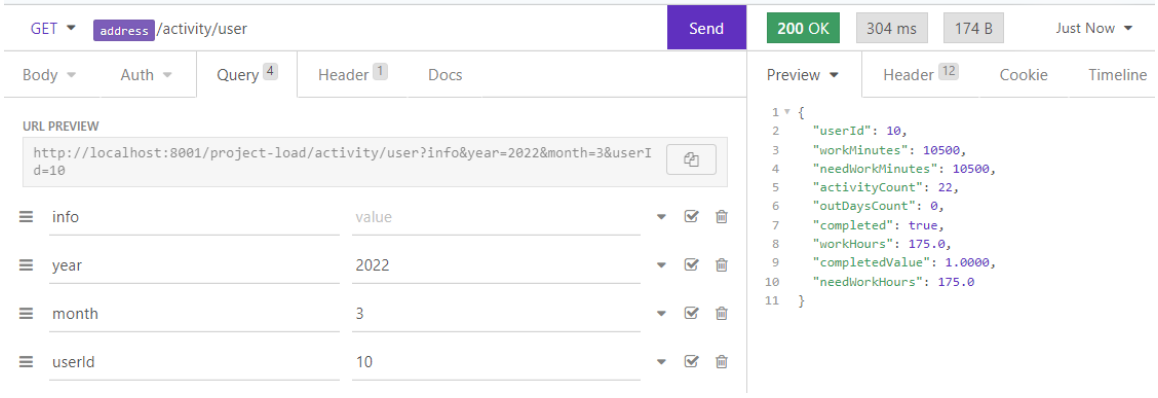


Рисунок 23 – Деятельность пользователя за месяц

Полное описание *API* для работы с активностями приведено в Приложении Г.

2 Безопасность жизнедеятельности

Воздействие персональных компьютеров на здоровье человека является важной проблемой современного общества.

Потенциально опасными или вредными факторами для здоровья сотрудника в процессе разработки программ могут являться:

- электрический ток;
- уровень шума и вибраций;
- повышенный уровень электромагнитных излучений;
- повышенный уровень статического электричества;
- пониженная ионизация воздуха;
- специфические физические нагрузки на отдельные мышцы рук, спины и шеи при общей гиподинамии;
- перенапряжение зрительных анализаторов.

2.1 Классификация и категорирование помещения, где производится работа

Для защиты от поражения человека электрическим током при устройстве помещений необходимо предусматривать меры обеспечения безопасности. С целью их оптимального выбора разработана классификация помещений. Все помещения делятся по степени поражения людей электрическим током на три класса: без повышенной опасности, с повышенной опасностью, особо опасные.

Помещения без повышенной опасности – это сухие, беспыльные помещения с нормальной температурой воздуха и с изолирующими полами, например, из дерева, то есть в которых отсутствуют условия, свойственные помещениям с повышенной опасностью и особо опасным.

Помещения с повышенной опасностью характеризуются наличием одного из следующих условий, создающих повышенную опасность:

– сырости, относительная влажность воздуха в помещении длительно превышает 75%;

– высокой температуры, температура воздуха длительно превышает +35° С;

– токопроводящей пыли, по условиям производства в помещениях выделяется токопроводящая технологическая пыль в таком количестве, что она оседает на проводах, проникает внутрь машин, аппаратов;

– токопроводящих полов: металлических, земляных, железобетонных, кирпичных;

– возможности одновременного прикосновения человека к имеющим соединение с землей металлоконструкциям зданий, технологическим аппаратам, механизмам с одной стороны, и к металлическим корпусам электрооборудования – с другой.

Помещения особо опасные характеризуются наличием одного из следующих условий, создающих особую опасность:

– особой сырости, относительная влажность воздуха близка к 100%;

– химически активной или органической среды;

– одновременного наличия двух и более условий, свойственных помещениям с повышенной опасностью.

Офисные помещения относятся к помещениям без повышенной опасности.

По пожарной и взрывопожарной опасности помещения производственного и складского назначения подразделяются на следующие категории:

– А, повышенная взрывопожароопасность;

– Б, взрывопожароопасность;

– В1-В4, пожароопасность;

– Г, умеренная пожароопасность;

– Д, пониженная пожароопасность.

Помещения иного назначения разделению на категории не подлежат. Офисные помещения можно отнести к категории Д или не категоризировать вовсе.

2.2 Санитарно-гигиеническая характеристика производства

Общие требования к организации рабочих мест регулируются Трудовым кодексом, санитарно-эпидемиологическими правилами и нормативами СанПиН, а также другими правовыми документами.

Окна в помещениях, где работают с компьютерами, должны быть преимущественно ориентированы на север и северо-восток. Оконные проемы необходимо оборудовать регулируемыми жалюзи, внешними козырьками [16]. Эксплуатация ПЭВМ в помещениях без естественного освещения допускается только при наличии расчетов, обосновывающих соответствие нормам естественного освещения и безопасность их деятельности для здоровья работающих. Естественное и искусственное освещение должно соответствовать требованиям действующей нормативной документации, а именно СанПиН 1.2.3685-21 [17].

Искусственное освещение, используемое в помещениях, где сотрудники непосредственно связаны с ПЭВМ, обязано обеспечивать правильную работу глаз, а также приближать к оптимальным условиям зрительное восприятие, какое бывает при естественном освещении.

Для этого выделяют следующие требования к освещению на рабочих местах, оборудованных ПЭВМ:

– искусственное освещение в помещениях для эксплуатации ПЭВМ должно быть распределено равномерно;

– освещенность рабочего места должна быть порядка 300 – 500 лк;

– для общего искусственного освещения следует использовать источники света с индексом цветопередачи Ra 85%;

– естественный свет должен падать преимущественно слева от рабочего стола. Нельзя располагать оборудованное дисплеем рабочее место таким образом, чтобы в поле зрения оператора не попадали окна или осветительные приборы, а также они не должны находиться непосредственно за спиной пользователя;

– яркость бликов на экране ПЭВМ не должна превышать 40 кд/м^2 и яркость потолка не должна превышать 200 кд/м^2;

– на рабочие места необходимо устанавливать локальные источники света, настольные лампы с регулируемой яркостью и изменением расположения плафона.

Содержание вредных химических веществ в производственных помещениях, в которых работа с использованием ПЭВМ является основной, не должно превышать предельно допустимых концентраций загрязняющих веществ в атмосферном воздухе населенных мест в соответствии с действующими гигиеническими нормативами. Уровень шума и вибрации не должен превышать допустимых значений для жилых и общественных зданий в соответствии с действующими санитарно-эпидемиологическими нормативами, а именно СанПиН 1.2.3685-21 [17]. Шумящее оборудование (печатающие устройства, серверы и т.п), уровни шума которого превышают нормативные, должно размещаться вне помещений с ПЭВМ.

Допустимые величины параметров микроклимата на рабочих местах в помещениях оцениваются в зависимости от категории работ по уровню энерготрат организма.

В производственных помещениях, в которых работа с использованием ПЭВМ является основной и связана с нервно-эмоциональным напряжением,

должны обеспечиваться оптимальные параметры микроклимата для категории работ Iа (до 139 Вт) и Iб (140 - 174 Вт) Допустимые величины параметров микроклимата указаны в таблице 13.

Таблица 13 – Допустимые величины параметров микроклимата помещения

Период года	Категория работ по уровню энерготрат, Вт	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с, не более
Холодный	Iа	22 - 24	21 - 25	60 - 40	0,1
	Iб	21 - 23	20 - 24	60 - 40	0,1
Теплый	Iа	23 - 25	22 - 26	60 - 40	0,1
	Iб	22 - 24	21 - 25	60 - 40	0,1

Площадь на одно рабочее место пользователей ПЭВМ с визуальным дисплейным терминалом на базе плоских дискретных экранов (жидкокристаллические, плазменные) – 4,5 м^2.

Также устанавливаются правила организации рабочего пространства:

– расстояние между рабочими столами с видеомониторами должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м;

– экран монитора должен находиться от глаз пользователя на расстоянии 600 - 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов;

– рабочее кресло должно быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой;

– сиденье и спинка стула должны быть покрыты не электризующимися полумягкими материалами.

В связи с напряженностью трудового процесса у пользователей ПЭВМ рекомендуется организовать рабочую смену путем чередования работ с использованием ПЭВМ и без него. Санитарными правилами устанавливается следующий внутрисменный режим труда и отдыха при восьмичасовой дневной рабочей смене: для разработчиков программ следует назначать регламентируемую паузу для отдыха длительностью 15 минут через каждый час работы за ПЭВМ. Во всех случаях, когда производственная ситуация не позволяет устанавливать регламентируемые перерывы, длительность непрерывной работы за монитором не должна превышать 4 часов.

Психологическая разгрузка работников пользователей ЭВМ должна проводиться в специально оборудованных помещениях (комнатах психологической разгрузки) во время регламентируемых перерывов или в конце рабочего дня.

Помимо этого, рекомендуется проводить упражнения на глаза, снижающие напряжение.

2.3 Электробезопасность

В соответствии с СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»[16] помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением в соответствии с техническими требованиями по эксплуатации электроустановок и вычислительной техники.

Рабочие места с ПЭВМ не следует размещать вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

Поскольку непосредственно на ПЭВМ должно подаваться стабилизированное электропитание подачу электроэнергии в компьютерные

помещения следует осуществлять от отдельного независимого источника питания.

В дальнейшем при эксплуатации ПЭВМ необходимо соблюдать следующие рекомендации [18]:

– постоянно контролировать надежность соединения контактов трехпроводных розеток;

– подключать дисплей (при наличии только двухпроводной однофазной сети) рекомендуется через согласующее устройство;

– не ставить системный блок в зоне повышенной влажности и повышенного содержания пыли, на пол, у ног оператора;

– во избежание поражения электрическим током запрещается прикасаться к задней панели системного блока;

– необходимо устанавливать ПЭВМ только на жестко закрепленной подставке, исключающей даже случайное сотрясение системного блока;

– не рекомендуется установка ПЭВМ и его клавиатуры на поверхности, накапливающие статическое электричество (органическое стекло и полированные лаковые поверхности).

2.4 Пожарная безопасность

Основными причинами возникновения пожаров в офисных помещениях являются неосторожное обращение с огнем и неисправность или неправильная эксплуатация электрооборудования.

Согласно статье 37 «Права и обязанности организаций в области пожарной безопасности» Федерального закона «О пожарной безопасности» [19] руководство организации обязано:

– проводить противопожарную пропаганду, а также обучать своих работников мерам пожарной безопасности;

– содержать в исправном состоянии системы и средства противопожарной защиты, включая первичные средства тушения пожаров, не допускать их использования не по назначению;

– оказывать содействие пожарной охране при тушении пожаров, установлении причин и условий их возникновения и развития, а также при выявлении лиц, виновных в нарушении требований пожарной безопасности и возникновении пожаров;

– обеспечивать доступ должностным лицам пожарной охраны при осуществлении ими служебных обязанностей на территории, в здания, сооружения и на иные объекты предприятий;

– незамедлительно сообщать в пожарную охрану о возникших пожарах, неисправностях имеющихся систем и средств противопожарной защиты, об изменении состояния дорог и проездов;

– содействовать деятельности добровольных пожарных.

Проведение противопожарного инструктажа включает в себя ознакомление работников организаций с:

– правилами содержания территории, зданий (сооружений) и помещений, в том числе эвакуационных путей, наружного и внутреннего водопровода, систем оповещения о пожаре и управления процессом эвакуации людей;

– требованиями пожарной безопасности, исходя из специфики пожарной опасности объекта;

– мероприятиями по обеспечению пожарной безопасности при эксплуатации зданий, оборудования, производстве пожароопасных работ;

– правилами применения открытого огня и проведения огневых работ;

– обязанностями и действиями работников при пожаре, правилами вызова пожарной охраны, правилами применения средств пожаротушения и установок пожарной автоматики.

По характеру и времени проведения противопожарный инструктаж подразделяется на: вводный, первичный на рабочем месте, повторный, внеплановый и целевой.

При огнетушении в офисных помещениях, где основной является работа с ПЭВМ, в основном применяются хладоновые огнетушители, составы которых не повреждают защищаемое оборудование и объекты. Запрещается применять порошковые и углекислотные огнетушители для тушения электрооборудования, находящегося под напряжением выше 1 кВ и 10 кВ соответственно. Воздушно-пенные, водные огнетушители не должны применяться для тушения оборудования, находящегося под электрическим напряжением.

2.5 Расчет общего искусственного освещения помещения, где работают на ПЭВМ

Расчет искусственного освещения заключается в определении числа и мощности источников света, обеспечивающих нормированную (с учетом коэффициентов запаса) освещенность рабочих поверхностей.

Расчет искусственного освещения будет проводиться для помещения общей площадью $S = 24$ м^2, длина $a = 6$ м, ширина $b = 4$ м, высота $h = 3$ м.

Для освещения помещения используется система общего освещения, так как она обеспечивает равномерность освещенности рабочих поверхностей и минимизирует бликование.

Расчет общего равномерного освещения будет производиться методом коэффициента использования светового потока, метод позволяет определить световой поток ламп при заданной освещенности рабочей поверхности, общем освещении с равномерным расположением светильников, с учетом отраженного стенами и потолком света.

Так как работа оператора ПЭВМ, также как и разработчика ПО, относится к категории работ высокой точности (наименьший размер объекта различения составляет от 0,30 мм до 0,50 мм), требуемый нормативный уровень освещенности рабочей поверхности для III разряда зрительных работ составит Е= от 300 лк до 500 лк при системе общего освещения [20].

Для освещения данного помещения выбран встраиваемый светодиодный светильник Грильято (световой поток 3756 лм, потребляемая мощность 32 Вт, индекс цветопередачи 85%, одна лампа). Световой поток светильника рассчитывается по формуле:

$$\Phi_{л} = \frac{E_{н} * S * z * k_{з}}{N * \eta}, \quad (1)$$

где $E_{н}$ – нормативный уровень освещенности рабочей поверхности [лк];

S – площадь помещения [м^2];

z – коэффициент неравномерности освещения;

$k_{з}$ – коэффициент запаса;

N – количество светильников;

η – коэффициент использования светового потока.

Нормативный уровень освещенности $E_{н}$ принят 400 лк. Площадь помещения – 24 м^2. Коэффициент неравномерности освещения z принимается равным 1,5. Коэффициент запаса $k_{з}$ учитывает запыленность помещения, снижение светового потока ламп в процессе эксплуатации. Для светодиодных ламп, используемых в офисных помещениях, значение коэффициента составляет 1.

Для определения коэффициента использования светового потока η находится индекс помещения:

$$i = \frac{A * B}{h_{р} * (A + B)},$$

где А – длина помещения в плане;

В – ширина помещения в плане;

$h_{\text{р}}$ – высота подвеса светильников над рабочей поверхностью.

Высота подвеса светильников составляет 2,2 м, длина помещения 6 м, ширина помещения 4 м. Индекс помещения тогда равен 1,09091.

Согласно таблице коэффициентов отражения коэффициент отражения потолка (плитка подвесного потолка белая) равен 70%, пола, покрытого светлой паркетной доской – 20%, стен бежевого цвета – 50%. Исходя из коэффициентов отражения потолка, стен и пола и индекса помещения коэффициент использования светового потока η будет равен 37%.

Для расчета количества светильников в помещении используется формула (2), она выводится из формулы (1).

$$N = \frac{E_{\text{н}} * S * z * k_{\text{з}}}{n * \eta * \Phi_{\text{л}}}, \quad (2)$$

где n – количество ламп в светильнике.

Тогда количество светильников в помещении равно:

$$N = 10{,}36 \approx 10.$$

Фактический уровень освещенности $E_{\text{факт}}$ тогда равен 386,033 лк, значение находится в пределах нормы.

Заключение

В ходе выполнения данной работы была изучена методология проектирования приложения с микросервисной архитектурой и приобретены навыки работы с фрейворком *Spring*. При реализации поставленной задачи были получены следующие результаты:

– разработана структура базы данных для СУБД *PostgreSQL;*

– разработан прикладной интерфейс для взаимодействия с остальными микросервисами приложения (*REST API*);

– разработано и отлажено приложение учета загрузки сотрудников.

Сервис реализует полный функционал для интеграции с остальными микросервисами внутрикорпоративного продукта.

В дальнейшем в проект стоит ввести интеграцию с часто используемыми трекерами задач: *Jira*, *Redmine*, *YouTrack*.

Список использованных источников

1. A2B – система управления и автоматизации бизнеса [Электронный ресурс] – URL: https://a2b.su (дата обращения: 26.04.2022 г)
2. OfficeTime – Time Tracking for Mac, PC, iPhone, iPad [Электронный ресурс] – URL: https://officetime.net (дата обращения: 27.04.2022 г)
3. Harvesrt Features [Электронный ресурс] – URL: https://www.getharvest.com (дата обращения: 28.04.2022 г)
4. Employee Computer Monitoring Software Kickidler Features [Электронный ресурс] – URL: https://www.kickidler.com (дата обращения: 29.04.2022 г)
5. Крис Р. Микросервисы. Паттерны разработки и рефакторинга – СПб: Питер, 2020 – 544 с.
6. Database Oracle [Электронный ресурс] – URL: https://www.oracle.com/ (дата обращения: 02.05.2022 г)
7. MS SQL Server 2019 [Электронный ресурс] – URL: https://www.microsoft.com/ (дата обращения: 02.05.2022 г)
8. Эрик Редмонд, Джим. Р. Уилсон. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. Под редакцией Жаклин Картер. : Пер. с англ. Слинкин А.А. – М.: ДМК Пресс, 2013 – 384 с.
9. The Go Programming Language [Электронный ресурс] – URL: https://go.dev (дата обращения: 03.05.2022 г)
10. Лутц М. Изучаем Python, 3-е издание.: Пер. с англ. – СПб.: Символ-Плюс, 2009. – 848 с
11. Д. Крокфорд Как устроен JavaScript – СПб.: Питер, 2019. – 304 с.
12. Кей С. Хорстманн Java. Библиотека профессионала, том 1. Основы. 11-е изд.: Пер. с англ. – СПб.: ООО "Диалектика", 2019. – 864 с.
13. Spring 5 для профессионалов.: Пер. с англ. – СПб.: ООО "Диалектика", 2019. – 1120 с.

14. Blade – Полный путеводитель [Электронный ресурс] – URL: https://javascopes.com/ (дата обращения: 03.05.2022 г)

15. Getting Started Dropwizard [Электронный ресурс] – URL: https://www.dropwizard.io/ (дата обращения: 04.05.2022 г)

16. СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к ПЭВМ и организации работы» [Электронный ресурс] – URL: http://www.consultant.ru/ (дата обращения: 25.05.2022 г)

17. СанПиН 1.2.3685-21 "Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания" [Электронный ресурс] – URL: https://www.opengost.ru/ (дата обращения: 25.05.2022 г)

18. Электробезопасность при работе оператора ПЭВМ: методические указания по выполнению в дипломных проектах и работах раздела «Безопасность объектов» /сост.:И.О. Протодьяконов, В.И. Сарже, О.И. Протодьяконова – СПб.: СПбГТУРП, 2013. –13 с.

19. Федеральный закон от 21.12.1994 № 69-ФЗ «О пожарной безопасности» (ред. от 13.07.2015) [Электронный ресурс] – URL: https://www.consultant.ru/ (дата обращения: 26.05.2022 г)

20. Свод правил СП 52.13330.2016 "Естественное и искусственное освещение" [Электронный ресурс] – URL: https://www.consultant.ru/ (дата обращения: 26.05.2022 г)

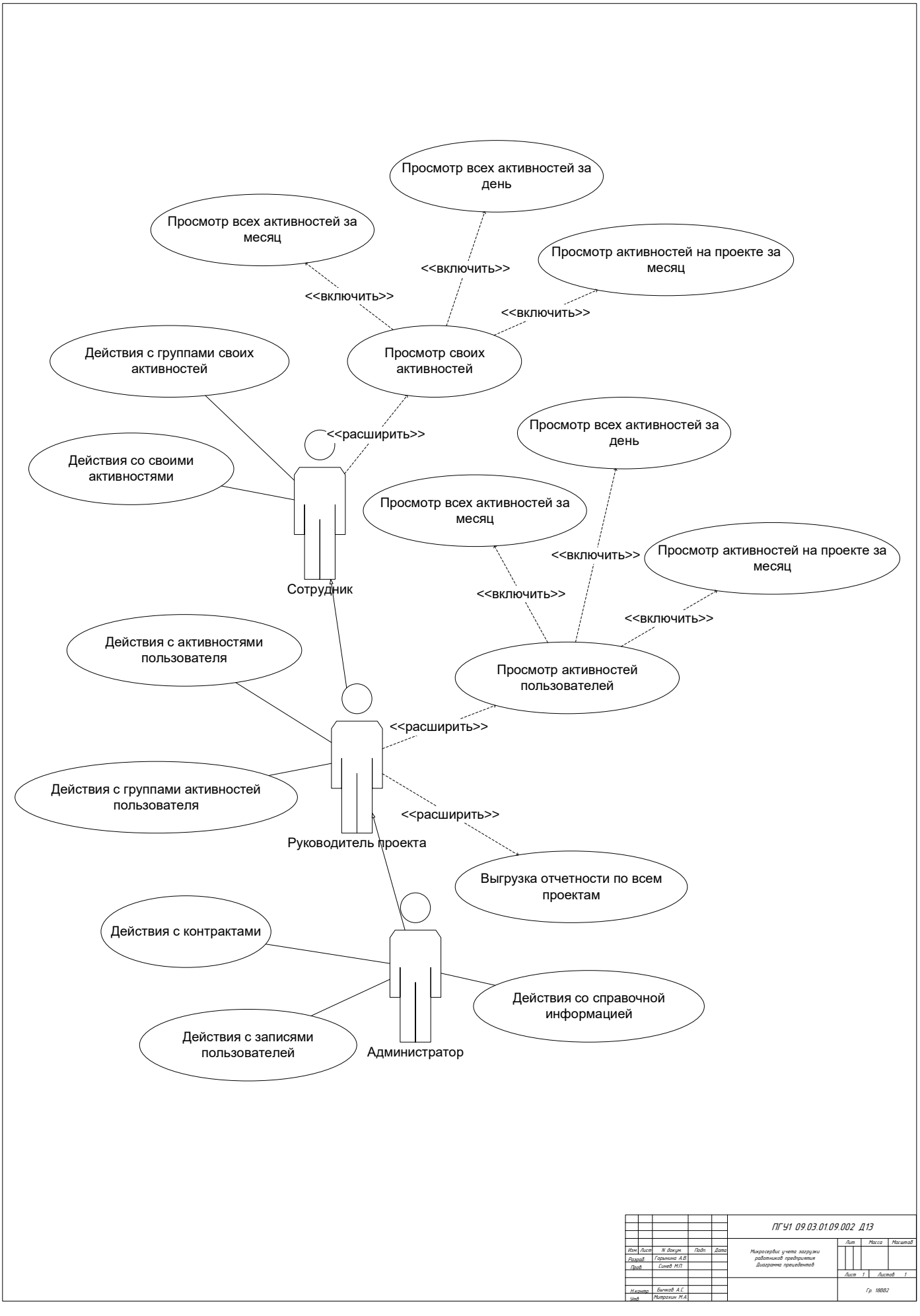
Приложение А

(обязательное)

Микросервис учета загрузки работников предприятия

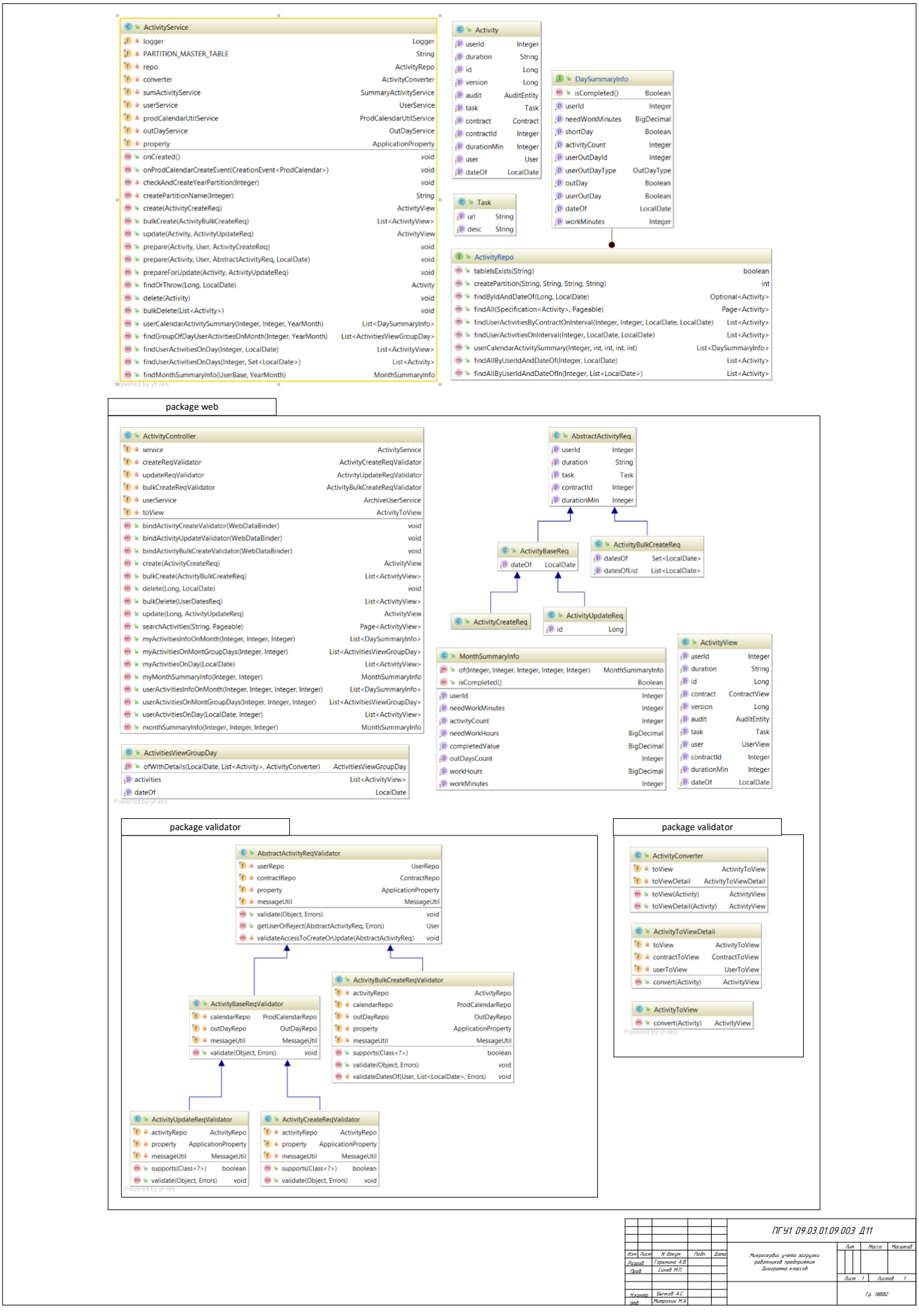
UML-диаграммы

Приложение А.1 – Диаграмма прецедентов



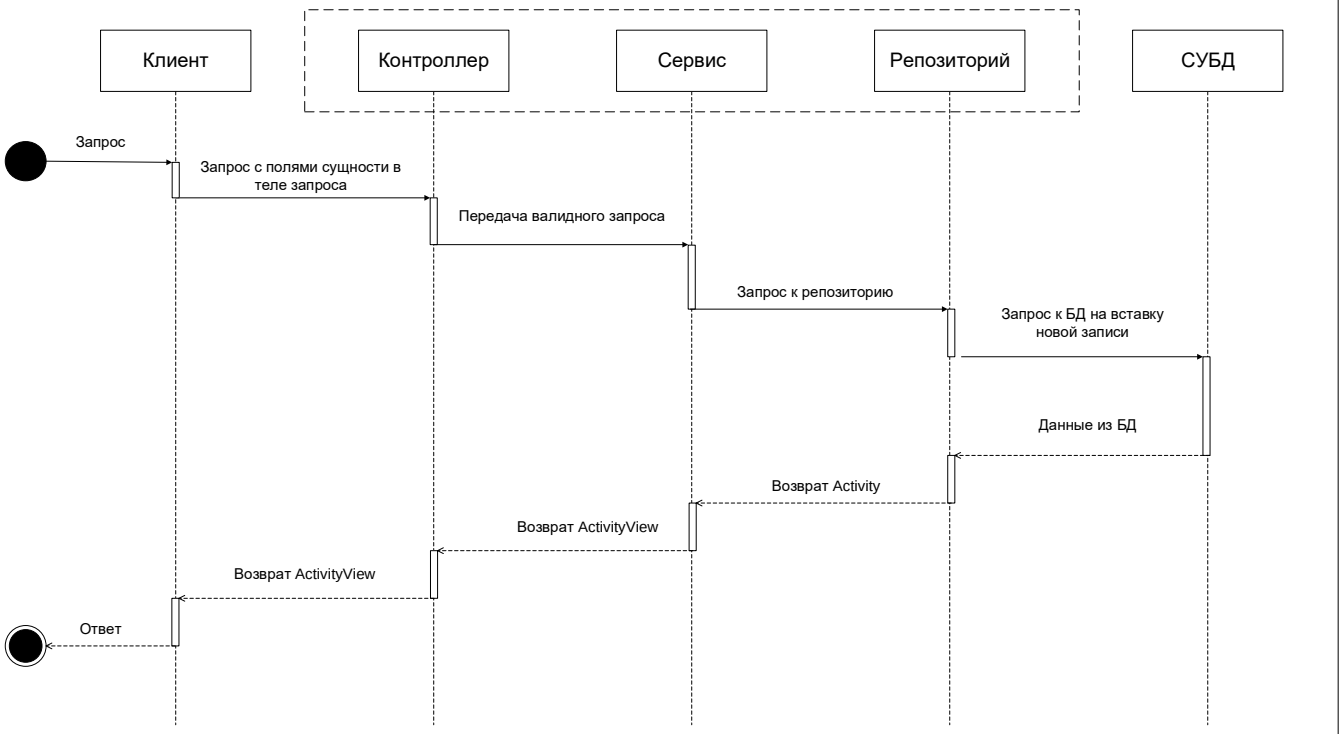
						ПГУ1 09.03.01.09.002 Д13	
Изм.	Лист	N докум.	Подп.	Дата			
Разраб.		Горынина А.В.			Микросервис учета загрузки работников предприятия Диаграмма прецедентов	Лит.	Масса / Масштаб
Пров.		Синев М.П.					
						Лист 1	Листов 1
Н.контр.		Бычков А.С.				Гр. 18882	
Утв.		Митрохин М.А.					

Приложение А.2 – Диаграмма классов



								ПГЧ1 09.03.01.09.003 Д11	
Изм.	Лист	N докум.	Подп.	Дата					
Разраб.		Гарнина А.В.			Микросервис учета загрузки работников предприятия Диаграмма классов		Лит.	Масса	Масштаб
Пров.		Синев М.П.							
							Лист 1	Листов 1	
Н.контр.		Бычков А.С.					Гр. 18882		
Утв.		Матрохин М.А.							

Приложение А.3 – Диаграмма последовательностей



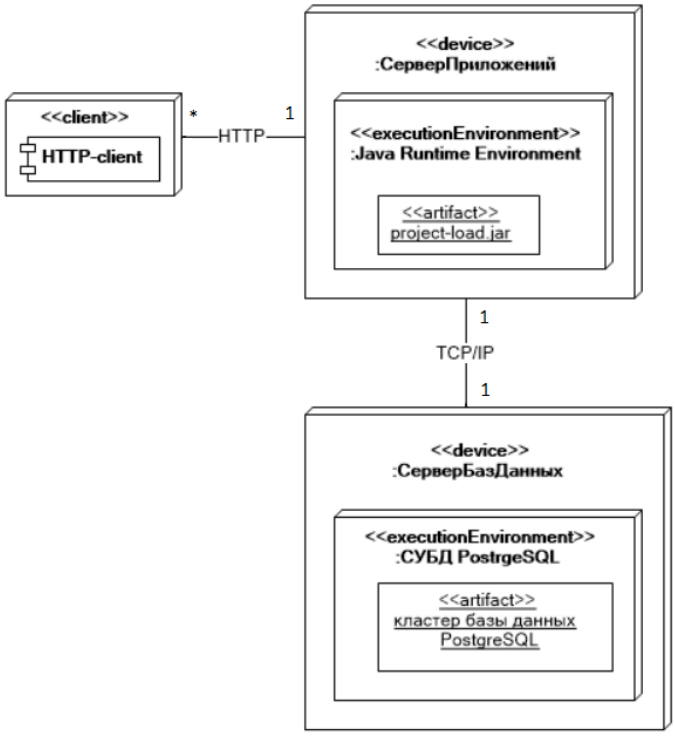
ПГУ1 09.03.01.09.004 Д14

Микросервис учета загрузки работников предприятия
Диаграмма последовательностей

Лит | Масса | Масштаб
Лист 1 | Листов 1

Гр. 18882

Приложение А.5 – Диаграмма развертывания



ПГУ1 09.03.01.09.005 Д19

Микросервис учета загрузки работников предприятия
Диаграмма развертывания

Лит | Масса | Масштаб
Лист 1 | Листов 1

Гр. 18882

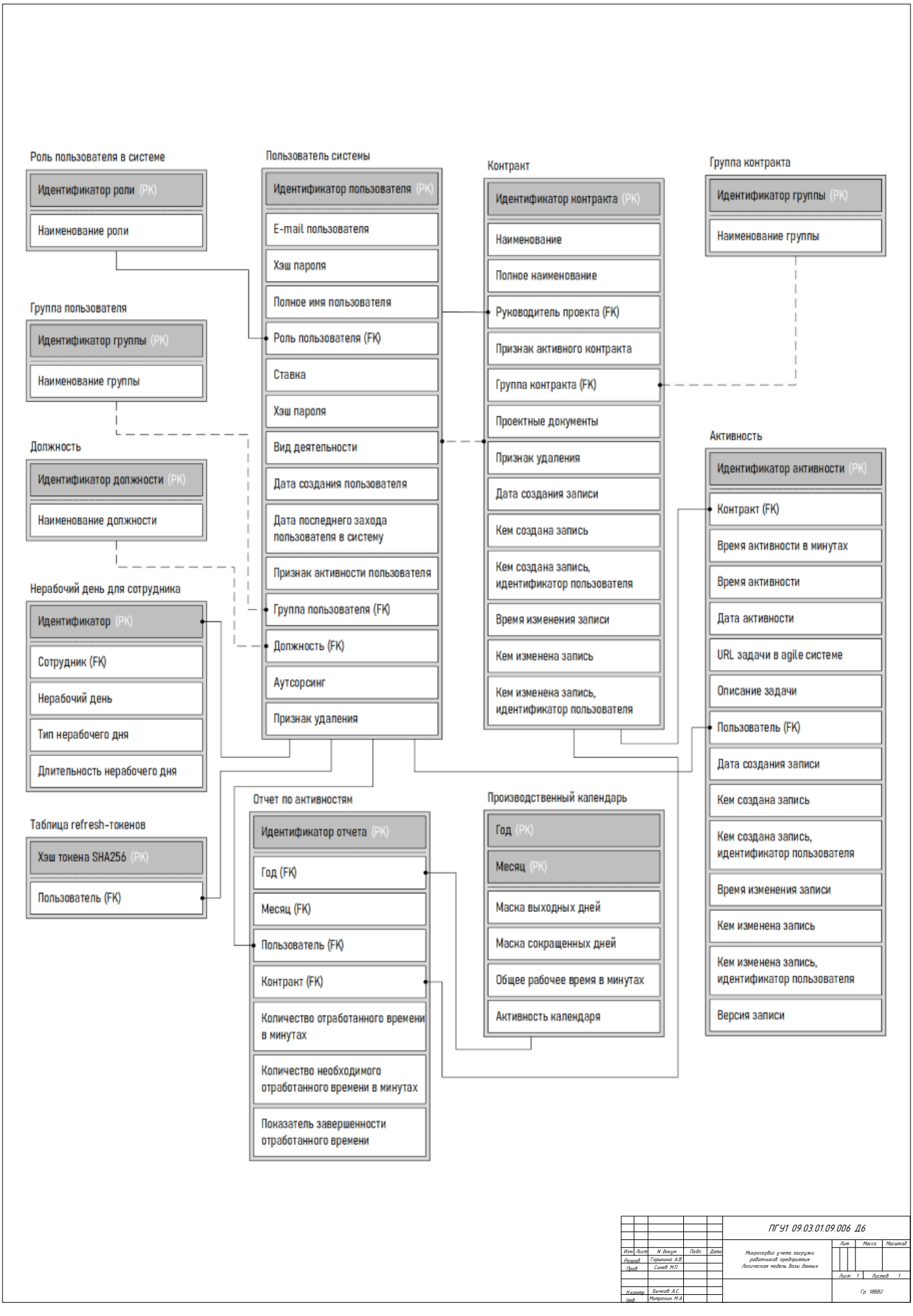
Приложение Б

(обязательное)

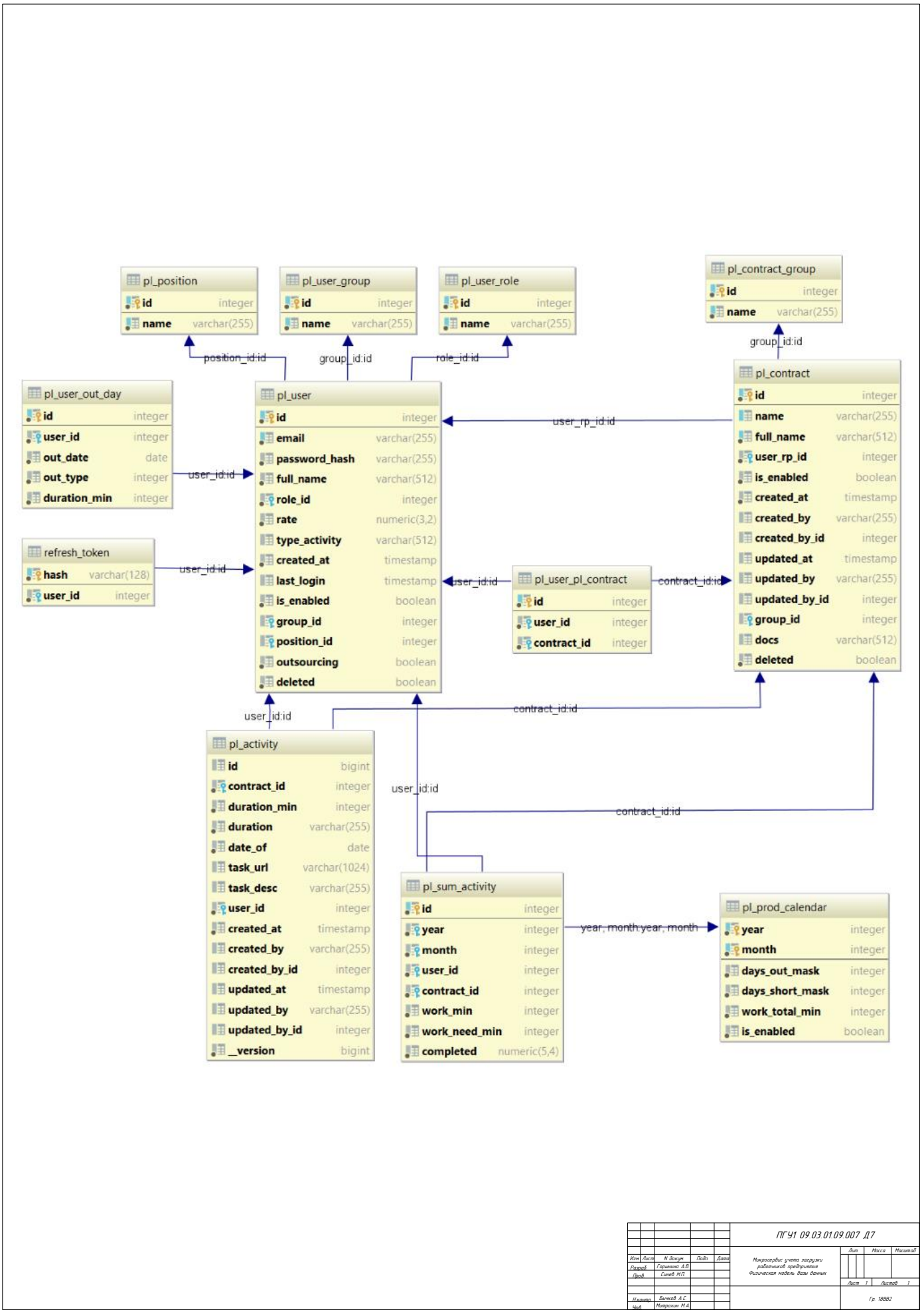
Микросервис учета загрузки работников предприятия

Модели базы данных

Приложение Б.1 – Логическая модель базы данных



Приложение Б.2 – Физическая модель базы данных



						ПГУ1 09.03.01.09.007 Д7	
Изм.	Лист	N докум.	Подп.	Дата			
Разраб.		Горынина А.В.			Микросервис учета загрузки работников предприятия Физическая модель базы данных	Лит.	Масса / Масштаб
Пров.		Сычев М.П.					
						Лист 1	Листов 1
Н.контр.		Бычков А.С.				Гр. 18ВВ2	
Утв.		Митрохин М.А.					

Приложение В

(обязательное)

Микросервис учета загрузки работников предприятия

SQL-файл базы данных

```
CREATE EXTENSION IF NOT EXISTS pgcrypto;
CREATE FUNCTION pl_activity_create_partition(master_table text,
partition_table text, check_from text, check_to text) RETURNS void
    LANGUAGE plpgsql
    AS $$
BEGIN
    EXECUTE format(
        'CREATE TABLE %2$I PARTITION OF %1$I FOR VALUES FROM
(''%3$s'') TO (''%4$s'');' ||
        'ALTER TABLE %2$s ADD CONSTRAINT pk_%2$I PRIMARY KEY(id);' ||
        'ALTER TABLE %2$s ALTER COLUMN id SET DEFAULT
nextval(''pl_activity_seq'');' ||
        'CREATE INDEX idx_%2$I_contract_id_id ON %2$I USING btree
(contract_id);' ||
        'CREATE INDEX idx_%2$I_date_of ON %2$I USING btree
(date_of);',
            master_table, partition_table, check_from, check_to);
END;
$$;

CREATE TABLE pl_user_role (
    id INTEGER,
    name VARCHAR(255) NOT NULL,

    CONSTRAINT pk_pl_user_role PRIMARY KEY (id),
    CONSTRAINT unq_pl_user_role_name UNIQUE (name)
);
COMMENT ON TABLE pl_user_role IS 'Роль пользователя в системе';
COMMENT ON COLUMN pl_user_role.id IS 'Идентификатор роли';
COMMENT ON COLUMN pl_user_role.name IS 'Наименование роли';

CREATE TABLE pl_position (
  id INTEGER,
  name VARCHAR(255) NOT NULL,

  CONSTRAINT pk_pl_position PRIMARY KEY (id),
  CONSTRAINT unq_pl_position_name UNIQUE (name)
);
COMMENT ON TABLE pl_position IS 'Должность';
COMMENT ON COLUMN pl_position.id IS 'Идентификатор должности';
COMMENT ON COLUMN pl_position.name IS 'Наименование должности';

CREATE SEQUENCE pl_position_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 2147483648
  START 1
  CACHE 1;
  CREATE TABLE pl_user_group (
  id INTEGER,
  name VARCHAR(255) NOT NULL,

  CONSTRAINT pk_pl_user_group PRIMARY KEY (id),
  CONSTRAINT unq_pl_user_group_name UNIQUE (name)
);
```

```
COMMENT ON TABLE pl_user_group IS 'Группа пользователя (структурная
единица организации)';
COMMENT ON COLUMN pl_user_group.id IS 'Идентификатор группы';
COMMENT ON COLUMN pl_user_group.name IS 'Наименование группы';

CREATE SEQUENCE pl_user_group_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 2147483648
  START 1
  CACHE 1;

CREATE TABLE pl_user (
    id INTEGER,
    email VARCHAR(255) NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    full_name VARCHAR(512) NOT NULL,
    role_id INTEGER NOT NULL,
  group_id INTEGER,
    position_id INTEGER,
    rate NUMERIC(3, 2) NOT NULL,
    type_activity VARCHAR (512),
    created_at TIMESTAMP NOT NULL,
    last_login TIMESTAMP,
    is_enabled BOOLEAN DEFAULT true NOT NULL,
    outsourcing BOOLEAN NOT NULL DEFAULT false,
    deleted BOOLEAN NOT NULL DEFAULT FALSE,
    CONSTRAINT pk_pl_user PRIMARY KEY (id),
    CONSTRAINT fk_pl_user_pl_user_role FOREIGN KEY (role_id)
        REFERENCES pl_user_role (id)
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT fk_pl_user_pl_position FOREIGN KEY (position_id)
        REFERENCES pl_position (id)
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT fk_pl_user_pl_user_group FOREIGN KEY (group_id)
        REFERENCES pl_user_group(id)
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT unq_pl_user_email UNIQUE (email)
);
COMMENT ON TABLE pl_user IS 'Пользователь системы';
COMMENT ON COLUMN pl_user.id IS 'Идентификатор пользователя';
COMMENT ON COLUMN pl_user.email IS 'E-mail пользователя';
COMMENT ON COLUMN pl_user.password_hash IS 'Хэш пароля';
COMMENT ON COLUMN pl_user.full_name IS 'Полное имя пользователя';
COMMENT ON COLUMN pl_user.role_id IS 'Роль пользователя';
COMMENT ON COLUMN pl_user.group_id IS 'Группа пользователя';
COMMENT ON COLUMN pl_user.position_id IS 'Должность';
COMMENT ON COLUMN pl_user.rate IS 'Ставка';
COMMENT ON COLUMN pl_user.type_activity IS 'Вид деятельности';
COMMENT ON COLUMN pl_user.created_at IS 'Дата создания пользователя';
COMMENT ON COLUMN pl_user.last_login IS 'Дата последнего захода
пользователя в систему';
COMMENT ON COLUMN pl_user.is_enabled IS 'Признак активности
пользователя';
COMMENT ON COLUMN pl_user.outsourcing IS 'Аутсорсинг';
```

```
COMMENT ON COLUMN pl_user.deleted IS 'Признак удаления';

CREATE SEQUENCE pl_user_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 2147483648
  START 1
  CACHE 1;

CREATE TABLE refresh_token
(
  hash VARCHAR(128),
  user_id INTEGER NOT NULL,
  CONSTRAINT pk_refresh_token_hash PRIMARY KEY (hash),
  CONSTRAINT fk_refresh_token_pl_user FOREIGN KEY (user_id)
    REFERENCES pl_user (id)
    ON UPDATE NO ACTION ON DELETE CASCADE
);
COMMENT ON TABLE refresh_token IS 'Таблица refresh-токенов';
COMMENT ON COLUMN refresh_token.hash IS 'Хэш токена SHA256';
COMMENT ON COLUMN refresh_token.user_id IS 'Пользователь';

CREATE TABLE pl_contract_group (
  id INTEGER,
  name VARCHAR(255) NOT NULL,
  CONSTRAINT pk_pl_contract_group PRIMARY KEY (id),
  CONSTRAINT unq_pl_contract_group_name UNIQUE (name)
);
COMMENT ON TABLE pl_contract_group IS 'Группа контракта';
COMMENT ON COLUMN pl_contract_group.id IS 'Идентификатор группы';
COMMENT ON COLUMN pl_contract_group.name IS 'Наименование группы';

CREATE SEQUENCE pl_contract_group_seq
    INCREMENT 1
    MINVALUE 1
    MAXVALUE 2147483648
    START 1
    CACHE 1;

CREATE TABLE pl_contract (
    id INTEGER,
    name VARCHAR(255),
    full_name VARCHAR(512) NOT NULL,
    user_rp_id INTEGER NOT NULL,
    is_enabled BOOLEAN DEFAULT true NOT NULL,
  group_id INTEGER,
    docs VARCHAR(512),
    deleted BOOLEAN NOT NULL DEFAULT FALSE,
    -- audit
    created_at TIMESTAMP NOT NULL,
    created_by VARCHAR(255),
    created_by_id INTEGER,
    updated_at TIMESTAMP,
    updated_by VARCHAR(255),
    updated_by_id INTEGER,
```

```
    CONSTRAINT pk_pl_contract PRIMARY KEY (id),
    CONSTRAINT fk_pl_contract_pl_user_rp FOREIGN KEY (user_rp_id)
        REFERENCES pl_user (id)
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT fk_pl_contract_pl_contract_group FOREIGN KEY (group_id)
        REFERENCES pl_contract_group (id)
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT unq_pl_contract_full_name_name UNIQUE (full_name, name)
);
COMMENT ON TABLE pl_contract IS 'Контракт';
COMMENT ON COLUMN pl_contract.id IS 'Идентификатор контракта';
COMMENT ON COLUMN pl_contract.name IS 'Наименование';
COMMENT ON COLUMN pl_contract.full_name IS 'Полное наименование';
COMMENT ON COLUMN pl_contract.user_rp_id IS 'Руководитель проекта';
COMMENT ON COLUMN pl_contract.is_enabled IS 'Признак активного
контракта';
COMMENT ON COLUMN pl_contract.group_id IS 'Группа контракта';
COMMENT ON COLUMN pl_contract.docs IS 'Проектные документы (ссылка)';
COMMENT ON COLUMN pl_contract.deleted IS 'Признак удаления';

CREATE SEQUENCE pl_contract_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 2147483648
  START 1
  CACHE 1;

CREATE TABLE pl_activity (
    id BIGINT,
    contract_id INTEGER NOT NULL,
    duration_min INTEGER NOT NULL,
    duration VARCHAR(255) NOT NULL,
    date_of DATE NOT NULL,
    task_url VARCHAR (1024),
    task_desc VARCHAR (255),
    user_id INTEGER NOT NULL,
     -- audit
    created_at TIMESTAMP NOT NULL,
    created_by VARCHAR(255),
    created_by_id INTEGER,
    updated_at TIMESTAMP,
    updated_by VARCHAR(255),
    updated_by_id INTEGER,
    __version BIGINT NOT NULL,
    CONSTRAINT fk_pl_activity_pl_contract FOREIGN KEY (contract_id)
        REFERENCES pl_contract (id)
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT fk_pl_activity_pl_user FOREIGN KEY (user_id)
        REFERENCES pl_user (id)
        ON UPDATE NO ACTION ON DELETE NO ACTION
) PARTITION BY RANGE (date_of);
COMMENT ON TABLE pl_activity IS 'Активность';
COMMENT ON COLUMN pl_activity.id IS 'Идентификатор активности';
COMMENT ON COLUMN pl_activity.contract_id IS 'Контракт';
```

```
COMMENT ON COLUMN pl_activity.duration_min IS 'Время активности в
минутах';
COMMENT ON COLUMN pl_activity.date_of IS 'Дата активности';
COMMENT ON COLUMN pl_activity.task_url IS 'URL задачи в agile
системе';
COMMENT ON COLUMN pl_activity.task_desc IS 'Описание задачи';
COMMENT ON COLUMN pl_activity.user_id IS 'Пользователь';

CREATE SEQUENCE pl_activity_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 9223372036854775807
  START 1
  CACHE 1
  OWNED BY pl_activity.id;

CREATE TABLE pl_prod_calendar (
    year INTEGER,
    month INTEGER,
    days_out_mask INTEGER NOT NULL,
    days_short_mask INTEGER NOT NULL,
    work_total_min INTEGER NOT NULL,
    is_enabled BOOLEAN NOT NULL DEFAULT true,
    CONSTRAINT pk_pl_prod_calendar PRIMARY KEY (year, month)
);
COMMENT ON TABLE pl_prod_calendar IS 'Производственный календарь';
COMMENT ON COLUMN pl_prod_calendar.year IS 'Год';
COMMENT ON COLUMN pl_prod_calendar.month IS 'Месяц';
COMMENT ON COLUMN pl_prod_calendar.days_out_mask IS 'Маска выходных
дней';
COMMENT ON COLUMN pl_prod_calendar.days_short_mask IS 'Маска
сокращенных дней';
COMMENT ON COLUMN pl_prod_calendar.work_total_min IS 'Общее рабочее
время в минутах';
COMMENT ON COLUMN pl_prod_calendar.is_enabled IS 'Активность
календаря';

CREATE TABLE pl_sum_activity (
    id INTEGER,
    year INTEGER NOT NULL,
    month INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    contract_id INTEGER NOT NULL,
    work_min INTEGER NOT NULL,
    work_need_min INTEGER NOT NULL,
    completed NUMERIC (5, 4) NOT NULL,
    CONSTRAINT pk_pl_sum_activity PRIMARY KEY (id),
    CONSTRAINT fk_pl_sum_activity_pl_prod_calendar FOREIGN KEY (year,
month)
        REFERENCES pl_prod_calendar (year, month)
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT fk_pl_sum_activity_pl_contract FOREIGN KEY
(contract_id)
        REFERENCES pl_contract (id)
        ON UPDATE NO ACTION ON DELETE NO ACTION,
```

```
    CONSTRAINT fk_pl_sum_activity_pl_user FOREIGN KEY (user_id)
        REFERENCES pl_user (id)
        ON UPDATE NO ACTION ON DELETE NO ACTION
);
COMMENT ON TABLE pl_sum_activity IS 'Таблица отчета по активностям';
COMMENT ON COLUMN pl_sum_activity.id IS 'Идентификатор  отчета';
COMMENT ON COLUMN pl_sum_activity.year IS 'Год';
COMMENT ON COLUMN pl_sum_activity.month IS 'Месяц';
COMMENT ON COLUMN pl_sum_activity.user_id IS 'Пользователь';
COMMENT ON COLUMN pl_sum_activity.contract_id IS 'Контракт';
COMMENT ON COLUMN pl_sum_activity.work_min IS 'Количество
отработанного времени в минутах';
COMMENT ON COLUMN pl_sum_activity.work_need_min IS 'Количество
необходимого отработанного времени в минутах';
COMMENT ON COLUMN pl_sum_activity.completed IS 'Показатель
завершенности отработанного времени';

CREATE SEQUENCE pl_sum_activity_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 2147483648
  START 1
  CACHE 1;

  CREATE TABLE pl_user_out_day (
  id INTEGER,
  user_id INTEGER NOT NULL,
  out_date DATE NOT NULL,
  out_type INTEGER NOT NULL,
  duration_min INTEGER NOT NULL,
  CONSTRAINT pk_pl_user_out_day PRIMARY KEY (id),
  CONSTRAINT fk_pl_user_out_day_pl_user FOREIGN KEY (user_id)
    REFERENCES pl_user (id)
    ON UPDATE NO ACTION ON DELETE NO ACTION
);
COMMENT ON TABLE pl_user_out_day IS 'Нерабочий день для сотрудника';
COMMENT ON COLUMN pl_user_out_day.id IS 'Идентификатор';
COMMENT ON COLUMN pl_user_out_day.user_id IS 'Сотрудник';
COMMENT ON COLUMN pl_user_out_day.out_date IS 'Нерабочий день';
COMMENT ON COLUMN pl_user_out_day.out_type IS 'Тип нерабочего дня
(отпуск, больничный, отгул)';
COMMENT ON COLUMN pl_user_out_day.duration_min IS 'Длительность
нерабочего дня';

CREATE SEQUENCE pl_user_out_day_seq
    INCREMENT 1
    MINVALUE 1
    MAXVALUE 2147483648
    START 1
    CACHE 1;

CREATE TABLE pl_user_pl_contract (
  id SERIAL,
  user_id INTEGER NOT NULL,
  contract_id INTEGER NOT NULL,
```

```
  CONSTRAINT pk_pl_user_pl_contract PRIMARY KEY (id),
  CONSTRAINT fk_pl_user_pl_contract_pl_user FOREIGN KEY (user_id)
    REFERENCES pl_user(id)
    ON UPDATE NO ACTION ON DELETE CASCADE,
  CONSTRAINT fk_pl_user_pl_contract_pl_contract FOREIGN KEY
(contract_id)
    REFERENCES pl_contract(id)
    ON UPDATE NO ACTION ON DELETE CASCADE
);
COMMENT ON TABLE pl_user_pl_contract IS 'Пользователь - Контракт';
COMMENT ON COLUMN pl_user_pl_contract.id IS 'Идентификатор';
COMMENT ON COLUMN pl_user_pl_contract.user_id IS 'Пользователь';
COMMENT ON COLUMN pl_user_pl_contract.contract_id IS 'Контракт';
```

Приложение Г

(обязательное)

Микросервис учета загрузки работников предприятия

Описание *REST API* для работы с активностями

Запрос «Создание активности»

```
Метод: POST
URI: /project-load/activity
Тело запроса:
                {
                  "contractId": int,
                  "duration": "string",
                  "dateOf": "date",
                  "task": {
                    "url": "string",
                    "desc": "string"
                  },
                  "userId": int
                }
Ответ:
 201     Created
         {
           "id": int,
           "contract": null,
           "contractId": int,
           "duration": "string",
           "durationMin": int,
           "dateOf": date,
           "task": {
             "url": "string",
             "desc": "string"
           },
           "audit": {
             "createdAt": timestamp,
             "createdBy": "string",
             "createdById": int,
             "updatedAt": null,
             "updatedBy": null,
             "updatedById": null
           },
           "version": 0,
           "user": null,
           "userId": int
         }
 400     Bad Request
         { "errorMessage": "string" }
 401     Not authorized
         { "errorMessage": "string" }
 403     Forbidden
         { "errorMessage": "string" }
 500     Server error
         { "errorMessage": "string" }
```

Запрос «Обновление активности»

```
Метод: PUT
URI: /project-load/activity/[activityId]
Тело запроса:
                {
```

```
                  "id": int,
                  "contractId": int,
                  "duration": "string",
                  "dateOf": "date",
                  "task": {
                    "url": "string",
                    "desc": "string"
                  },
                  "userId": int
                }
Ответ:
 200     OK
         {
           "id": int,
           "contract": null,
           "contractId": int,
           "duration": "string",
           "durationMin": int,
           "dateOf": date,
           "task": {
             "url": "string",
             "desc": "string"
           },
           "audit": {
             "createdAt": timestamp,
             "createdBy": "string",
             "createdById": int,
             "updatedAt": timestamp,
             "updatedBy": "string",
             "updatedById": int
           },
           "version": int,
           "user": null,
           "userId": int
         }
 400     Bad Request
         { "errorMessage": "string" }
 401     Not Authorized
         { "errorMessage": "string" }
 403     Forbidden
         { "errorMessage": "string" }
 500     Server Error
         { "errorMessage": "string" }
```

Запрос «Удаление активности»

```
Метод: DELETE
URI: /project-load/activity/[activityId]
Ответ:
 204     No Content
 400     Bad Request
         { "errorMessage": "string" }
 401     Not Authorized
         { "errorMessage": "string" }
 404     Not Found
```

```
        { "errorMessage": "string" }
 500    Server Error
        { "errorMessage": "string" }
```

Запрос «Создание группы активностей»

```
Метод: POST
URI: /project-load/activity/bulk
Тело запроса:
                {
                  "contractId": int,
                  "duration": "string",
                  "datesOf": [date, ..., date],
                  "task": {
                    "url": "string",
                    "desc": "string"
                  },
                  "userId": int
                }
Ответ:
 200    OK
        [
          {
            "id": int,
            "contract": null,
            "contractId": int,
            "duration": "string",
            "durationMin": int,
            "dateOf": date,
            "task": {
              "url": "string",
              "desc": "string"
            },
            "audit": {
              "createdAt": timestamp,
              "createdBy": "string",
              "createdById": int,
              "updatedAt": null,
              "updatedBy": null,
              "updatedById": null
            },
            "version": 0,
            "user": null,
            "userId": int
          },
        ...,
          {
            "id": int,
            "contract": null,
            "contractId": int,
            "duration": "string",
            "durationMin": int,
            "dateOf": date,
            "task": {
              "url": "string",
```

```
            "desc": "string"
          },
          "audit": {
            "createdAt": timestamp,
            "createdBy": "string",
            "createdById": int,
            "updatedAt": null,
            "updatedBy": null,
            "updatedById": null
          },
          "version": 0,
          "user": null,
          "userId": int
        }
      ]
 400  Bad Request
      { "errorMessage": "string" }
 401  Not authorized
      { "errorMessage": "string" }
 403  Forbidden
      { "errorMessage": "string" }
 500  Server error
      { "errorMessage": "string" }
```

Запрос «Удаление группы активностей»

```
Метод: DELETE
URI: /project-load/activity/bulk
Тело запроса:
            {
              "user":
              {
                "id":3
              },
              "datesOf": [date, ..., date]
            }
Ответ:
 200  OK
      [
        {
          "id": int,
          "contract": null,
          "contractId": int,
          "duration": "string",
          "durationMin": int,
          "dateOf": date,
          "task": {
            "url": "string",
            "desc": "string"
          },
          "audit": {
            "createdAt": timestamp,
            "createdBy": "string",
            "createdById": int,
            "updatedAt": timestamp,
```

```
            "updatedBy": "string",
            "updatedById": int
          },
          "version": int,
          "user": null,
          "userId": int
        },
      ...,
        {
          "id": int,
          "contract": null,
          "contractId": int,
          "duration": "string",
          "durationMin": int,
          "dateOf": date,
          "task": {
            "url": "string",
            "desc": "string"
          },
          "audit": {
            "createdAt": timestamp,
            "createdBy": "string",
            "createdById": int,
            "updatedAt": timestamp,
            "updatedBy": "string",
            "updatedById": int
          },
          "version": int,
          "user": null,
          "userId": int
        }
      ]
 401  Not Authorized
      { "errorMessage": "string" }
 500  Server Error
      { "errorMessage": "string" }
```

Запрос «Поиск активностей»

```
Метод: GET
URI:/project-load/activity/?search=[searchTerm]
Ответ:
 200  OK
      {
        "content":[
        {
          "id": int,
          "contract": null,
          "contractId": int,
          "duration": "string",
          "durationMin": int,
          "dateOf": date,
          "task": {
            "url": "string",
            "desc": "string"
```

```
      },
      "audit": {
        "createdAt": timestamp,
        "createdBy": "string",
        "createdById": int,
        "updatedAt": timestamp,
        "updatedBy": "string",
        "updatedById": int
      },
      "version": int,
      "user": null,
      "userId": int
    },
    ...,
    {
      "id": int,
      "contract": null,
      "contractId": int,
      "duration": "string",
      "durationMin": int,
      "dateOf": date,
      "task": {
        "url": "string",
        "desc": "string"
      },
      "audit": {
        "createdAt": timestamp,
        "createdBy": "string",
        "createdById": int,
        "updatedAt": timestamp,
        "updatedBy": "string",
        "updatedById": int
      },
      "version": int,
      "user": null,
      "userId": int
    }
  ],
  "pageable": {
      "sort": {
        "sorted": boolean,
        "unsorted": boolean,
        "empty": boolean
      },
      "offset": int,
      "pageSize": int,
      "pageNumber": int,
      "unpaged": boolean,
      "paged": boolean
    },
    "last": boolean,
    "totalElements": int,
    "totalPages": int,
    "number": int,
    "size": int,
```

```
          "sort": {
            "sorted": boolean,
            "unsorted": boolean,
            "empty": boolean
          },
          "first": boolean,
          "numberOfElements": int,
          "empty": boolean
        }
 400    Bad Request
        { "errorMessage": "string" }
 401    Not Authorized
        { "errorMessage": "string" }
 500    Server Error
        { "errorMessage": "string" }
```

Запрос «Получение моего календаря со сводной информацией по активностям»

```
Метод: GET
URI: /project-load/activity/my?calendar&year=[year]& month=[month]
Ответ:
 200    OK
        [
          {
            "completed": boolean,
            "dateOf": date,
            "userId": int,
            "workMinutes": int,
            "needWorkMinutes": boolean
            "activityCount": int,
            "shortDay": boolean,
            "userOutDayId": int,
            "userOutDay": boolean,
            "outDay": boolean,
            "userOutDayType": null
          },
          ...,
          {
            "completed": boolean,
            "dateOf": date,
            "userId": int,
            "workMinutes": int,
            "needWorkMinutes": boolean
            "activityCount": int,
            "shortDay": boolean,
            "userOutDayId": int,
            "userOutDay": boolean,
            "outDay": boolean,
            "userOutDayType": null
          }
        ]
 400    Bad Request
        { "errorMessage": "string" }
```

```
401     Not Authorized
        { "errorMessage": "string" }
500     Server Error
        { "errorMessage": "string" }
```

Запрос «Получение моих активностей за месяц с групировкой по дням»

```
Метод: GET
URI: /project-load/activity/my?list&year=[year]& month=[month]
Ответ:
200     OK
        [
          {
            "dateOf": date,
            "activities": [
              {
                "id": int,
                "contract": {
                  "id": int,
                  "name": "string",
                  "fullName": "string",
                  "userRpId": int,
                  "userRp": null,
                  "groupId": int,
                  "group": null,
                  "docs": "string",
                  "audit": {
                    "createdAt": timestamp,
                    "createdBy": "string",
                    "createdById": int,
                    "updatedAt": timestamp,
                    "updatedBy": "string",
                    "updatedById": int
                  },
                  "enabled": boolean
                },
                "contractId": int,
                "duration": "string",
                "durationMin": int,
                "dateOf": date,
                "task": {
                  "url": "string",
                  "desc": "string"
                },
                "audit": {
                  "createdAt": timestamp,
                  "createdBy": "string",
                  "createdById": int,
                  "updatedAt": timestamp,
                  "updatedBy": "string",
                  "updatedById": int
                },
                "version": int,
                "user": {
                  "id": int,
```

```
      "email": "string",
      "fullName": "string",
      "roleId": "string",
      "createdAt": timestamp,
      "lastLogin": timestamp,
      "rate": float,
      "position": null,
      "positionId": int,
      "typeActivity": "string",
      "outsourcing": boolean,
      "groupId": int,
      "group": null,
      "enabled": boolean
    },
    "userId": int
  },
  ...,
  {
    "id": int,
    "contract": {
      "id": int,
      "name": "string",
      "fullName": "string",
      "userRpId": int,
      "userRp": null,
      "groupId": int,
      "group": null,
      "docs": "string",
      "audit": {
        "createdAt": timestamp,
        "createdBy": "string",
        "createdById": int,
        "updatedAt": timestamp,
        "updatedBy": "string",
        "updatedById": int
      },
      "enabled": boolean
    },
    "contractId": int,
    "duration": "string",
    "durationMin": int,
    "dateOf": date,
    "task": {
      "url": "string",
      "desc": "string"
    },
    "audit": {
      "createdAt": timestamp,
      "createdBy": "string",
      "createdById": int,
      "updatedAt": timestamp,
      "updatedBy": "string",
      "updatedById": int
    },
    "version": int,
```

```
        "user": {
          "id": int,
          "email": "string",
          "fullName": "string",
          "roleId": "string",
          "createdAt": timestamp,
          "lastLogin": timestamp,
          "rate": float,
          "position": null,
          "positionId": int,
          "typeActivity": "string",
          "outsourcing": boolean,
          "groupId": int,
          "group": null,
          "enabled": boolean
        },
        "userId": int
      }
    ]
  },
  ...,
  {
    "dateOf": date,
    "activities": [
      {
        "id": int,
        "contract": {
          "id": int,
          "name": "string",
          "fullName": "string",
          "userRpId": int,
          "userRp": null,
          "groupId": int,
          "group": null,
          "docs": "string",
          "audit": {
            "createdAt": timestamp,
            "createdBy": "string",
            "createdById": int,
            "updatedAt": timestamp,
            "updatedBy": "string",
            "updatedById": int
          },
          "enabled": boolean
        },
        "contractId": int,
        "duration": "string",
        "durationMin": int,
        "dateOf": date,
        "task": {
          "url": "string",
          "desc": "string"
        },
        "audit": {
          "createdAt": timestamp,
```

```
      "createdBy": "string",
      "createdById": int,
      "updatedAt": timestamp,
      "updatedBy": "string",
      "updatedById": int
    },
    "version": int,
    "user": {
      "id": int,
      "email": "string",
      "fullName": "string",
      "roleId": "string",
      "createdAt": timestamp,
      "lastLogin": timestamp,
      "rate": float,
      "position": null,
      "positionId": int,
      "typeActivity": "string",
      "outsourcing": boolean,
      "groupId": int,
      "group": null,
      "enabled": boolean
    },
    "userId": int
  },
  ...,
  {
    "id": int,
    "contract": {
      "id": int,
      "name": "string",
      "fullName": "string",
      "userRpId": int,
      "userRp": null,
      "groupId": int,
      "group": null,
      "docs": "string",
      "audit": {
        "createdAt": timestamp,
        "createdBy": "string",
        "createdById": int,
        "updatedAt": timestamp,
        "updatedBy": "string",
        "updatedById": int
      },
      "enabled": boolean
    },
    "contractId": int,
    "duration": "string",
    "durationMin": int,
    "dateOf": date,
    "task": {
      "url": "string",
      "desc": "string"
    },
```

```
                "audit": {
                  "createdAt": timestamp,
                  "createdBy": "string",
                  "createdById": int,
                  "updatedAt": timestamp,
                  "updatedBy": "string",
                  "updatedById": int
                },
                "version": int,
                "user": {
                  "id": int,
                  "email": "string",
                  "fullName": "string",
                  "roleId": "string",
                  "createdAt": timestamp,
                  "lastLogin": timestamp,
                  "rate": float,
                  "position": null,
                  "positionId": int,
                  "typeActivity": "string",
                  "outsourcing": boolean,
                  "groupId": int,
                  "group": null,
                  "enabled": boolean
                },
                "userId": int
              }
            ]
          }
        ]
 400    Bad Request
        { "errorMessage": "string" }
 401    Not Authorized
        { "errorMessage": "string" }
 500    Server Error
        { "errorMessage": "string" }
```

Запрос «Моя сводная информацмя за месяц»

```
Метод: GET
URI: /project-load/activity/my?info&year=[year]& month=[month]
Ответ:
 200    OK
        {
          "userId": int,
          "workMinutes": int,
          "needWorkMinutes": int,
          "activityCount": int,
          "outDaysCount": int,
          "completed": boolean,
          "workHours": float,
          "needWorkHours": float,
          "completedValue": float
        }
 400    Bad Request
```

```
        { "errorMessage": "string" }
 401    Not Authorized
        { "errorMessage": "string" }
 500    Server Error
        { "errorMessage": "string" }
```

Запрос «Получение моих активностей за день»

```
Метод: GET
URI: /project-load/activity/my?day&dateOf=[date]
Ответ:
 200    OK
        [
          {
            "id": int,
            "contract": {
              "id": int,
              "name": "string",
              "fullName": "string",
              "userRpId": int,
              "userRp": null,
              "groupId": int,
              "group": null,
              "docs": "string",
              "audit": {
                "createdAt": timestamp,
                "createdBy": "string",
                "createdById": int,
                "updatedAt": timestamp,
                "updatedBy": "string",
                "updatedById": int
              },
              "enabled": boolean
            },
            "contractId": int,
            "duration": "string",
            "durationMin": int,
            "dateOf": date,
            "task": {
              "url": "string",
              "desc": "string"
            },
            "audit": {
              "createdAt": timestamp,
              "createdBy": "string",
              "createdById": int,
              "updatedAt": timestamp,
              "updatedBy": "string",
              "updatedById": int
            },
            "version": int,
            "user": {
              "id": int,
              "email": "string",
              "fullName": "string",
```

```
        "roleId": "string",
        "createdAt": timestamp,
        "lastLogin": timestamp,
        "rate": float,
        "position": null,
        "positionId": int,
        "typeActivity": "string",
        "outsourcing": boolean,
        "groupId": int,
        "group": null,
        "enabled": boolean
      },
      "userId": int
    },
  ...,
    {
      "id": int,
      "contract": {
        "id": int,
        "name": "string",
        "fullName": "string",
        "userRpId": int,
        "userRp": null,
        "groupId": int,
        "group": null,
        "docs": "string",
        "audit": {
          "createdAt": timestamp,
          "createdBy": "string",
          "createdById": int,
          "updatedAt": timestamp,
          "updatedBy": "string",
          "updatedById": int
        },
        "enabled": boolean
      },
      "contractId": int,
      "duration": "string",
      "durationMin": int,
      "dateOf": date,
      "task": {
        "url": "string",
        "desc": "string"
      },
      "audit": {
        "createdAt": timestamp,
        "createdBy": "string",
        "createdById": int,
        "updatedAt": timestamp,
        "updatedBy": "string",
        "updatedById": int
      },
      "version": int,
      "user": {
        "id": int,
```

```
            "email": "string",
            "fullName": "string",
            "roleId": "string",
            "createdAt": timestamp,
            "lastLogin": timestamp,
            "rate": float,
            "position": null,
            "positionId": int,
            "typeActivity": "string",
            "outsourcing": boolean,
            "groupId": int,
            "group": null,
            "enabled": boolean
          },
          "userId": int
        }
      ]
400   Bad Request
      { "errorMessage": "string" }
401   Not Authorized
      { "errorMessage": "string" }
500   Server Error
      { "errorMessage": "string" }
```

Запрос «Получение календаря со сводной информацией по активностям пользователя»

```
Метод: GET
URI: /project-load/activity/my?calendar&year=[year]&
month=[month]&userId=[userId]
Ответ:
200   OK
      [
        {
          "completed": boolean,
          "dateOf": date,
          "userId": int,
          "workMinutes": int,
          "needWorkMinutes": boolean
          "activityCount": int,
          "shortDay": boolean,
          "userOutDayId": int,
          "userOutDay": boolean,
          "outDay": boolean,
          "userOutDayType": null
        },
        ...,
        {
          "completed": boolean,
          "dateOf": date,
          "userId": int,
          "workMinutes": int,
          "needWorkMinutes": boolean
          "activityCount": int,
```

```
            "shortDay": boolean,
            "userOutDayId": int,
            "userOutDay": boolean,
            "outDay": boolean,
            "userOutDayType": null
          }
        ]
400     Bad Request
        { "errorMessage": "string" }
401     Not Authorized
        { "errorMessage": "string" }
404     Not Found
        { "errorMessage": "string" }
500     Server Error
        { "errorMessage": "string" }
```

Запрос «Получение активностей пользователя за месяц с групировкой по дням»

```
Метод: GET
URI: /project-load/activity/my?list&year=[year]&
month=[month]&userId=[userId]
Ответ:
 200     OK
         [
           {
             "dateOf": date,
             "activities": [
               {
                 "id": int,
                 "contract": {
                   "id": int,
                   "name": "string",
                   "fullName": "string",
                   "userRpId": int,
                   "userRp": null,
                   "groupId": int,
                   "group": null,
                   "docs": "string",
                   "audit": {
                     "createdAt": timestamp,
                     "createdBy": "string",
                     "createdById": int,
                     "updatedAt": timestamp,
                     "updatedBy": "string",
                     "updatedById": int
                   },
                   "enabled": boolean
                 },
                 "contractId": int,
                 "duration": "string",
                 "durationMin": int,
                 "dateOf": date,
                 "task": {
```

```
      "url": "string",
      "desc": "string"
    },
    "audit": {
      "createdAt": timestamp,
      "createdBy": "string",
      "createdById": int,
      "updatedAt": timestamp,
      "updatedBy": "string",
      "updatedById": int
    },
    "version": int,
    "user": {
      "id": int,
      "email": "string",
      "fullName": "string",
      "roleId": "string",
      "createdAt": timestamp,
      "lastLogin": timestamp,
      "rate": float,
      "position": null,
      "positionId": int,
      "typeActivity": "string",
      "outsourcing": boolean,
      "groupId": int,
      "group": null,
      "enabled": boolean
    },
    "userId": int
  },
  ...,
  {
    "id": int,
    "contract": {
      "id": int,
      "name": "string",
      "fullName": "string",
      "userRpId": int,
      "userRp": null,
      "groupId": int,
      "group": null,
      "docs": "string",
      "audit": {
        "createdAt": timestamp,
        "createdBy": "string",
        "createdById": int,
        "updatedAt": timestamp,
        "updatedBy": "string",
        "updatedById": int
      },
      "enabled": boolean
    },
    "contractId": int,
    "duration": "string",
    "durationMin": int,
```

```
        "dateOf": date,
        "task": {
          "url": "string",
          "desc": "string"
        },
        "audit": {
          "createdAt": timestamp,
          "createdBy": "string",
          "createdById": int,
          "updatedAt": timestamp,
          "updatedBy": "string",
          "updatedById": int
        },
        "version": int,
        "user": {
          "id": int,
          "email": "string",
          "fullName": "string",
          "roleId": "string",
          "createdAt": timestamp,
          "lastLogin": timestamp,
          "rate": float,
          "position": null,
          "positionId": int,
          "typeActivity": "string",
          "outsourcing": boolean,
          "groupId": int,
          "group": null,
          "enabled": boolean
        },
        "userId": int
      }
    ]
  },
  ...,
  {
    "dateOf": date,
    "activities": [
      {
        "id": int,
        "contract": {
          "id": int,
          "name": "string",
          "fullName": "string",
          "userRpId": int,
          "userRp": null,
          "groupId": int,
          "group": null,
          "docs": "string",
          "audit": {
            "createdAt": timestamp,
            "createdBy": "string",
            "createdById": int,
            "updatedAt": timestamp,
            "updatedBy": "string",
```

```
        "updatedById": int
      },
      "enabled": boolean
    },
    "contractId": int,
    "duration": "string",
    "durationMin": int,
    "dateOf": date,
    "task": {
      "url": "string",
      "desc": "string"
    },
    "audit": {
      "createdAt": timestamp,
      "createdBy": "string",
      "createdById": int,
      "updatedAt": timestamp,
      "updatedBy": "string",
      "updatedById": int
    },
    "version": int,
    "user": {
      "id": int,
      "email": "string",
      "fullName": "string",
      "roleId": "string",
      "createdAt": timestamp,
      "lastLogin": timestamp,
      "rate": float,
      "position": null,
      "positionId": int,
      "typeActivity": "string",
      "outsourcing": boolean,
      "groupId": int,
      "group": null,
      "enabled": boolean
    },
    "userId": int
  },
  ...,
  {
    "id": int,
    "contract": {
      "id": int,
      "name": "string",
      "fullName": "string",
      "userRpId": int,
      "userRp": null,
      "groupId": int,
      "group": null,
      "docs": "string",
      "audit": {
        "createdAt": timestamp,
        "createdBy": "string",
        "createdById": int,
```

```
                    "updatedAt": timestamp,
                    "updatedBy": "string",
                    "updatedById": int
                  },
                  "enabled": boolean
                },
                "contractId": int,
                "duration": "string",
                "durationMin": int,
                "dateOf": date,
                "task": {
                  "url": "string",
                  "desc": "string"
                },
                "audit": {
                  "createdAt": timestamp,
                  "createdBy": "string",
                  "createdById": int,
                  "updatedAt": timestamp,
                  "updatedBy": "string",
                  "updatedById": int
                },
                "version": int,
                "user": {
                  "id": int,
                  "email": "string",
                  "fullName": "string",
                  "roleId": "string",
                  "createdAt": timestamp,
                  "lastLogin": timestamp,
                  "rate": float,
                  "position": null,
                  "positionId": int,
                  "typeActivity": "string",
                  "outsourcing": boolean,
                  "groupId": int,
                  "group": null,
                  "enabled": boolean
                },
                "userId": int
              }
            ]
          }
        ]
400     Bad Request
        { "errorMessage": "string" }
401     Not Authorized
        { "errorMessage": "string" }
404     Not Found
        { "errorMessage": "string" }
500     Server Error
        { "errorMessage": "string" }
```

Запрос «Сводная информация пользователя за месяц»

```
Метод: GET
URI: /project-load/activity/user?info&year=[year]&month=[month]
&userId=[userId]
Ответ:
 200     OK
         {
           "userId": int,
           "workMinutes": int,
           "needWorkMinutes": int,
           "activityCount": int,
           "outDaysCount": int,
           "completed": boolean,
           "workHours": float,
           "needWorkHours": float,
           "completedValue": float
         }
 400     Bad Request
         { "errorMessage": "string" }
 401     Not Authorized
         { "errorMessage": "string" }
 404     Not Found
         { "errorMessage": "string" }
 500     Server Error
         { "errorMessage": "string" }
```

Запрос «Получение активностей пользователя за день»

```
Метод: GET
URI: /project-load/activity/my?day&dateOf=[date]& userId=[userId]
Ответ:
 200     OK
         [
           {
             "id": int,
             "contract": {
               "id": int,
               "name": "string",
               "fullName": "string",
               "userRpId": int,
               "userRp": null,
               "groupId": int,
               "group": null,
               "docs": "string",
               "audit": {
                 "createdAt": timestamp,
                 "createdBy": "string",
                 "createdById": int,
                 "updatedAt": timestamp,
                 "updatedBy": "string",
                 "updatedById": int
               },
               "enabled": boolean
             },
```

```
      "contractId": int,
      "duration": "string",
      "durationMin": int,
      "dateOf": date,
      "task": {
        "url": "string",
        "desc": "string"
      },
      "audit": {
        "createdAt": timestamp,
        "createdBy": "string",
        "createdById": int,
        "updatedAt": timestamp,
        "updatedBy": "string",
        "updatedById": int
      },
      "version": int,
      "user": {
        "id": int,
        "email": "string",
        "fullName": "string",
        "roleId": "string",
        "createdAt": timestamp,
        "lastLogin": timestamp,
        "rate": float,
        "position": null,
        "positionId": int,
        "typeActivity": "string",
        "outsourcing": boolean,
        "groupId": int,
        "group": null,
        "enabled": boolean
      },
      "userId": int
    },
  ...,
    {
      "id": int,
      "contract": {
        "id": int,
        "name": "string",
        "fullName": "string",
        "userRpId": int,
        "userRp": null,
        "groupId": int,
        "group": null,
        "docs": "string",
        "audit": {
          "createdAt": timestamp,
          "createdBy": "string",
          "createdById": int,
          "updatedAt": timestamp,
          "updatedBy": "string",
          "updatedById": int
        },
```

```
            "enabled": boolean
          },
          "contractId": int,
          "duration": "string",
          "durationMin": int,
          "dateOf": date,
          "task": {
            "url": "string",
            "desc": "string"
          },
          "audit": {
            "createdAt": timestamp,
            "createdBy": "string",
            "createdById": int,
            "updatedAt": timestamp,
            "updatedBy": "string",
            "updatedById": int
          },
          "version": int,
          "user": {
            "id": int,
            "email": "string",
            "fullName": "string",
            "roleId": "string",
            "createdAt": timestamp,
            "lastLogin": timestamp,
            "rate": float,
            "position": null,
            "positionId": int,
            "typeActivity": "string",
            "outsourcing": boolean,
            "groupId": int,
            "group": null,
            "enabled": boolean
          },
          "userId": int
        }
      ]
400   Bad Request
      { "errorMessage": "string" }
401   Not Authorized
      { "errorMessage": "string" }
404   Not Found
      { "errorMessage": "string" }
500   Server Error
      { "errorMessage": "string" }
```

Приложение Д

(обязательное)

Микросервис учета загрузки работников предприятия

Листинг программы

Файл «Activity.java»

```
package ru.iteco.projload.core.activity;

import ru.iteco.projload.audit.AuditEntity;
import ru.iteco.projload.audit.AuditEntityListener;
import ru.iteco.projload.audit.Auditable;
import ru.iteco.projload.core.contract.Contract;
import ru.iteco.projload.core.user.User;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Table(name="pl_activity")
@EntityListeners(AuditEntityListener.class)
@NamedEntityGraphs({
        @NamedEntityGraph(name = "graph.Activity.all",
                attributeNodes = {
                        @NamedAttributeNode(value = "contract"),
                        @NamedAttributeNode(value = "user"),
                })
})
public class Activity implements Auditable {
    @Id
    @SequenceGenerator(
            name = "pl_activity_seq",
            sequenceName = "pl_activity_seq",
            allocationSize = 1)
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"pl_activity_seq")
    @Column(name = "id", nullable = false, updatable = false)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "contract_id", insertable = false, updatable =
false)
    private Contract contract;

    @Column(name = "contract_id", nullable = false)
    private Integer contractId;

    @Column(name = "duration", nullable = false)
    private String duration;

    @Column(name = "duration_min", nullable = false)
    private Integer durationMin;

    @Column(name = "date_of", nullable = false)
    private LocalDate dateOf;

    @Embedded
    private Task task;

    @ManyToOne(fetch = FetchType.LAZY)
```

```
    @JoinColumn(name = "user_id", insertable = false, updatable =
false)
    private User user;

    @Column(name = "user_id", nullable = false)
    private Integer userId;

    @Embedded
    private AuditEntity audit;

    @Version
    @Column(name = "__version")
    private Long version;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Contract getContract() {
        return contract;
    }

    public void setContract(Contract contract) {
        this.contract = contract;
    }

    public Integer getContractId() {
        return contractId;
    }

    public void setContractId(Integer contractId) {
        this.contractId = contractId;
    }

    public String getDuration() {
        return duration;
    }

    public void setDuration(String duration) {
        this.duration = duration;
    }

    public Integer getDurationMin() {
        return durationMin;
    }

    public void setDurationMin(Integer durationMin) {
        this.durationMin = durationMin;
    }

    public LocalDate getDateOf() {
```

```
        return dateOf;
    }

    public void setDateOf(LocalDate dateOf) {
        this.dateOf = dateOf;
    }

    public Task getTask() {
        return task;
    }

    public void setTask(Task task) {
        this.task = task;
    }

    @Override
    public AuditEntity getAudit() {
        return audit;
    }

    @Override
    public void setAudit(AuditEntity audit) {
        this.audit = audit;
    }

    public Long getVersion() {
        return version;
    }

    public void setVersion(Long version) {
        this.version = version;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Integer getUserId() {
        return userId;
    }

    public void setUserId(Integer userId) {
        this.userId = userId;
    }
}
```

Файл «ActivityRepo.java»

```
package ru.iteco.projload.core.activity;

import org.springframework.data.domain.Page;
```

```
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.data.jpa.repository.EntityGraph;
import org.springframework.data.jpa.repository.JpaRepository;
import
org.springframework.data.jpa.repository.JpaSpecificationExecutor;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import ru.iteco.projload.core.useroutday.OutDayType;

import java.math.BigDecimal;
import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

@Repository
public interface ActivityRepo extends JpaRepository<Activity, Long>,
JpaSpecificationExecutor<Activity> {
    @Query(value = "SELECT EXISTS ( " +
            "   SELECT 1 " +
            "   FROM   information_schema.tables " +
            "   WHERE  table_schema = 'public' " +
            "   AND    table_name = :tableName " +
            "   );",
            nativeQuery = true)
    boolean tableIsExists(@Param("tableName") String tableName);

    @Query(value =
            "SELECT count(1) FROM
pl_activity_create_partition(:masterTable, :partitionTable,
:checkFrom, :checkTo)",
            nativeQuery = true)
    int createPartition(@Param("masterTable") String masterTable,
                        @Param("partitionTable") String
partitionTable,
                        @Param("checkFrom") String checkFrom,
                        @Param("checkTo") String checkTo);

    Optional<Activity> findByIdAndDateOf(Long id, LocalDate dateOf);

    @Override
    @EntityGraph(value = "graph.Activity.all", type =
EntityGraph.EntityGraphType.LOAD)
    Page<Activity> findAll(Specification<Activity> specification,
Pageable pageable);

    @Query("select a from Activity a" +
            " where a.userId = :userId" +
            "   and a.contractId = :contractId" +
            "   and a.dateOf >= :fromDate" +
            "   and a.dateOf < :toDate" +
            " order by a.dateOf asc")
    List<Activity> findUserActivitiesByContractOnInterval(
            @Param("userId") Integer userId,
```

```
            @Param("contractId") Integer contractId,
            @Param("fromDate") LocalDate from,
            @Param("toDate") LocalDate to);

    @EntityGraph(value = "graph.Activity.all", type =
EntityGraph.EntityGraphType.LOAD)
    @Query("select a from Activity a" +
            " where a.userId = :userId" +
            "   and a.dateOf >= :fromDate" +
            "   and a.dateOf < :toDate" +
            " order by a.dateOf asc")
    List<Activity> findUserActivitiesOnInterval(
            @Param("userId") Integer userId,
            @Param("fromDate") LocalDate from,
            @Param("toDate") LocalDate to);

    @Query(value = "SELECT " +
            "u.id userId, " +
            "t2.d dateOf, " +
            "t2.is_out_day outDay, " +
            "t2.is_short_day shortDay, " +
            "t2.work_min workMinutes, " +
            "CASE WHEN t2.is_out_day THEN 0  " +
            "     WHEN t2.is_short_day THEN ((60 * (:hourInDay - 1)) *
u.rate) - t2.out_duration_min " +
            "     ELSE (60 * :hourInDay * u.rate) -
t2.out_duration_min END needWorkMinutes, " +
            "t2.activity_count activityCount, " +
            "CASE WHEN t2.out_day_id IS NULL THEN boolean 'false' ELSE
boolean 'true' END userOutDay, " +
            "t2.out_day_type userOutDayType, " +
            "t2.out_day_id userOutDayId " +
            "FROM pl_user u " +
            "INNER JOIN ( " +
            "  SELECT " +
            "    s1.user_id, " +
            "    s1.d, " +
            "    COALESCE(t1.work_min, 0) work_min, " +
            "    CASE WHEN  " +
            "      (ABS(CAST(c.days_out_mask & (1 << CAST(EXTRACT(DAY
FROM s1.d) AS int)) AS bigint))) > 0 " +
            "      THEN boolean 'true' " +
            "      ELSE boolean 'false'  " +
            "      END is_out_day, " +
            "    CASE WHEN  " +
            "      (ABS(CAST(c.days_short_mask & (1 <<
CAST(EXTRACT(DAY FROM s1.d) AS int)) AS bigint))) > 0 " +
            "      THEN boolean 'true' " +
            "      ELSE boolean 'false' " +
            "      END is_short_day, " +
            "    COALESCE(t1.activity_count, 0) activity_count, " +
            "    COALESCE(od.duration_min, 0) out_duration_min, " +
            "    od.id out_day_id, " +
            "    od.out_type out_day_type " +
            "  FROM ( " +
```

```
            "    SELECT  " +
            "      CAST(d AS date), " +
            "      :userId as user_id " +
            "    FROM generate_series( " +
            "      make_timestamp(:year, :month, 1, 0, 0, 0), " +
            "      make_timestamp(:year, :month, 1, 0, 0, 0) +
interval '1 month' - interval '1 day', " +
            "      '1 day' " +
            "    ) gs(d)) s1 " +
            "  LEFT OUTER JOIN ( " +
            "    SELECT " +
            "      a.date_of, " +
            "      a.user_id, " +
            "      sum(a.duration_min) work_min, " +
            "      count(a.id) activity_count " +
            "    FROM pl_activity a " +
            "    WHERE " +
            "      a.user_id = :userId AND " +
            "      (:contractId = 0 OR a.contract_id = :contractId)
AND " +
            "      a.date_of >= make_date(:year, :month, 1) AND " +
            "      a.date_of < make_date(:year, :month , 1) + interval
'1 month' " +
            "    GROUP BY a.date_of, a.user_id " +
            "  ) t1 ON s1.d = t1.date_of " +
            "  INNER JOIN pl_prod_calendar c ON " +
            "    c.year = CAST(EXTRACT(YEAR FROM s1.d) AS int) AND
c.month = CAST(EXTRACT(MONTH FROM s1.d) AS int) " +
            "  LEFT OUTER JOIN pl_user_out_day od ON " +
            "    s1.d = od.out_date AND s1.user_id = od.user_id" +
            ") t2 ON u.id = t2.user_id " +
            "WHERE u.id = :userId " +
            "ORDER BY t2.d ASC", nativeQuery = true)
    List<DaySummaryInfo> userCalendarActivitySummary(@Param("userId")
Integer userId,
@Param("contractId") int contractId,
@Param("year") int year,
@Param("month") int month,
@Param("hourInDay") int hourInDay);

    interface DaySummaryInfo {
        Integer getUserId();
        LocalDate getDateOf();
        Boolean getOutDay();
        Boolean getShortDay();
        Integer getWorkMinutes();
        BigDecimal getNeedWorkMinutes();
        Integer getActivityCount();
        Boolean getUserOutDay();
        OutDayType getUserOutDayType();
        Integer getUserOutDayId();
        default Boolean isCompleted() {
            return
BigDecimal.valueOf(getWorkMinutes()).compareTo(getNeedWorkMinutes())
>= 0;
```

```
        }
    }

    List<Activity> findAllByUserIdAndDateOf(Integer userId, LocalDate
date);

    List<Activity> findAllByUserIdAndDateOfIn(Integer userId,
List<LocalDate> datesOf);
}
```

Файл «Task.java»

```
package ru.iteco.projload.core.activity;

import org.hibernate.validator.constraints.URL;

import javax.persistence.Column;
import javax.persistence.Embeddable;
import java.io.Serializable;

@Embeddable
public class Task implements Serializable {

    @URL
    @Column(name = "task_url")
    private String url;

    @Column(name = "task_desc")
    private String desc;

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getDesc() {
        return desc;
    }

    public void setDesc(String desc) {
        this.desc = desc;
    }
}
```

Файл «ActivityService.java»

```
package ru.iteco.projload.core.activity;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.annotation.DependsOn;
import org.springframework.stereotype.Service;
```

```
import
org.springframework.transaction.event.TransactionalEventListener;
import ru.iteco.projload.base.CreationEvent;
import ru.iteco.projload.base.FindableService;
import ru.iteco.projload.core.activity.web.*;
import
ru.iteco.projload.core.activity.web.converter.ActivityConverter;
import
ru.iteco.projload.core.activity.web.converter.ActivityToViewDetail;
import ru.iteco.projload.core.prodcalendar.ProdCalendar;
import ru.iteco.projload.core.prodcalendar.ProdCalendarUtilService;
import ru.iteco.projload.core.sumactivity.SummaryActivityService;
import ru.iteco.projload.core.user.User;
import ru.iteco.projload.core.user.UserBase;
import ru.iteco.projload.core.user.UserService;
import ru.iteco.projload.core.useroutday.OutDay;
import ru.iteco.projload.core.useroutday.OutDayService;
import ru.iteco.projload.error.EntityNotFoundException;
import ru.iteco.projload.property.ApplicationProperty;
import ru.iteco.projload.util.TimeUtil;

import javax.annotation.PostConstruct;
import javax.transaction.Transactional;
import java.time.LocalDate;
import java.time.YearMonth;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

import static ru.iteco.projload.core.user.UserService.currentUser;

@Service
@DependsOn({"prodCalendarService"})
public class ActivityService extends FindableService<Activity,
ActivityView> {
    private static Logger logger =
LoggerFactory.getLogger(ActivityService.class);

    private static final String PARTITION_MASTER_TABLE =
"pl_activity";

    private final ActivityRepo repo;
    private final ActivityConverter converter;
    private final SummaryActivityService sumActivityService;
    private final UserService userService;
    private final ProdCalendarUtilService prodCalendarUtilService;
    private final OutDayService outDayService;
    private final ApplicationProperty property;

    public ActivityService(final ActivityRepo repo,
                           final ActivityConverter converter,
                           final ActivityToViewDetail toViewDetail,
```

```
                            final SummaryActivityService
sumActivityService,
                            final UserService userService,
                            final ProdCalendarUtilService
prodCalendarUtilService,
                            OutDayService outDayService, final
ApplicationProperty property) {
        super(repo, toViewDetail);
        this.repo = repo;
        this.converter = converter;
        this.sumActivityService = sumActivityService;
        this.userService = userService;
        this.prodCalendarUtilService = prodCalendarUtilService;
        this.outDayService = outDayService;
        this.property = property;
    }

    @PostConstruct
    public void onCreated() {
        List<Integer> years =
prodCalendarUtilService.availableYears();
        for (Integer year : years) {
            this.checkAndCreateYearPartition(year);
        }
    }

    @TransactionalEventListener
    public void onProdCalendarCreateEvent(CreationEvent<ProdCalendar>
event) {
        logger.info("Handling production calendar create event...");
        ProdCalendar prodCalendar = event.getObject();

this.checkAndCreateYearPartition(prodCalendar.getId().getYear());
        logger.info("Success handling");
    }

    private void checkAndCreateYearPartition(Integer year) {
        String partitionName = this.createPartitionName(year);
        if (repo.tableIsExists(partitionName))
            return;
        DateTimeFormatter iso = DateTimeFormatter.ISO_LOCAL_DATE;
        String from = TimeUtil.startOfYear(year).format(iso);
        String to = TimeUtil.endOfYear(year).plusDays(1).format(iso);
        logger.info("Creating partition with name " + partitionName);
        int result = repo.createPartition(PARTITION_MASTER_TABLE,
partitionName, from, to);
        if (result > 0)
            logger.info("Partition created");
        else logger.error("Failed create partition");
    }

    private String createPartitionName(Integer year) {
        return String.format("%s_y%s", PARTITION_MASTER_TABLE, year);
    }
```

```
    @Transactional
    public ActivityView create(ActivityCreateReq req) {
        Activity activity = new Activity();
        User user = req.getUserId() == null ? currentUser() :
                userService.findUserOrThrow(req.getUserId());
        this.prepare(activity, user, req);
        Activity instance = repo.save(activity);
        sumActivityService.addWorkMinutes(activity);
        return converter.toView(instance);
    }

    @Transactional
    public List<ActivityView> bulkCreate(ActivityBulkCreateReq req) {
        List<Activity> activities = new ArrayList<>();
        User user = req.getUserId() == null ? currentUser() :
                userService.findUserOrThrow(req.getUserId());
        for (LocalDate dateOf : req.getDatesOf()) {
            Activity activity = new Activity();
            this.prepare(activity, user, req, dateOf);
            activities.add(activity);
        }
        List<Activity> instances = repo.saveAll(activities);
        sumActivityService.addWorkMinutes(instances);
        return instances.stream()
                .map(converter::toView)
                .collect(Collectors.toList());
    }

    @Transactional
    public ActivityView update(Activity activity, ActivityUpdateReq
req) {
        boolean contractUpdate =
!activity.getContractId().equals(req.getContractId());
        if (contractUpdate)
            sumActivityService.subtractWorkMinutes(activity);
        this.prepareForUpdate(activity, req);
        Activity instance = repo.save(activity);
        sumActivityService.updateWorkMinutes(activity);
        return converter.toView(instance);
    }

    public void prepare(Activity activity, User user,
ActivityCreateReq req) {
        this.prepare(activity, user, req, req.getDateOf());
    }

    public void prepare(Activity activity, User user,
AbstractActivityReq req, LocalDate dateOf) {
        activity.setDuration(req.getDuration());
        activity.setDurationMin(req.getDurationMin());
        activity.setContractId(req.getContractId());
        activity.setDateOf(dateOf);
        activity.setTask(req.getTask());
        activity.setUser(user);
        activity.setUserId(user.getId());
```

```
    }

    public void prepareForUpdate(Activity activity, ActivityUpdateReq
req) {
        activity.setDuration(req.getDuration());
        activity.setDurationMin(req.getDurationMin());
        activity.setDateOf(req.getDateOf());
        activity.setTask(req.getTask());
        activity.setContractId(req.getContractId());
    }

    public Activity findOrThrow(Long id, LocalDate dateOf) {
        return repo.findByIdAndDateOf(id, dateOf)
                .orElseThrow(() -> new EntityNotFoundException(
                        String.format("Activity with id %s and date %s
not found ", id, dateOf)));
    }

    @Transactional
    public void delete(Activity activity) {
        sumActivityService.subtractWorkMinutes(activity);
        repo.delete(activity);
    }

    @Transactional
    public void bulkDelete(List<Activity> activities) {
        if (activities.isEmpty())
            return;
        sumActivityService.subtractWorkMinutes(activities);
        repo.deleteAll(activities);
    }

    public List<ActivityRepo.DaySummaryInfo>
userCalendarActivitySummary(Integer userId, Integer contractId,
YearMonth yearMonth) {
        int hourInDay = property.getWorkTypeCompany().getHourInDay();
        int safeContractId = contractId == null ? 0 : contractId;
        return repo.userCalendarActivitySummary(userId,
safeContractId, yearMonth.getYear(), yearMonth.getMonthValue(),
hourInDay);
    }

    public List<ActivitiesViewGroupDay>
findGroupOfDayUserActivitiesOnMonth(Integer userId, YearMonth
yearMonth) {
        LocalDate dateOfMonth = TimeUtil.startDateOfMonth(yearMonth);
        LocalDate dateOfNextMonth =
TimeUtil.startDateOfNextMonth(yearMonth);
        List<Activity> activities = repo.findUserActivitiesOnInterval(
                userId, dateOfMonth, dateOfNextMonth);
        final List<ActivitiesViewGroupDay> viewGroupDays = new
ArrayList<>();
        activities.stream()
                .collect(Collectors.groupingBy(
                        Activity::getDateOf,
```

```
                        LinkedHashMap::new, // guarantee save ordering
                        Collectors.toList()))
                .forEach((k, v) -> {
                    ActivitiesViewGroupDay viewGroupDay =
ActivitiesViewGroupDay.ofWithDetails(k, v, converter);
                    viewGroupDays.add(viewGroupDay);
                });
        return viewGroupDays;
    }

    public List<ActivityView> findUserActivitiesOnDay(Integer userId,
LocalDate dateOf) {
        return repo.findAllByUserIdAndDateOf(userId, dateOf)
                .stream()
                .map(converter::toViewDetail)
                .collect(Collectors.toList());
    }

    public List<Activity> findUserActivitiesOnDays(Integer userId,
Set<LocalDate> dates) {
        return repo.findAllByUserIdAndDateOfIn(userId, new
ArrayList<>(dates));
    }

    public MonthSummaryInfo findMonthSummaryInfo(UserBase user,
YearMonth yearMonth) {
        LocalDate dateOfMonth = TimeUtil.startDateOfMonth(yearMonth);
        LocalDate dateOfNextMonth =
TimeUtil.startDateOfNextMonth(yearMonth);
        List<Activity> activities =
repo.findUserActivitiesOnInterval(user.getId(), dateOfMonth,
dateOfNextMonth);
        Integer workMinutes = activities.stream()
                .mapToInt(Activity::getDurationMin)
                .sum();
        Integer needWorkMinutes = prodCalendarUtilService
                .workMinutesOfMonthForUser(yearMonth.getYear(),
yearMonth.getMonthValue(), user);
        int year = yearMonth.getYear();
        int month = yearMonth.getMonthValue();
        List<OutDay> outDays =
outDayService.outDaysUserOnMonth(user.getId(), year, month);
        Integer outWorkMinutes = outDays
                .stream()
                .mapToInt(OutDay::getDurationMin)
                .sum();
        if (outWorkMinutes > 0)
            needWorkMinutes = needWorkMinutes - outWorkMinutes;
        return MonthSummaryInfo.of(user.getId(), workMinutes,
needWorkMinutes, activities.size(), outDays.size());
    }
}
```

Файл «AbstractActivityReq.java»

```
package ru.iteco.projload.core.activity.web;

import
com.github.jonpeterson.jackson.module.interceptor.JsonInterceptors;
import ru.iteco.projload.constraint.DurationFormat;
import ru.iteco.projload.core.activity.Task;
import ru.iteco.projload.util.ExtractDurationMinutesInterceptor;

import javax.validation.Valid;
import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import java.io.Serializable;

@JsonInterceptors(beforeDeserialization =
ExtractDurationMinutesInterceptor.class)
public abstract class AbstractActivityReq implements Serializable {
    @NotNull
    private Integer contractId;

    @NotEmpty
    @DurationFormat
    private String duration;

    @Min(30)
    @Max(1440)
    private Integer durationMin;

    private @Valid Task task;

    private Integer userId;

    public Integer getContractId() {
        return contractId;
    }

    public void setContractId(Integer contractId) {
        this.contractId = contractId;
    }

    public String getDuration() {
        return duration;
    }

    public void setDuration(String duration) {
        this.duration = duration;
    }

    public Task getTask() {
        return task;
    }

    public void setTask(Task task) {
        this.task = task;
```

```
    }

    public Integer getDurationMin() {
        return durationMin;
    }

    public void setDurationMin(Integer durationMin) {
        this.durationMin = durationMin;
    }

    public Integer getUserId() {
        return userId;
    }

    public void setUserId(Integer userId) {
        this.userId = userId;
    }
}
```

Файл «ActivitiesViewGroupDay.java»

```
package ru.iteco.projload.core.activity.web;

import ru.iteco.projload.core.activity.Activity;
import
ru.iteco.projload.core.activity.web.converter.ActivityConverter;

import java.io.Serializable;
import java.time.LocalDate;
import java.util.List;
import java.util.stream.Collectors;

public class ActivitiesViewGroupDay implements Serializable {
    private LocalDate dateOf;
    private List<ActivityView> activities;

    public static ActivitiesViewGroupDay ofWithDetails(LocalDate
dateOf, List<Activity> activities, ActivityConverter converter) {
        List<ActivityView> views = activities
                .stream()
                .map(converter::toViewDetail)
                .collect(Collectors.toList());
        ActivitiesViewGroupDay viewGroupDay = new
ActivitiesViewGroupDay();
        viewGroupDay.setDateOf(dateOf);
        viewGroupDay.setActivities(views);
        return viewGroupDay;
    }

    public LocalDate getDateOf() {
        return dateOf;
    }

    public void setDateOf(LocalDate dateOf) {
        this.dateOf = dateOf;
```

```
    }

    public List<ActivityView> getActivities() {
        return activities;
    }

    public void setActivities(List<ActivityView> activities) {
        this.activities = activities;
    }
}
```

Файл «ActivityBaseReq.java»

```
package ru.iteco.projload.core.activity.web;

import javax.validation.constraints.NotNull;
import java.time.LocalDate;

public abstract class ActivityBaseReq extends AbstractActivityReq {
    @NotNull
    private LocalDate dateOf;

    public LocalDate getDateOf() {
        return dateOf;
    }

    public void setDateOf(LocalDate dateOf) {
        this.dateOf = dateOf;
    }
}
```

Файл «ActivityBulkCreateReq.java»

```
package ru.iteco.projload.core.activity.web;

import javax.validation.constraints.NotEmpty;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;

public class ActivityBulkCreateReq extends AbstractActivityReq {
    @NotEmpty
    private Set<LocalDate> datesOf;

    public Set<LocalDate> getDatesOf() {
        return datesOf;
    }

    public List<LocalDate> getDatesOfList() {
        if (datesOf == null)
            return null;
        return new ArrayList<>(datesOf);
    }
```

```
    public void setDatesOf(Set<LocalDate> datesOf) {
        this.datesOf = datesOf;
    }
}
```

Файл «ActivityController.java»

```
package ru.iteco.projload.core.activity.web;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.http.HttpStatus;
import org.springframework.util.Assert;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.*;
import ru.iteco.projload.base.dto.UserDatesReq;
import ru.iteco.projload.core.activity.Activity;
import ru.iteco.projload.core.activity.ActivityRepo;
import ru.iteco.projload.core.activity.ActivityService;
import ru.iteco.projload.core.activity.web.converter.ActivityToView;
import
ru.iteco.projload.core.activity.web.validator.ActivityBulkCreateReqVal
idator;
import
ru.iteco.projload.core.activity.web.validator.ActivityCreateReqValidat
or;
import
ru.iteco.projload.core.activity.web.validator.ActivityUpdateReqValidat
or;
import ru.iteco.projload.core.user.ArchiveUser;
import ru.iteco.projload.core.user.ArchiveUserService;
import ru.iteco.projload.core.user.User;

import javax.validation.Valid;
import java.time.LocalDate;
import java.time.YearMonth;
import java.util.List;
import java.util.stream.Collectors;

import static ru.iteco.projload.core.user.UserService.currentUser;
import static
ru.iteco.projload.search.SearchHelper.createJPASpecification;


@Api(value = "activity", tags = {"Операции с активностями
сотрудников"})
@RestController
@RequestMapping("/activity")
public class ActivityController {
```

```
    private final ActivityService service;
    private final ActivityCreateReqValidator createReqValidator;
    private final ActivityUpdateReqValidator updateReqValidator;
    private final ActivityBulkCreateReqValidator
bulkCreateReqValidator;
    private final ArchiveUserService userService;
    private final ActivityToView toView;

    public ActivityController(final ActivityService service,
                              final ActivityCreateReqValidator
createReqValidator,
                              final ActivityUpdateReqValidator
updateReqValidator,
                              final ActivityBulkCreateReqValidator
bulkCreateReqValidator,
                              final ArchiveUserService userService,
                              final ActivityToView toView) {
        this.service = service;
        this.createReqValidator = createReqValidator;
        this.updateReqValidator = updateReqValidator;
        this.bulkCreateReqValidator = bulkCreateReqValidator;
        this.userService = userService;
        this.toView = toView;
    }

    @InitBinder("activityCreateReq")
    public void bindActivityCreateValidator(WebDataBinder binder) {
        binder.addValidators(createReqValidator);
    }

    @InitBinder("activityUpdateReq")
    public void bindActivityUpdateValidator(WebDataBinder binder) {
        binder.addValidators(updateReqValidator);
    }

    @InitBinder("activityBulkCreateReq")
    public void bindActivityBulkCreateValidator(WebDataBinder binder)
{
        binder.addValidators(bulkCreateReqValidator);
    }

    @ApiOperation(value = "Создание активности")
    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    @ResponseBody
    public ActivityView create(@RequestBody @Valid ActivityCreateReq
req) {
        return service.create(req);
    }

    @ApiOperation(value = "Создание активностей на несколько дней")
    @PostMapping(value = "/bulk")
    public List<ActivityView> bulkCreate(@RequestBody @Valid
ActivityBulkCreateReq req) {
        return service.bulkCreate(req);
```

```
    }

    @ApiOperation(value = "Удаление активности")
    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void delete(@PathVariable Long id,
                       @RequestParam @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) LocalDate dateOf) {
        Activity activity = service.findOrThrow(id, dateOf);
        service.delete(activity);
    }

    @ApiOperation(value = "Удаление активностей")
    @DeleteMapping(value = "/bulk")
    @ResponseBody
    public List<ActivityView> bulkDelete(@RequestBody @Valid
UserDatesReq req) {
        List<Activity> activities =
service.findUserActivitiesOnDays(req.getUser().getId(),
req.getDatesOf());
        List<ActivityView> views = activities
                .stream()
                .map(toView::convert)
                .collect(Collectors.toList());
        service.bulkDelete(activities);
        return views;
    }

    @ApiOperation(value = "Обновление активности")
    @PutMapping("/{id}")
    @ResponseBody
    public ActivityView update(@PathVariable Long id, @RequestBody
@Valid ActivityUpdateReq req) {
        Activity activity = service.findOrThrow(id, req.getDateOf());
        return service.update(activity, req);
    }

    @ApiOperation(value = "Поиск активностей")
    @GetMapping(value = {"", "/"})
    @ResponseBody
    public Page<ActivityView> searchActivities(@RequestParam(value =
"search") String search,
                                               @PageableDefault(sort =
{"audit.createdAt"}, direction = Sort.Direction.DESC) Pageable
pageable) {
        return service.findAll(createJPASpecification(search, true),
pageable);
    }

    @ApiOperation(value = "Мой календарь со сводной информацией по
аквтивностям")
    @GetMapping(value = "/my", params = "calendar")
    @ResponseBody
    public List<ActivityRepo.DaySummaryInfo>
myActivitiesInfoOnMonth(@RequestParam(value = "year") Integer year,
```

```
@RequestParam(value = "month") Integer month,

@RequestParam(value = "contractId", required = false) Integer
contractId) {
        User user = currentUser();
        Assert.notNull(user, "Current user is null!");
        return service.userCalendarActivitySummary(user.getId(),
contractId, YearMonth.of(year, month));
    }

    @ApiOperation(value = "Мои активности за месяц сгрупированные по
дням")
    @GetMapping(value = "/my", params = "list")
    @ResponseBody
    public List<ActivitiesViewGroupDay>
myActivitiesOnMontGroupDays(@RequestParam(value = "year") Integer
year,

@RequestParam(value = "month") Integer month) {
        User user = currentUser();
        Assert.notNull(user, "Current user is null!");
        return
service.findGroupOfDayUserActivitiesOnMonth(user.getId(),
YearMonth.of(year, month));
    }

    @ApiOperation(value = "Мои активности за день")
    @GetMapping(value = "/my", params = "day")
    @ResponseBody
    public List<ActivityView> myActivitiesOnDay(
            @RequestParam(value = "dateOf")
            @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate
dateOf) {
        User user = currentUser();
        Assert.notNull(user, "Current user is null!");
        return service.findUserActivitiesOnDay(user.getId(), dateOf);
    }

    @ApiOperation(value = "Моя сводная информация за месяц")
    @GetMapping(value = "/my", params = "info")
    @ResponseBody
    public MonthSummaryInfo myMonthSummaryInfo(@RequestParam(value =
"year") Integer year,
                                               @RequestParam(value =
"month") Integer month) {
        User user = currentUser();
        Assert.notNull(user, "Current user is null!");
        return service.findMonthSummaryInfo(user, YearMonth.of(year,
month));
    }

    @ApiOperation(value = "Календарь со сводной информацией по
аквтивностям пользователя")
    @GetMapping(value = "/user", params = "calendar")
```

```
    @ResponseBody
    public List<ActivityRepo.DaySummaryInfo>
userActivitiesInfoOnMonth(@RequestParam(value = "year") Integer year,

@RequestParam(value = "month") Integer month,

@RequestParam(value = "userId") Integer userId,

@RequestParam(value = "contractId", required = false) Integer
contractId) {
        ArchiveUser user = userService.findUserOrThrow(userId);
        return service.userCalendarActivitySummary(user.getId(),
contractId, YearMonth.of(year, month));
    }

    @ApiOperation(value = "Активности пользователя за месяц
сгрупированные по дням")
    @GetMapping(value = "/user", params = "list")
    @ResponseBody
    public List<ActivitiesViewGroupDay>
userActivitiesOnMontGroupDays(@RequestParam(value = "year") Integer
year,

@RequestParam(value = "month") Integer month,

@RequestParam(value = "userId") Integer userId) {
        ArchiveUser user = userService.findUserOrThrow(userId);
        return
service.findGroupOfDayUserActivitiesOnMonth(user.getId(),
YearMonth.of(year, month));
    }

    @ApiOperation(value = "Активности пользователя за день")
    @GetMapping(value = "/user", params = "day")
    @ResponseBody
    public List<ActivityView> userActivitiesOnDay(
            @RequestParam(value = "dateOf")
            @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate
dateOf,
            @RequestParam(value = "userId") Integer userId) {
        ArchiveUser user = userService.findUserOrThrow(userId);
        return service.findUserActivitiesOnDay(user.getId(), dateOf);
    }

    @ApiOperation(value = "Сводная информация пользователя за месяц")
    @GetMapping(value = "/user", params = "info")
    @ResponseBody
    public MonthSummaryInfo monthSummaryInfo(@RequestParam(value =
"year") Integer year,
                                             @RequestParam(value =
"month") Integer month,
                                             @RequestParam(value =
"userId") Integer userId) {
        ArchiveUser user = userService.findUserOrThrow(userId);
```

```
        return service.findMonthSummaryInfo(user, YearMonth.of(year,
month));
    }
}
```

Файл «ActivityCreateReq.java»

```
package ru.iteco.projload.core.activity.web;

public class ActivityCreateReq extends ActivityBaseReq {
}
```

Файл «ActivityUpdateReq.java»

```
package ru.iteco.projload.core.activity.web;

import javax.validation.constraints.NotNull;

public class ActivityUpdateReq extends ActivityBaseReq {
    @NotNull
    private Long id;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
}
```

Файл «ActivityView.java»

```
package ru.iteco.projload.core.activity.web;

import ru.iteco.projload.audit.AuditEntity;
import ru.iteco.projload.core.activity.Task;
import ru.iteco.projload.core.contract.web.ContractView;
import ru.iteco.projload.core.user.web.UserView;

import java.io.Serializable;
import java.time.LocalDate;

public class ActivityView implements Serializable {
    private Long id;
    private ContractView contract;
    private Integer contractId;
    private String duration;
    private Integer durationMin;
    private LocalDate dateOf;
    private Task task;
    private AuditEntity audit;
    private Long version;
```

```
private UserView user;
private Integer userId;

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public ContractView getContract() {
    return contract;
}

public void setContract(ContractView contract) {
    this.contract = contract;
}

public Integer getContractId() {
    return contractId;
}

public void setContractId(Integer contractId) {
    this.contractId = contractId;
}

public String getDuration() {
    return duration;
}

public void setDuration(String duration) {
    this.duration = duration;
}

public Integer getDurationMin() {
    return durationMin;
}

public void setDurationMin(Integer durationMin) {
    this.durationMin = durationMin;
}

public LocalDate getDateOf() {
    return dateOf;
}

public void setDateOf(LocalDate dateOf) {
    this.dateOf = dateOf;
}

public Task getTask() {
    return task;
}
```

```
    public void setTask(Task task) {
        this.task = task;
    }

    public AuditEntity getAudit() {
        return audit;
    }

    public void setAudit(AuditEntity audit) {
        this.audit = audit;
    }

    public Long getVersion() {
        return version;
    }

    public void setVersion(Long version) {
        this.version = version;
    }

    public UserView getUser() {
        return user;
    }

    public void setUser(UserView user) {
        this.user = user;
    }

    public Integer getUserId() {
        return userId;
    }

    public void setUserId(Integer userId) {
        this.userId = userId;
    }
}
```

Файл «MonthSummaryInfo.java»

```
package ru.iteco.projload.core.activity.web;

import com.fasterxml.jackson.annotation.JsonProperty;
import ru.iteco.projload.util.SumActivityUtil;
import ru.iteco.projload.util.TimeUtil;

import java.io.Serializable;
import java.math.BigDecimal;


public class MonthSummaryInfo implements Serializable {
    private Integer userId;
    private Integer workMinutes;
    private Integer needWorkMinutes;
    private Integer activityCount;
    private Integer outDaysCount;
```

```
    public static MonthSummaryInfo of(Integer userId,
                                      Integer workMinutes,
                                      Integer needWorkMinutes,
                                      Integer activityCount,
                                      Integer outDaysCount) {
        MonthSummaryInfo info = new MonthSummaryInfo();
        info.setUserId(userId);
        info.setWorkMinutes(workMinutes);
        info.setNeedWorkMinutes(needWorkMinutes);
        info.setActivityCount(activityCount);
        info.setOutDaysCount(outDaysCount);
        return info;
    }

    @JsonProperty("completed")
    public Boolean isCompleted() {
        return this.workMinutes >= this.needWorkMinutes;
    }

    @JsonProperty("workHours")
    public BigDecimal getWorkHours() {
        if (this.workMinutes == null)
            return null;
        return TimeUtil.minutesToHours(this.workMinutes);
    }

    @JsonProperty("needWorkHours")
    public BigDecimal getNeedWorkHours() {
        if (this.needWorkMinutes == null)
            return null;
        return TimeUtil.minutesToHours(this.needWorkMinutes);
    }

    @JsonProperty("completedValue")
    public BigDecimal getCompletedValue() {
        if (this.workMinutes == null || this.needWorkMinutes == null)
            return null;
        return
SumActivityUtil.calculateSafeCompletedValue(this.workMinutes,
this.needWorkMinutes);
    }

    public Integer getUserId() {
        return userId;
    }

    public void setUserId(Integer userId) {
        this.userId = userId;
    }

    public Integer getWorkMinutes() {
        return workMinutes;
    }
```

```
    public void setWorkMinutes(Integer workMinutes) {
        this.workMinutes = workMinutes;
    }

    public Integer getNeedWorkMinutes() {
        return needWorkMinutes;
    }

    public void setNeedWorkMinutes(Integer needWorkMinutes) {
        this.needWorkMinutes = needWorkMinutes;
    }

    public Integer getActivityCount() {
        return activityCount;
    }

    public void setActivityCount(Integer activityCount) {
        this.activityCount = activityCount;
    }

    public Integer getOutDaysCount() {
        return outDaysCount;
    }

    public void setOutDaysCount(Integer outDaysCount) {
        this.outDaysCount = outDaysCount;
    }
}
```

Файл «AbstractActivityReqValidator.java»

```
package ru.iteco.projload.core.activity.web.validator;

import org.springframework.security.access.AccessDeniedException;
import org.springframework.util.Assert;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;
import ru.iteco.projload.core.activity.web.AbstractActivityReq;
import ru.iteco.projload.core.contract.ContractRepo;
import ru.iteco.projload.core.user.User;
import ru.iteco.projload.core.user.UserRepo;
import ru.iteco.projload.core.user.UserRole;
import ru.iteco.projload.property.ApplicationProperty;
import ru.iteco.projload.util.MessageUtil;

import static ru.iteco.projload.core.user.UserService.currentUser;


public abstract class AbstractActivityReqValidator implements
Validator {
    private final UserRepo userRepo;
    private final ContractRepo contractRepo;
    private final ApplicationProperty property;
    private final MessageUtil messageUtil;
```

```
    public AbstractActivityReqValidator(final UserRepo userRepo,
                                        final ContractRepo
contractRepo,
                                        final ApplicationProperty
property,
                                        final MessageUtil messageUtil)
{
        this.userRepo = userRepo;
        this.contractRepo = contractRepo;
        this.property = property;
        this.messageUtil = messageUtil;
    }

    @Override
    public void validate(Object o, Errors errors) {
        if (errors.hasErrors()) return;
        AbstractActivityReq req = (AbstractActivityReq) o;
        this.validateAccessToCreateOrUpdate(req);
        if (req.getDurationMin() > property.getMaxWorkMinutesInDay())
            errors.reject("activity.ManyWorkTime",
                    messageUtil.getValidationMessage(
                            "activity.ManyWorkTime",
req.getDuration(), property.getMaxWorkHoursInDay()));
        if (!contractRepo.existsById(req.getContractId())) {
            errors.reject("activity.ContractNotExists",

messageUtil.getValidationMessage("activity.ContractNotExists",
req.getContractId()));
        }
    }

    public User getUserOrReject(AbstractActivityReq req, Errors
errors) {
        User user = req.getUserId() == null ?
                currentUser() :
                userRepo.findById(req.getUserId()).orElse(null);
        if (user == null)
            errors.reject("profile.NotFound",
messageUtil.getValidationMessage("profile.NotFound"));
        return user;
    }

    private void validateAccessToCreateOrUpdate(AbstractActivityReq
req) {
        boolean isAccess = true;
        User user = currentUser();
        Assert.notNull(user, "Current user is null");
        if (req.getUserId() != null &&
!user.getId().equals(req.getUserId())) {
            UserRole.Type role = user.getRoleId();
            isAccess = role.equals(UserRole.Type.ADMIN) ||
role.equals(UserRole.Type.RP);
        }
```

```
        if (!isAccess) throw new AccessDeniedException("Access to
create or update activity denied");
    }
}
```

Файл «ActivityBaseReqValidator.java»

```
package ru.iteco.projload.core.activity.web.validator;

import org.springframework.validation.Errors;
import ru.iteco.projload.core.activity.web.ActivityBaseReq;
import ru.iteco.projload.core.contract.ContractRepo;
import ru.iteco.projload.core.prodcalendar.ProdCalendar;
import ru.iteco.projload.core.prodcalendar.ProdCalendarId;
import ru.iteco.projload.core.prodcalendar.ProdCalendarRepo;
import ru.iteco.projload.core.user.UserRepo;
import ru.iteco.projload.core.useroutday.OutDayRepo;
import ru.iteco.projload.property.ApplicationProperty;
import ru.iteco.projload.util.MessageUtil;

import java.time.LocalDate;

public abstract class ActivityBaseReqValidator extends
AbstractActivityReqValidator {
    private final ProdCalendarRepo calendarRepo;
    private final OutDayRepo outDayRepo;
    private final MessageUtil messageUtil;

    public ActivityBaseReqValidator(final UserRepo userRepo,
                                    final ContractRepo contractRepo,
                                    final ApplicationProperty
property,
                                    final ProdCalendarRepo
calendarRepo,
                                    final OutDayRepo outDayRepo,
                                    final MessageUtil messageUtil) {
        super(userRepo, contractRepo, property, messageUtil);
        this.calendarRepo = calendarRepo;
        this.outDayRepo = outDayRepo;
        this.messageUtil = messageUtil;
    }

    @Override
    public void validate(Object o, Errors errors) {
        if (errors.hasErrors()) return;
        super.validate(o, errors);
        if (errors.hasErrors()) return;
        ActivityBaseReq req = (ActivityBaseReq) o;
        LocalDate dateOf = req.getDateOf();
        ProdCalendarId calendarId = new
ProdCalendarId(dateOf.getYear(), dateOf.getMonthValue());
        ProdCalendar calendar =
calendarRepo.findById(calendarId).orElse(null);
        if (calendar == null) {
            errors.reject("activity.ProdCalendarNotExists",
```

```
messageUtil.getValidationMessage("activity.ProdCalendarNotExists",
dateOf));
            return;
        }
        if (!calendar.getEnabled()) {
            errors.reject("activity.ProdCalendarDisabled",

messageUtil.getValidationMessage("activity.ProdCalendarDisabled"));
        }
        outDayRepo.findFirstByUserIdAndDate(req.getUserId(),
dateOf).ifPresent(
                userOutDay -> errors.reject("activity.IsUserOutDay",

messageUtil.getValidationMessage("activity.IsUserOutDay",
                                 dateOf,
userOutDay.getOutType().getLabelRu())));
    }
}
```

Файл «ActivityBulkCreateReqValidator.java»

```
package ru.iteco.projload.core.activity.web.validator;

import org.springframework.stereotype.Component;
import org.springframework.validation.Errors;
import ru.iteco.projload.core.activity.Activity;
import ru.iteco.projload.core.activity.ActivityRepo;
import ru.iteco.projload.core.activity.web.ActivityBulkCreateReq;
import ru.iteco.projload.core.contract.ContractRepo;
import ru.iteco.projload.core.prodcalendar.ProdCalendar;
import ru.iteco.projload.core.prodcalendar.ProdCalendarId;
import ru.iteco.projload.core.prodcalendar.ProdCalendarRepo;
import ru.iteco.projload.core.user.User;
import ru.iteco.projload.core.user.UserRepo;
import ru.iteco.projload.core.useroutday.OutDay;
import ru.iteco.projload.core.useroutday.OutDayRepo;
import ru.iteco.projload.property.ApplicationProperty;
import ru.iteco.projload.util.MessageUtil;
import ru.iteco.projload.util.TimeUtil;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

import static java.util.stream.Collectors.groupingBy;
import static java.util.stream.Collectors.summingInt;

@Component
public class ActivityBulkCreateReqValidator extends
AbstractActivityReqValidator {
    private final ActivityRepo activityRepo;
    private final ProdCalendarRepo calendarRepo;
```

```
    private final OutDayRepo outDayRepo;
    private final ApplicationProperty property;
    private final MessageUtil messageUtil;

    public ActivityBulkCreateReqValidator(final UserRepo userRepo,
                                          final ActivityRepo
activityRepo,
                                          final ContractRepo
contractRepo,
                                          final ProdCalendarRepo
calendarRepo,
                                          final OutDayRepo outDayRepo,
                                          final ApplicationProperty
property,
                                          final MessageUtil
messageUtil) {
        super(userRepo, contractRepo, property, messageUtil);
        this.activityRepo = activityRepo;
        this.calendarRepo = calendarRepo;
        this.outDayRepo = outDayRepo;
        this.property = property;
        this.messageUtil = messageUtil;
    }

    @Override
    public boolean supports(Class<?> aClass) {
        return aClass.equals(ActivityBulkCreateReq.class);
    }

    @Override
    public void validate(Object o, Errors errors) {
        if (errors.hasErrors()) return;
        super.validate(o, errors);
        if (errors.hasErrors()) return;
        ActivityBulkCreateReq req = (ActivityBulkCreateReq) o;
        User user = super.getUserOrReject(req, errors);
        if (user == null) return;
        this.validateDatesOf(user, req.getDatesOfList(), errors);
        Map<LocalDate, Integer> dateOfSumWorkMinutesMap =
activityRepo.findAllByUserIdAndDateOfIn(user.getId(),
req.getDatesOfList())
                .stream()
                .collect(groupingBy(Activity::getDateOf,
                        summingInt(Activity::getDurationMin)));
        for (Map.Entry<LocalDate, Integer> dateOfSumWorkMinutes :
dateOfSumWorkMinutesMap.entrySet()) {
            LocalDate dateOf = dateOfSumWorkMinutes.getKey();
            Integer sumWorkMinutes = dateOfSumWorkMinutes.getValue();
            if (sumWorkMinutes + req.getDurationMin() >
property.getMaxWorkMinutesInDay()) {
                errors.reject("activity.ManyWorkTime",
                        messageUtil.getValidationMessage(
                                "activity.create.ManyWorkTime",
                                property.getMaxWorkHoursInDay(),
dateOf, TimeUtil.formatMinutes(sumWorkMinutes)));
```

```
                break;
            }
        }
    }

    private void validateDatesOf(User user, List<LocalDate> datesOf,
Errors errors) {
        List<ProdCalendarId> ids = datesOf
                .stream()
                .map(d -> new ProdCalendarId(d.getYear(),
d.getMonthValue()))
                .distinct()
                .collect(Collectors.toList());
        List<ProdCalendar> prodCalendars =
calendarRepo.findAllByIdIn(ids);
        List<ProdCalendarId> idsDb = prodCalendars
                .stream()
                .map(ProdCalendar::getId)
                .collect(Collectors.toList());
        List<ProdCalendarId> idsDiff = ids
                .stream()
                .filter(id -> !idsDb.contains(id))
                .collect(Collectors.toList());
        if (idsDiff.size() > 0) {
            errors.reject("activity.bulk.ProdCalendarNotExists",
                    messageUtil.getValidationMessage(
                            "activity.bulk.ProdCalendarNotExists",
idsDiff.stream()

.map(ProdCalendarId::getFormatYearMonth)
                                    .collect(Collectors.joining(",
")))));
        }
        boolean isDisabled = prodCalendars.stream().anyMatch(c ->
!c.getEnabled());
        if (isDisabled)
            errors.reject("activity.ProdCalendarDisabled",

messageUtil.getValidationMessage("activity.ProdCalendarDisabled"));
        List<OutDay> userOutDays =
outDayRepo.findAllByUserIdAndDateIn(user.getId(), datesOf);
        if (!userOutDays.isEmpty()) {
            String message = userOutDays.stream()
                    .map(outDay -> String.format(
                            "%s - %s",
outDay.getDate().format(DateTimeFormatter.ISO_DATE),
outDay.getOutType().getLabelRu()))
                    .collect(Collectors.joining(", "));
            errors.reject("activity.bulk.IsUserOutDays",

messageUtil.getValidationMessage("activity.bulk.IsUserOutDays",
message));
        }
    }
}
```

Файл «ActivityCreateReqValidator.java»

```
package ru.iteco.projload.core.activity.web.validator;

import org.springframework.stereotype.Component;
import org.springframework.validation.Errors;
import ru.iteco.projload.core.activity.Activity;
import ru.iteco.projload.core.activity.ActivityRepo;
import ru.iteco.projload.core.activity.web.ActivityCreateReq;
import ru.iteco.projload.core.contract.ContractRepo;
import ru.iteco.projload.core.prodcalendar.ProdCalendarRepo;
import ru.iteco.projload.core.user.User;
import ru.iteco.projload.core.user.UserRepo;
import ru.iteco.projload.core.useroutday.OutDayRepo;
import ru.iteco.projload.property.ApplicationProperty;
import ru.iteco.projload.util.MessageUtil;
import ru.iteco.projload.util.TimeUtil;

@Component
public class ActivityCreateReqValidator extends
ActivityBaseReqValidator {
    private final ActivityRepo activityRepo;
    private final ApplicationProperty property;
    private final MessageUtil messageUtil;

    public ActivityCreateReqValidator(final UserRepo userRepo,
                                      final ActivityRepo activityRepo,
                                      final ContractRepo contractRepo,
                                      final ApplicationProperty
property,
                                      final ProdCalendarRepo
calendarRepo,
                                      final OutDayRepo outDayRepo,
                                      final MessageUtil messageUtil) {
        super(userRepo, contractRepo, property, calendarRepo,
outDayRepo, messageUtil);
        this.activityRepo = activityRepo;
        this.property = property;
        this.messageUtil = messageUtil;
    }

    @Override
    public boolean supports(Class<?> aClass) {
        return aClass.equals(ActivityCreateReq.class);
    }

    @Override
    public void validate(Object o, Errors errors) {
        if (errors.hasErrors()) return;
        super.validate(o, errors);
        if (errors.hasErrors()) return;
        ActivityCreateReq req = (ActivityCreateReq) o;
        User user = super.getUserOrReject(req, errors);
```

```
        if (user == null) return;
        int sumWorkMinutes =
activityRepo.findAllByUserIdAndDateOf(user.getId(), req.getDateOf())
                .stream()
                .mapToInt(Activity::getDurationMin)
                .sum();
        if (sumWorkMinutes + req.getDurationMin() >
property.getMaxWorkMinutesInDay())
            errors.reject("activity.create.ManyWorkTime",
                    messageUtil.getValidationMessage(
                            "activity.create.ManyWorkTime",
                            property.getMaxWorkHoursInDay(),
req.getDateOf(), TimeUtil.formatMinutes(sumWorkMinutes)));
    }
}
```

Файл «ActivityUpdateReqValidator.java»

```
package ru.iteco.projload.core.activity.web.validator;

import org.springframework.stereotype.Component;
import org.springframework.validation.Errors;
import ru.iteco.projload.core.activity.Activity;
import ru.iteco.projload.core.activity.ActivityRepo;
import ru.iteco.projload.core.activity.web.ActivityUpdateReq;
import ru.iteco.projload.core.contract.ContractRepo;
import ru.iteco.projload.core.prodcalendar.ProdCalendarRepo;
import ru.iteco.projload.core.user.User;
import ru.iteco.projload.core.user.UserRepo;
import ru.iteco.projload.core.useroutday.OutDayRepo;
import ru.iteco.projload.error.EntityNotFoundException;
import ru.iteco.projload.property.ApplicationProperty;
import ru.iteco.projload.util.MessageUtil;
import ru.iteco.projload.util.TimeUtil;

@Component
public class ActivityUpdateReqValidator extends
ActivityBaseReqValidator {
    private final ActivityRepo activityRepo;
    private final ApplicationProperty property;
    private final MessageUtil messageUtil;

    public ActivityUpdateReqValidator(final UserRepo userRepo,
                                      final ActivityRepo activityRepo,
                                      final ContractRepo contractRepo,
                                      final ApplicationProperty
property,
                                      final ProdCalendarRepo
calendarRepo,
                                      final OutDayRepo outDayRepo,
                                      final MessageUtil messageUtil) {
        super(userRepo, contractRepo, property, calendarRepo,
outDayRepo, messageUtil);
        this.activityRepo = activityRepo;
```

```
        this.property = property;
        this.messageUtil = messageUtil;
    }

    @Override
    public boolean supports(Class<?> aClass) {
        return aClass.equals(ActivityUpdateReq.class);
    }

    @Override
    public void validate(Object o, Errors errors) {
        if (errors.hasErrors()) return;
        super.validate(o, errors);
        if (errors.hasErrors()) return;
        ActivityUpdateReq req = (ActivityUpdateReq) o;
        User user = super.getUserOrReject(req, errors);
        if (user == null) return;
        Activity activity =
activityRepo.findByIdAndDateOf(req.getId(), req.getDateOf())
                .orElseThrow(() -> new EntityNotFoundException(
                        String.format("Activity with id %s and date %s
not found ", req.getId(), req.getDateOf())));
        int sumWorkMinutes =
activityRepo.findAllByUserIdAndDateOf(user.getId(), req.getDateOf())
                .stream()
                .mapToInt(Activity::getDurationMin)
                .sum();
        if (sumWorkMinutes - activity.getDurationMin() +
req.getDurationMin() > property.getMaxWorkMinutesInDay())
            errors.reject("activity.update.ManyWorkTime",
                    messageUtil.getValidationMessage(
                            "activity.update.ManyWorkTime",
                            property.getMaxWorkHoursInDay(),
req.getDateOf(), TimeUtil.formatMinutes(sumWorkMinutes)));
    }
}
```

Файл «ActivityConverter.java»

```
package ru.iteco.projload.core.activity.web.converter;

import org.springframework.stereotype.Component;
import ru.iteco.projload.core.activity.Activity;
import ru.iteco.projload.core.activity.web.ActivityView;

@Component
public class ActivityConverter {
    private final ActivityToView toView;
    private final ActivityToViewDetail toViewDetail;

    public ActivityConverter(final ActivityToView toView, final
ActivityToViewDetail toViewDetail) {
        this.toView = toView;
        this.toViewDetail = toViewDetail;
```

```
    }

    public ActivityView toView(Activity activity) {
        return toView.convert(activity);
    }

    public ActivityView toViewDetail(Activity activity) {
        return toViewDetail.convert(activity);
    }
}
```

Файл «ActivityToView.java»

```
package ru.iteco.projload.core.activity.web.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.lang.NonNull;
import org.springframework.stereotype.Component;
import ru.iteco.projload.core.activity.Activity;
import ru.iteco.projload.core.activity.web.ActivityView;

@Component
public class ActivityToView implements Converter<Activity,
ActivityView> {

    @Override
    public ActivityView convert(@NonNull Activity a) {
        ActivityView view = new ActivityView();
        view.setAudit(a.getAudit());
        view.setContractId(a.getContractId());
        view.setDateOf(a.getDateOf());
        view.setDuration(a.getDuration());
        view.setDurationMin(a.getDurationMin());
        view.setTask(a.getTask());
        view.setVersion(a.getVersion());
        view.setId(a.getId());
        view.setUserId(a.getUserId());
        return view;
    }
}
```

Файл «ActivityToViewDetail.java»

```
package ru.iteco.projload.core.activity.web.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.lang.NonNull;
import org.springframework.stereotype.Component;
import org.springframework.util.Assert;
import ru.iteco.projload.core.activity.Activity;
import ru.iteco.projload.core.activity.web.ActivityView;
import ru.iteco.projload.core.contract.Contract;
import ru.iteco.projload.core.contract.web.ContractView;
```

```
import ru.iteco.projload.core.contract.web.converter.ContractToView;
import ru.iteco.projload.core.user.web.UserView;
import ru.iteco.projload.core.user.web.converter.UserToView;

@Component
public class ActivityToViewDetail implements Converter<Activity,
ActivityView> {
    private final ActivityToView toView;
    private final ContractToView contractToView;
    private final UserToView userToView;

    public ActivityToViewDetail(final ActivityToView toView,
                                final ContractToView contractToView,
                                final UserToView userToView) {
        this.toView = toView;
        this.contractToView = contractToView;
        this.userToView = userToView;
    }

    @Override
    public ActivityView convert(@NonNull Activity activity) {
        ActivityView view = toView.convert(activity);
        Assert.notNull(view, "ActivityView view is null!");
        Contract contract = activity.getContract();
        ContractView contractView = contractToView.convert(contract);
        view.setContract(contractView);
        UserView userView = userToView.convert(activity.getUser());
        view.setUser(userView);
        return view;
    }
}
```

Приложение Е

(обязательное)

Микросервис учета загрузки работников предприятия

Презентация

Выпускная квалификационная работа бакалавра на тему «Микросервис учета загрузки работников предприятия»

Подготовил: Горынина А.В., группа 18ВВ2
Руководитель: к.т.н., доцент каф. ВТ Синев М.П.

Проблема – неэффективное использование трудовых и материальных ресурсов

Решение - учет рабочего времени сотрудника

Дополнительные возможности программ тайм-трекеров:

- расчет показателя утилизации персонала позволяет более эффективно начислять заработную плату. За счет повышения доли оплачиваемого или признаваемого полезным времени сотрудников работодатель получает снижение издержек и себестоимости конечного продукта;
- позволяет оперативнее распределять и планировать трудовые ресурсы в зависимости от промежуточных итогов работы;
- использование тайм-трекеров дает возможность автоматизировать выставление счетов с детализацией работ клиенту;
- для руководителей учет рабочего времени дает сведения о загрузке сотрудников, что позволяет эффективнее управлять персоналом, не допуская «выгорания» или простоя.

Анализ аналогов

A2Б

- возможен сбор, ранжирование и запуск идей и рационализаторских предложений по улучшению компании, предоставляется корпоративный портал, чат;
- система включает возможности *CRM*-системы: ведение базы клиентов и управление продажами, электронный документооборот.

OfficeTime

- офлайн тайм-трекер, ПО хорошо подходит для проектов, где важна секретность;
- функция отслеживания простоев;
- расширенное описание задачи;
- таймер и ручной ввод задачи;
- присутствует функция отслеживания расходов, составления отчетов и базовых счетов.

Анализ аналогов

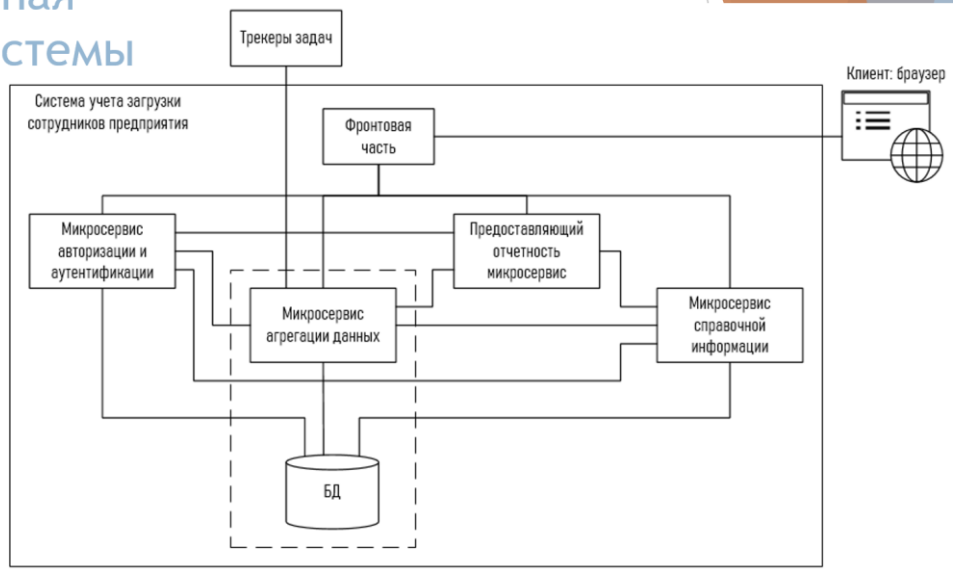
Harvest

- офлайн тайм-трекер;
- множество бухгалтерских функций;
- просмотр прогресса товарищей по команде в режиме реального времени;
- возможность расширенного описания задач;
- имеет множество интеграций со сторонними инструментами.

Kickidler

- анализ эффективности сотрудников, отчеты о динамике продуктивности сотрудников;
- кейлоггер;
- фиксация нарушений рабочего распорядка сотрудниками;
- интерфейс, дающий возможность сотрудникам видеть аналитику по собственной продуктивности.

Структурная схема системы

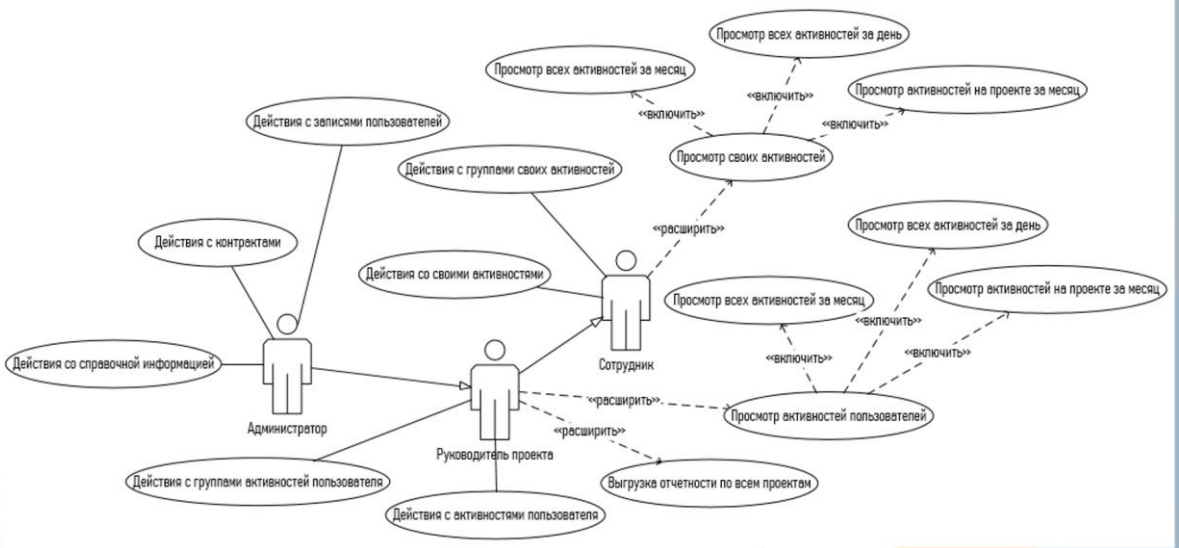


Стек технологий

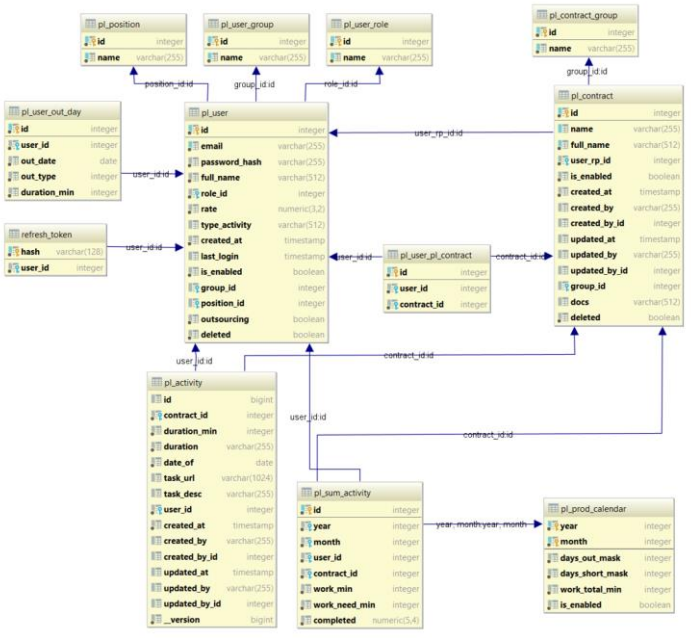
- язык программирования JAVA
- РСУБД PostgreSQL
- Spring framework
- ORM Hibernate

- IntelliJ IDEA
- Insomnia

Диаграмма прецедентов



Физическая схема базы данных



Функционал

- создание активности
- удаление активности
- удаление активностей
- создание активностей
- обновление активности
- поиск активностей
- мой календарь со сводной информацией по активностям
- мои активности за месяц сгруппированные по дням
- мои активности за день
- моя сводная информация за месяц
- календарь со сводной информацией по активностям пользователя
- активности пользователя за месяц сгруппированные по дням
- активности пользователя за день
- сводная информация пользователя за месяц

Основные функциональные блоки микросервиса

- контроллеры
- сущности
- представления
- репозитории
- сервисы

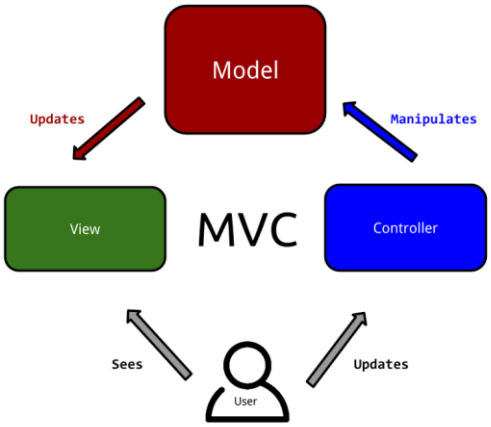
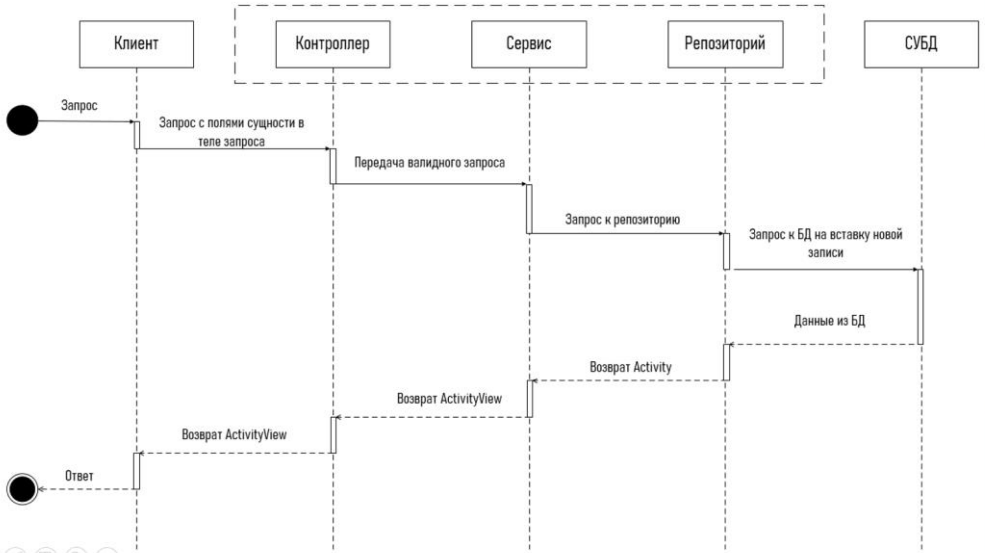


Диаграмма последовательности

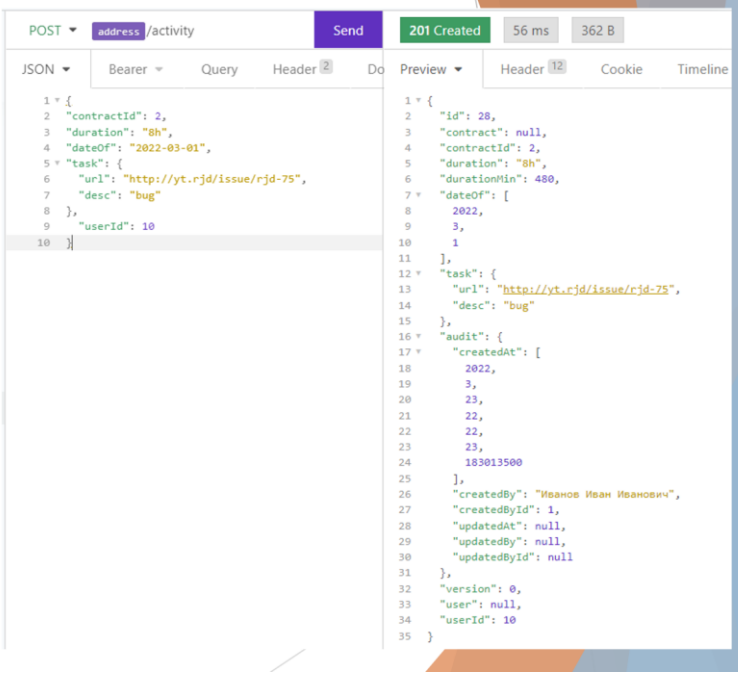


Результат выполнения запроса на создание активности

POST /project-load/user/register

POST /project-load/contract

POST /project-load/activity



Результат выполнения запроса на получение активностей за день

```
GET /project-load/activity/
user?day&dateOf=2022-03-01&userId=10
```

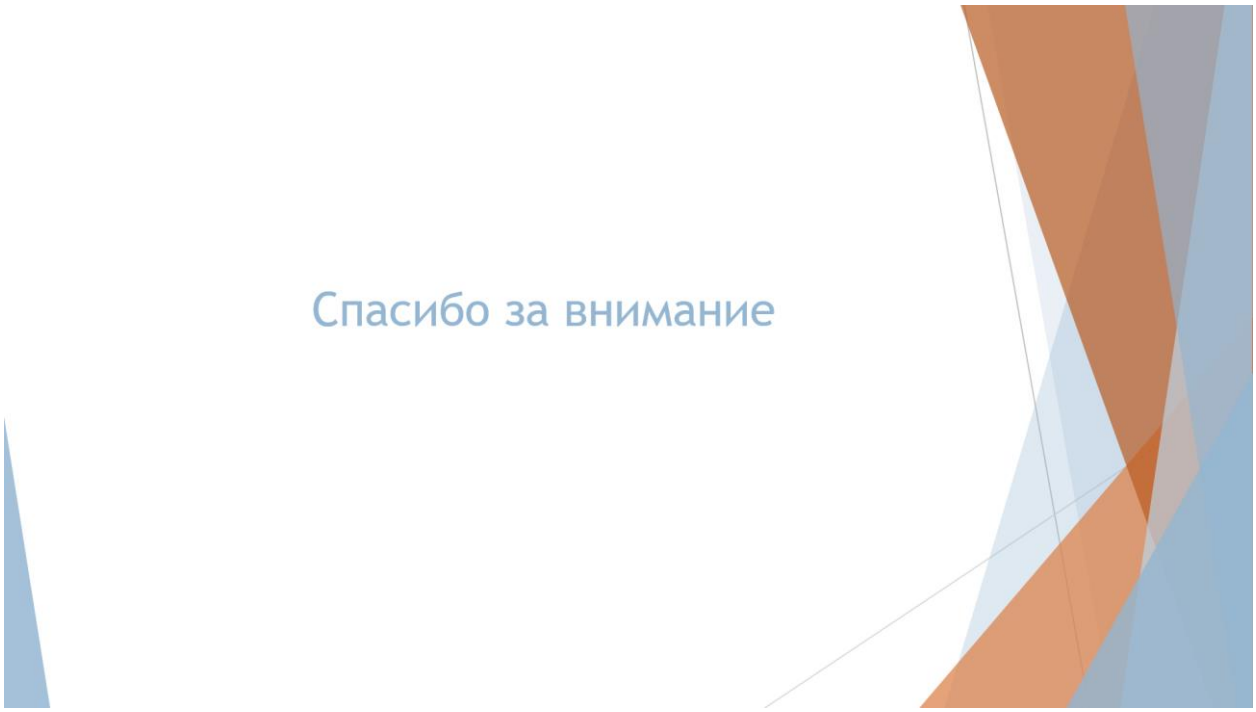
Заключение

В ходе выполнения данной работы была изучена методология проектирования приложения с микросервисной архитектурой и приобретены навыки работы с фрейворком *Spring*. При реализации поставленной задачи были получены следующие результаты:

- разработана структура базы данных для СУБД *PostgreSQL;*
- разработан прикладной интерфейс для взаимодействия с остальными микросервисами приложения (*REST API*);
- разработано и отлажено приложение учета загрузки сотрудников.

Сервис реализует полный функционал для интеграции с остальными микросервисами внутрикорпоративного продукта.

В дальнейшем в проект предстоит ввести интеграцию с часто используемыми трекерами задач: *Jira*, *Redmine*, *YouTrack*.


Спасибо за внимание