Содержание

Введение

Информационные технологии все больше и больше затрагивают сферы деятельности человека. И сейчас, под натиском информационных и телекоммуникационных технологий необходимо введение информационных систем в те области, где они не применяются или слабо развиты и которые могут уменьшить затраты, время на обработку данных и увеличить производительность труда.

Томсксофт занимается разработкой программного обеспечения.

Самые крупные проекты связаны с передачей и обработкой мультимедиа информации, формированием соцсетей для общения между пользователями этих проектов и технологическими инструментами для обработки мультимедиа. Каждый из таких проектов содержит несколько компонентов, включая, но не ограничиваясь:

1. Приложения для десктопных компьютеров;
2. ВЕБ-сайты и сервисы;
3. Сетевые сервисы, обслуживающие до нескольких сотен тысяч онлайн пользователей одновременно;
4. Приложения для мобильных ОС;
5. Системы мониторинга и защиты от сетевых атак.

Компания состоит из следующих отделов:

1. Администрация;
2. Отдел сетевых разработок;
3. Отдел разработки приложений;
4. Отдел системного администрирования и мониторинга;
5. Multimedia development department;
6. Отдел дизайна;

7. Отдел качества ПО и документирования;
8. Рабочие группы по проектам.

Каждая команда раз в неделю собирается в Zoom конференции, с участием представителя руководства компании для разбора сделанных за прошедшую неделю задач, решённых и нерешенных проблем, планирования действий на неделю. Остальные организационные принципы устанавливаются конкретной рабочей группой самостоятельно.

Объект и предмет проекта:

Объект: Информационное взаимодействие внутренних сервисов компании Томсксофт с единой внутрикорпоративной системой.

Предмет: Интеграция взаимодействия внутренних сервисов компании Томсксофт с единой внутрикорпоративной системой.

Цель проекта:

Разработка информационной системы для организации канала связи внутренних сервисов компании Томсксофт с внутрикорпоративной системой.

Задачи:

1. Изучить особенности взаимодействия сервисов с внутрикорпоративной системой;
2. Проанализировать возможности автоматизации данного процесса;
3. Рассмотреть существующие варианты программных продуктов, автоматизирующие данный процесс;
4. Определить основные технические и функциональные требования к разрабатываемой системе;
5. Выполнить работу по проектированию ИС с учётом заявленных требований;
6. Разработать спроектированную информационную систему;
7. Разработать документацию к API.

Практическая значимость и ожидаемые результаты

В результате выполнения проекта, будет разработана информационная система, позволяющая сторонним сервисам обращаться к системе для автоматизации комплекса используемых компании сервисов.

1 ОБЩАЯ ЧАСТЬ

1.1 Анализ предметной области

В компании Томсксофт имеется множество сотрудников. Для учёта рабочего процесса и времени которых применяется внутренняя intranet-система.

Данная система разработана для

1. Размещения внутрикорпоративной информации;
2. Автоматизированной рассылки данных;
3. Оповещения сотрудников посредством электронной почты;
4. Размещение файлов;
5. Хранение учётных данных внешних корпоративных сервисов (логины и пароли);
6. Предупреждение коллег о изменении рабочего времени;
7. Ведение информации о проектах;
8. Ведение информации о запланированной и проделанной работе.

Также система содержит в себе вывод простых отчётов на основе фильтров на web-страницу, после чего пользователь может использовать полученные данные для своих нужд:

1. Бухгалтер для расчёта зарплаты;
2. Администрация для учёта рабочего времени и отпусков;
3. Руководители проектов для контроля прогресса разработки.

С развитием сторонних систем надобность в ручном расчёте постепенно пропадает и всю работу пользователей составляет рутинный перенос данных из web-страницы в сторонний сервис.

К сторонним сервисам можно отнести:

1. 1С для расчёта заработной платы бухгалтерией;
2. 1С для расчёта отпусков администрацией;
3. Почтовая служба для рассылок писем;
4. Планируемая служба для автоматической генерации записей и проверки рабочего времени на основе внешних источников.

Исходя из всего вышеперечисленного было принято решение создать систему для интеграции данных компании со сторонними сервисами с целью автоматизации обработки и передачи данных.

1.2 Средства и среда разработки

1.2.1 Теоретическая часть

Прежде чем приступить к рассмотрению средств разработки, которые были применены для создания программы, необходимо определиться с основными понятиями и терминами.

Разработка программ — сложный процесс, основной целью которого является создание, сопровождение программного кода, обеспечивающего необходимый уровень надёжности и качества. Для достижения основной цели разработки программ используются средства разработки программного обеспечения.

Средства разработки программного обеспечения — совокупность приёмов, методов, методик, а также набор инструментальных программ (компиляторы, прикладные/системные библиотеки и т.д.), используемых разработчиком для создания программного кода программы, отвечающего заданным требованиям.

Основные средства, используемые на этапах разработки программы:

- Проектирование приложения;
- Реализация программного кода приложения;
- Тестирование приложения.

Проектирование приложения — это процесс создания проекта программного обеспечения (ПО), целью которого является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе выданных заказчиком требований к ПО. Эти требования подвергаются анализу. Средство содержит в себе наличие технического задания, эскизного и технического проекта и только потом рабочего проекта.

Реализация программного кода приложения — это процесс создания кода компонентов программного обеспечения на выбранном языке программирования. При процедурном подходе реализация заключается в программировании функций и файлов (модулей).

Объектно-ориентированное программирование (ООП) — методология программирования, совокупности основанная объектов, на каждый представлении из которых программы является в виде экземпляром определённого класса, а классы образуют иерархию наследования.

Интегрированная среда разработки программного обеспечения— система программных средств, используемая для разработки программного обеспечения.

Обычно среда разработки включает в себя текстовый редактор, компилятор и/или интерпретатор, средства автоматизации сборки и отладчик. Иногда также содержит средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения разработки приложения.

Язык программирования — формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих

внешний вид программы и действия, которые выполнит исполнитель (ЭВМ) под её управлением.

Фреймворк — это программный продукт, который упрощает создание и поддержку технически сложных или нагруженных проектов. Фреймворк, как правило, содержит только базовые программные модули, а все специфичные для проекта компоненты реализуются разработчиком на их основе. Тем самым достигается не только высокая скорость разработки, но и большая производительность и надёжность решений.

СУБД (Система управления базами данных) — комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система предоставляет средства для администрирования БД.

Информационная система — система, предназначенная для хранения, поиска, обработки информации и соответствующие организационные ресурсы.

CRUD — акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание, чтение, модификация, удаление.

1.2.2 Выбор языка программирования

При разработке информационной системы был использован такой язык программирования, как PHP8.

PHP (рекурсивный акроним словосочетания PHP: Hypertext Preprocessor) - это распространённый язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в тело ответа.

PHP крайне прост для освоения, но вместе с тем способен удовлетворить запросы профессиональных программистов.

Главная область применения PHP - написание скриптов, работающих на стороне сервера; таким образом, PHP способен обрабатывать данные форм, генерировать динамические страницы или отсылать и принимать cookies. Но PHP способен выполнять намного больше.

Существуют три основных области применения PHP:

1. Создание скриптов для выполнения на стороне сервера;
2. Создание скриптов для выполнения в командной строке;
3. Создание оконных приложений, выполняющихся на стороне клиента.

PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix (такие как HP-UX, Solaris и OpenBSD), Microsoft Windows, macOS, RISC OS и многие другие. Также в PHP включена поддержка большинства современных веб-серверов, таких как Apache, IIS и многих других.

1.2.3 Среда разработки

Для реализации кода была использована такая среда разработки, как Eclipse IDE.

Преимущества среды разработки:

1. Простая в установке и использовании;
2. Настраиваемый графический интерфейс;
3. Интуитивно понятный интерфейс;
4. Работа с множеством проектов;
5. Как программное обеспечение с открытым исходным кодом доступна бесплатно (в отличие от конкурентов, таких как IntelliJ IDEA);

6. Предоставленные функции, утилиты и автодополнение облегчают написание кода;

7. Ускоряет разработку приложений и повышает эффективность работы команды программистов;

8. Благодаря разнообразию плагинов тонко настраивается и расширяется дополнительными функциями;

9. Доступна для любой платформы;

10. Из-за многолетнего существования среды в сети находится многочисленные документация, советов и хитростей по использованию;

11. Большое сообщество разработчиков помогает с решением вопросов и проблем на форумах;

12. Предоставляет встроенные локальные серверы, где разработчики развёртывают и тестируют приложения, прежде чем отправлять в другие среды;

13. Сохраняет и восстанавливает сессии.

1.2.4 Выбор фреймворка

При разработке информационной системы был использован фреймворк Laravel.

Laravel — бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC (англ. Model View Controller — модель-представление-контроллер). Laravel выпущен под лицензией MIT.

Фреймворк отлично документирован: документация есть ко всему и на нескольких языках, в том числе на русском. Также каждый метод имеет шапку в PHPDoc. Помимо документации, есть множество руководств и форумов, помогающих разобраться с возникшими проблемами.

Можно выделить следующие плюсы:

1. Установка с помощью Composer;
2. Высокая производительность;
3. Встроенная валидация форм;
4. Возможность подключения сторонних библиотек;
5. Использование миграций баз данных;
6. Поддержка автоматического модульного и интеграционного тестирования;
7. CLI для автоматизации разработки.

Также рассматривался фреймворк Yii2.

Yii — объектно-ориентированный компонентный фреймворк, реализующий парадигму MVC.

Плюсы:

1. Легко изучается, низкий старт разработки;
2. Имеет множество встроенных решений для интерфейсов (встроенное использование фреймворка Bootstrap);

3. Генератор моделей, контроллеров И CRUD операций.

Минусы:

1. Преимущественно маршруты формируются структурой классов и файлов;
2. Дублирование кода при назначении модификаторов доступа;
3. Медленное развитие;
4. Сильно связанные клиентские и серверные части приложения.

Yii подходит на проекты с быстрым жизненным циклом благодаря генератору кода, готовыми виджетами и фреймворку Bootstrap3. Это полезно только на первых этапах развития приложения, данные аспекты перестают быть плюсом за ненадобностью. Однако минусы не исчезают, Yii становится слишком неповоротливым на развивающихся проектах и приложение становится слишком дорогим в обслуживании.

Исходя из всего вышеперечисленного, было принято решение использовать Laravel.

1.2.5 Библиотеки

Для разработки информационной системы использовались библиотеки Laravel Sanctum для авторизации и Eloquent power joins для построения запросов к БД.

Laravel Sanctum предлагает легковесную систему аутентификации для SPA (одностраничных приложений), мобильных приложений и простых API на основе токенов. Sanctum позволяет каждому пользователю вашего приложения создавать несколько токенов API для своей учетной записи. Этим токенам могут быть предоставлены полномочия / области, которые определяют, какие действия токенам разрешено выполнять.

Для запросов к БД в дополнение к Eloquent ORM использовался Eloquent power joins, являющийся построителем join-запросов для моделей, основанных на Eloquent.

В данном подходе можно выделить следующие плюсы:

1. Решение проблемы N+1: когда для одной записи данных требуется дополнительно загрузить ещё N записей.

2. Синтаксис методов предоставляет простой способ создания join запросов, которые не реализованы в Eloquent. Данные запросы реализованы в построителе запросов, что использует Eloquent, но слишком громоздки для использования.

2 СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1 Описание требований к информационной системе

Данные обрабатываемые приложением должны храниться в базе данных, которая должна совпадать с существующей и готова к дальнейшим изменениям.

Данные для аутентификации сотрудников должны быть использованы из LDAP сервера.

API должно соответствовать требованиям заказчика.

В информационной системе должны быть реализованы следующие функции:

Описание функций неавторизованного пользователя

Авторизация

Входные данные:

1. Ник;
2. Пароль.

Выходные данные:

1. Токен для взаимодействия с API.

Описание поведения:

1. Считывание данных, отправленных в запросе.
2. Проверка данных. Если имеются не поля с неверным форматом данных, то возвращается HTTP код ошибки и предупреждающее сообщение в теле ответа.
3. Если пользователь не смог войти, то возвращается HTTP код ошибки и предупреждающее сообщение в теле ответа.

4. Если пользователь успешно вошёл, то выдаётся токен пользователя для дальнейшего взаимодействия с API.

Описание функций авторизованного пользователя

Для авторизации необходимо предоставить токен доступа в заголовке X-Auth-Key.

Создание токенов для внешних сервисов

Входные данные:

1. Название токена;
2. Предоставляемые права.

Выходные данные:

1. Токен.

Описание поведения:

1. Создание токена с заданными именем и правами.
2. Отправка токена пользователю.

Просмотр созданных токенов пользователя

Входные данные: нет.

Выходные данные:

1. Список имён токенов и их прав.

Описание поведения:

1. Выборка токенов пользователя по данным аутентификации.
2. Отправка пользователю.

Просмотр токена

Входные данные:

1. Название токена.

Выходные данные:

1. Имя токена;
2. Права;
3. Дата и время изменения;
4. Дата и время последнего использования;

Описание поведения:

1. Выборка токена пользователя по его имени.
2. Отправка пользователю.

Редактирование токена

Входные данные:

1. Старое название токена;
2. Новое название токена;
3. Список прав токена.

Выходные данные:

1. Имя токена;
2. Права;
3. Дата и время последнего использования.

Описание поведения:

1. Выборка токена пользователя по его имени.
2. Изменение токена.
3. Если пользователь не обладает правами, которые хочет назначить токену, то эти права игнорируются и выдаются соответствующие сообщения.
4. Отправка ответа пользователю.

Удаление токена

Входные данные:

1. Название токена.

Выходные данные: нет.

Описание поведения:

1. Выборка токена пользователя по его имени.
2. Удаление токена.
3. Отправка ответа пользователю.

Получение контактов пользователя

Входные данные:

1. Ник пользователя.

Выходные данные:

Список контактов пользователя, где каждый элемент содержит:

1. Канал связи или тип контакта;
2. Текстовое представление контакта для связи.

Описание поведения:

1. Выборка всех контактов пользователя.
2. Группировка контактов по их типу.
3. Отправка ответа пользователю.

Отчёт по часам

Входные данные:

1. Дата начала отчётного периода;
2. Дата конца отчётного периода;

Выходные данные — список, где каждый элемент содержит:

1. Ник сотрудника;
2. ФИО сотрудника;
3. Департамент;
4. Сумма отработанных часов за заданный период.

При этом один сотрудник входит в список только один раз.

Описание поведения:

1. Выборка данных для отчёта.

2. Отправка результата.

Отчёт по часам, сгруппированный по проектам

Входные данные:

1. Дата начала отчётного периода;
2. Дата конца отчётного периода.

Выходные данные — список, где каждый элемент содержит:

1. Ник сотрудника;
2. ФИО сотрудника;
3. Департамент;
4. Данные проекта (id и название);
5. Данные заказчика (id и название);
6. Сумма отработанных часов за заданный период.

При этом один сотрудник входит в список только один раз.

Описание поведения:

1. Выборка данных для отчёта.
2. Отправка результата.

Получение расписания официальных рабочих дней

Входные данные:

1. Дата начала отчётного периода;
2. Дата конца отчётного периода.

Выходные данные — список дней, где каждый элемент содержит:

1. Статус дня: рабочий, выходной или конфликтный;
2. Дополнительная информация о причине изменений.

Описание поведения:

1. Выборка диапазонов выходных из БД.
2. Формирование расписания по дням.
3. Отправка результата.

Получение расписания рабочих дней сотрудников

Входные данные:

1. Дата начала отчётного периода;
2. Дата конца отчётного периода.

Выходные данные — список дней, сгруппированных по сотрудникам. Где каждый день содержит:

1. Статус дня: рабочий, выходной или конфликтный;
2. Дополнительная информация о причине изменений;
3. Сумма отработанных сотрудником часов за данный день.

Описание поведения:

1. Получение расписания официальных рабочих дней.
2. Выборка диапазон выходных дней сотрудников из БД и дополнения ими расписания каждого сотрудника.
3. Формирование расписания по дням
4. Дополнение расписания данными о выполненной работе.
5. Отправка результата.

Функционал администратора системы

Администратор системы имеет доступ к интерфейсу командной строки приложения и позволяет выполнять следующие действия:

— Создание пользователя для сервиса;

— Создание токена стороннему пользователю;

— Автоматическое тестирование системы.

2.1.1 Описание требований к среде выполнения

Требования серверной части

Операционная система:

- Ubuntu.

Версии ПО:

- Appache 2.4.41;
- PHP 8;
- MySQL 8.0.22.

Аппаратное обеспечение:

- Процессор – Intel Core i3, частота 2GHz;
- Оперативная память – 6 Гб;
- Жесткий диск – 1ТБ.

Требования клиентской части

- Возможность отправлять HTTP запросы;
- Поддержка JSON.

2.2 Диаграммы вариантов использования

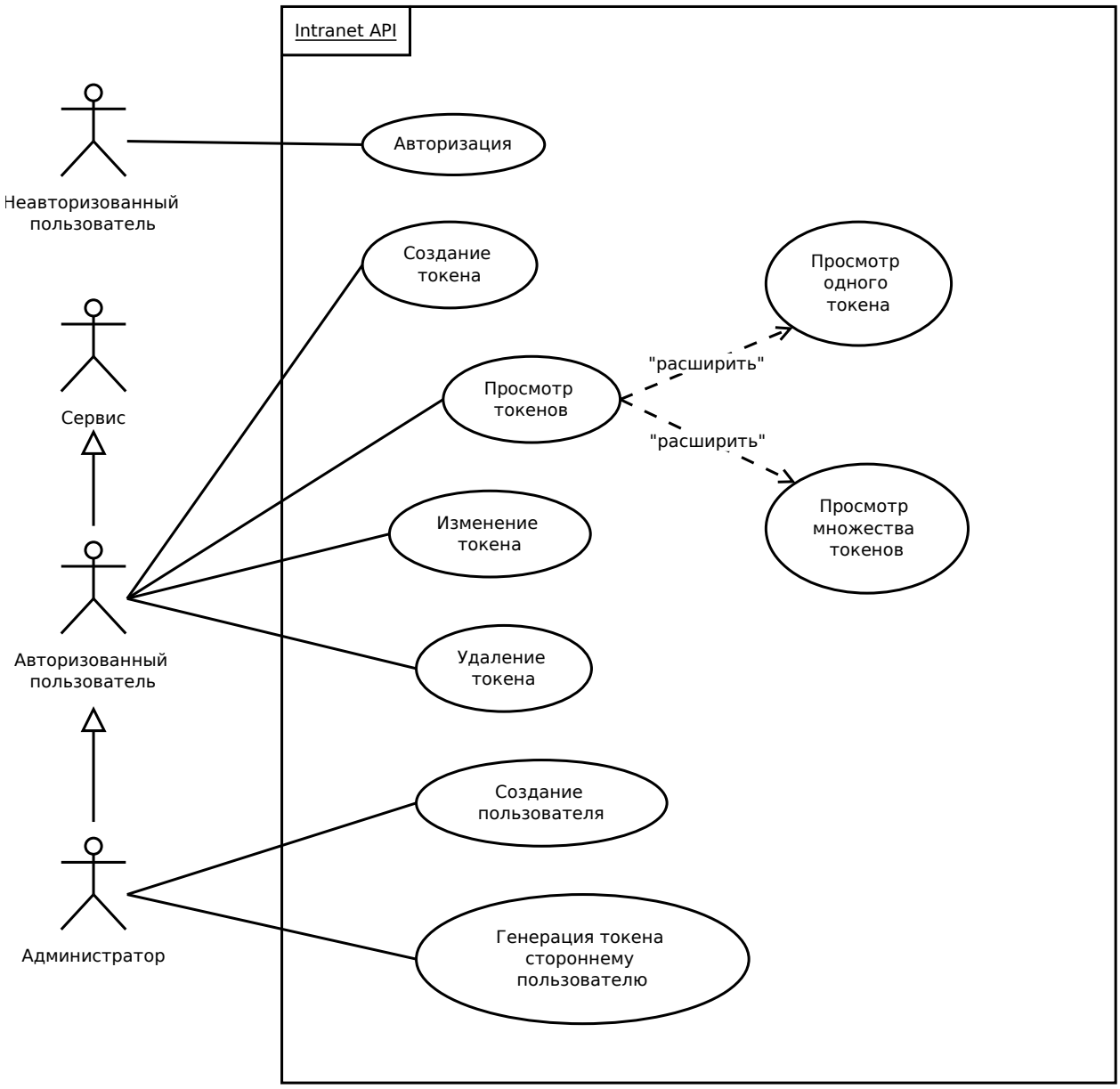

Рисунок 1. Диаграмма вариантов использования людьми

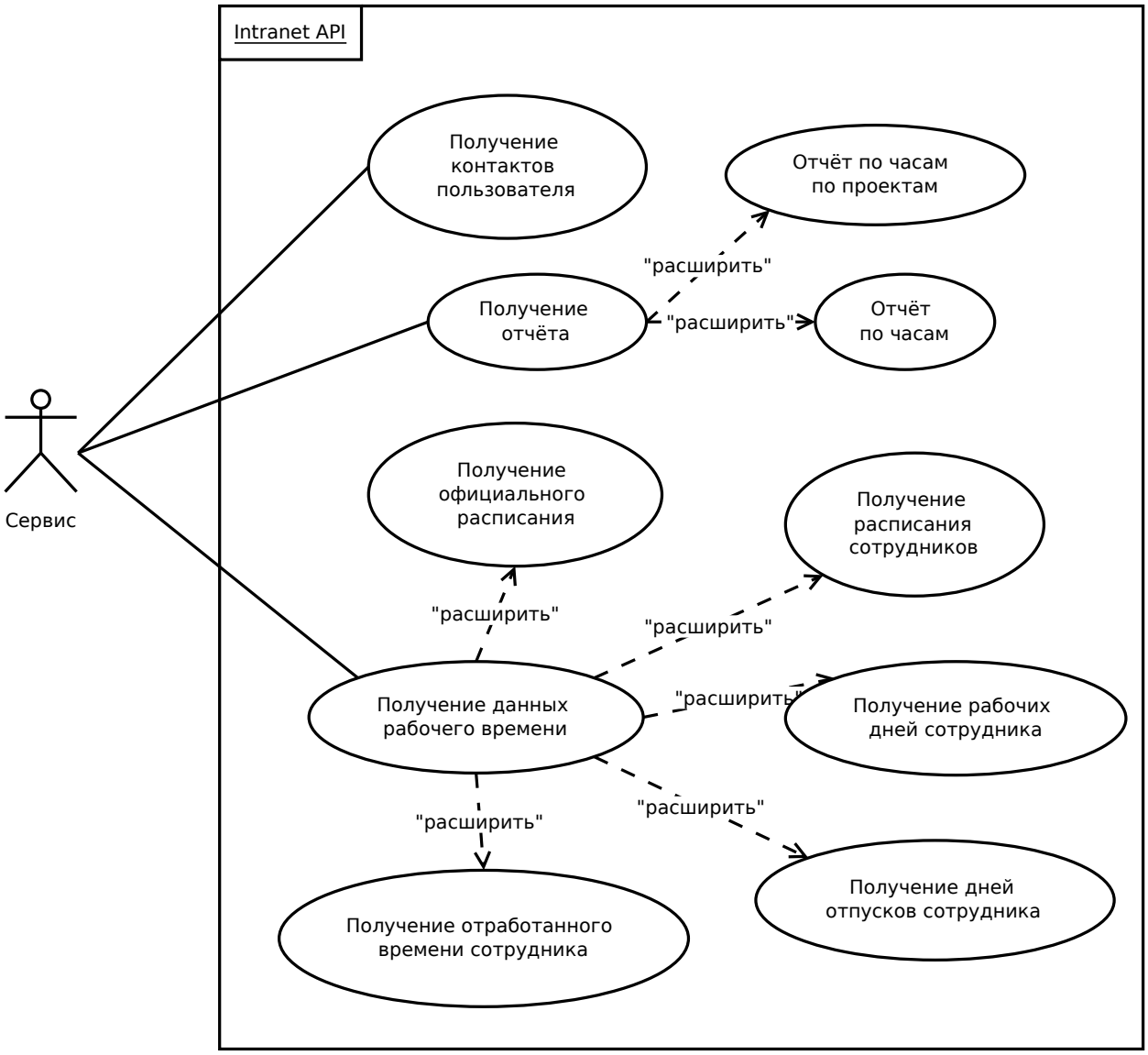

Рисунок 2. Диаграмма вариантов использования сторонними сервисами

2.3 Диаграммы состояний

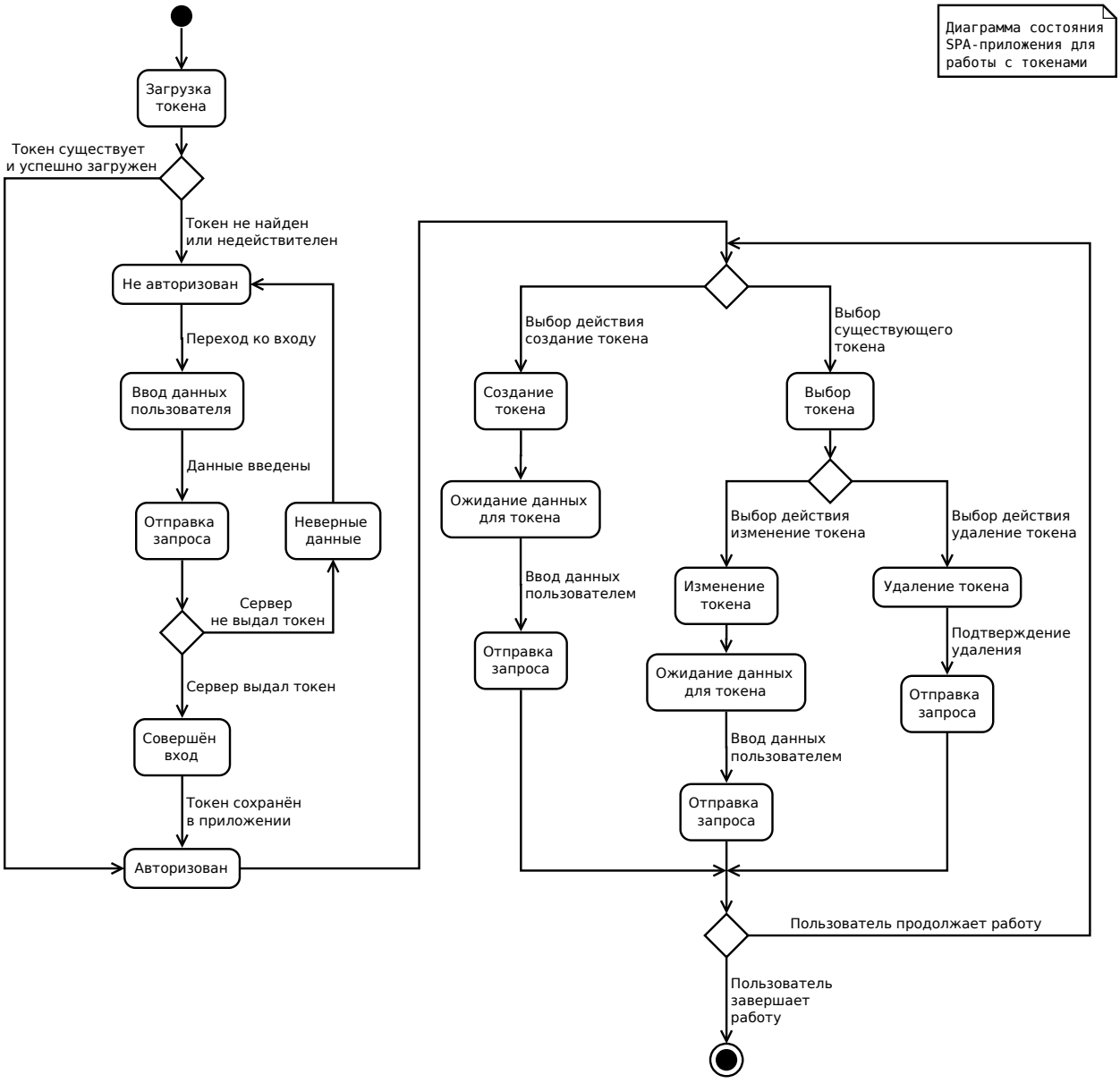

Рисунок 3. Диаграмма состояний абстрактного SPA

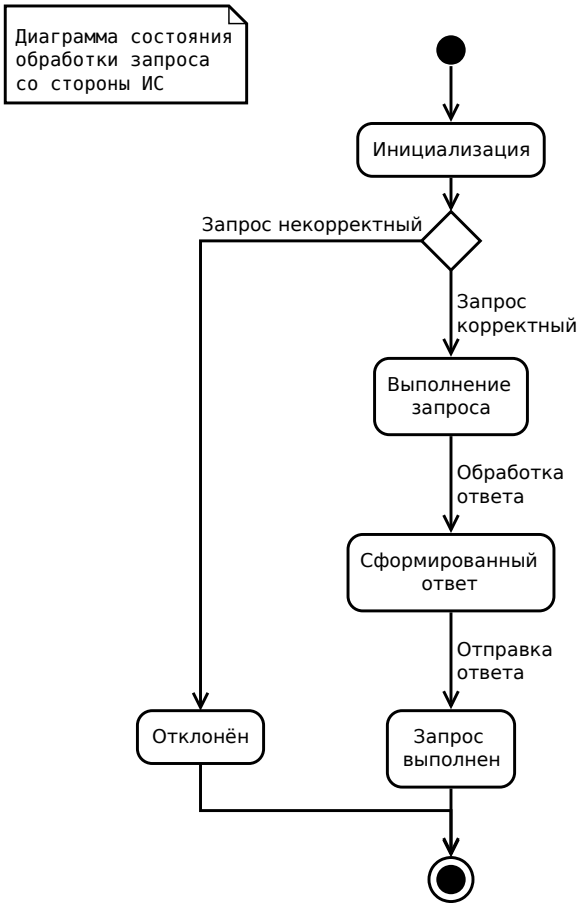

Рисунок 4. Общая диаграмма состояний

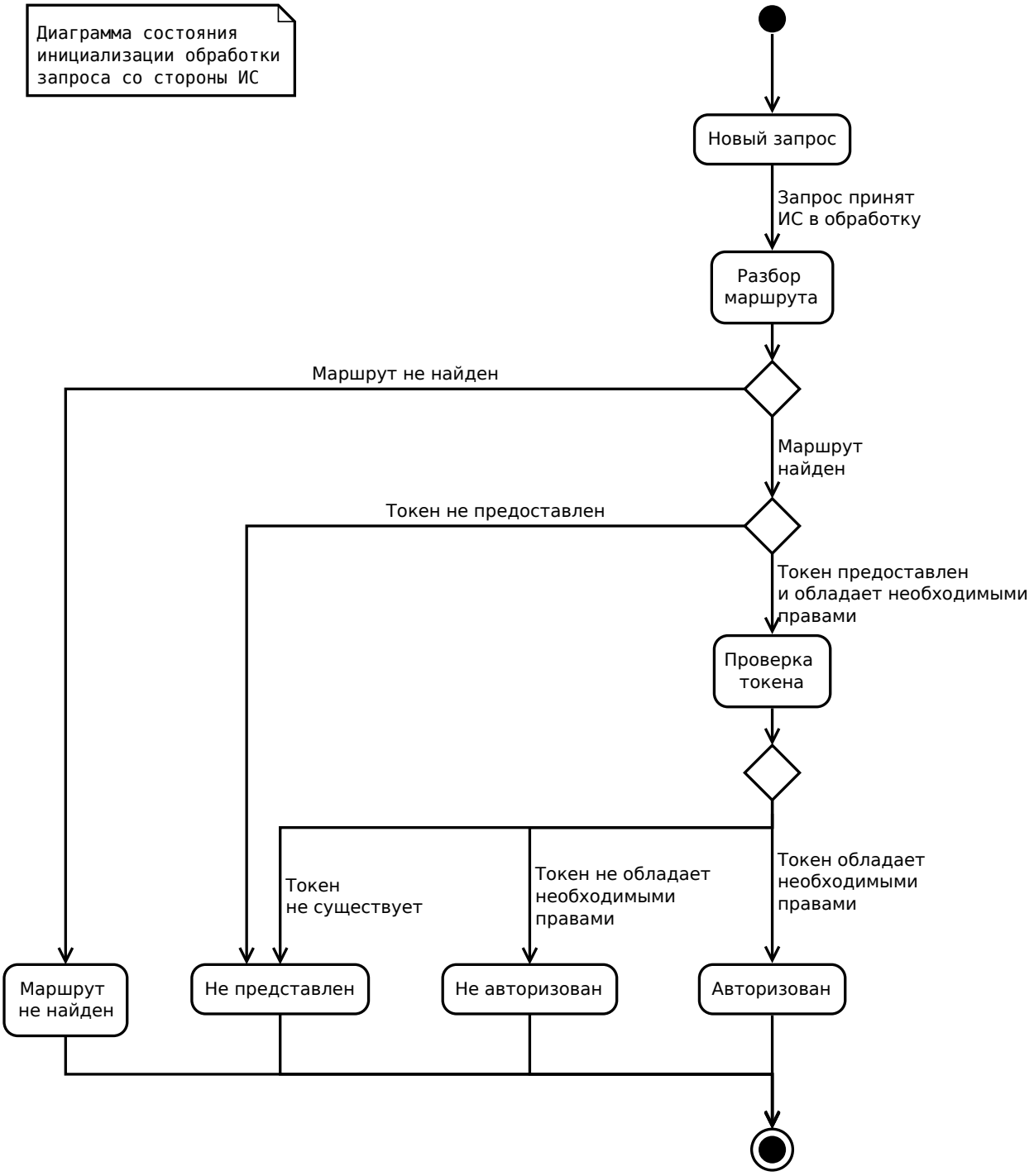

Рисунок 5. Диаграмма состояний начального разбора запроса

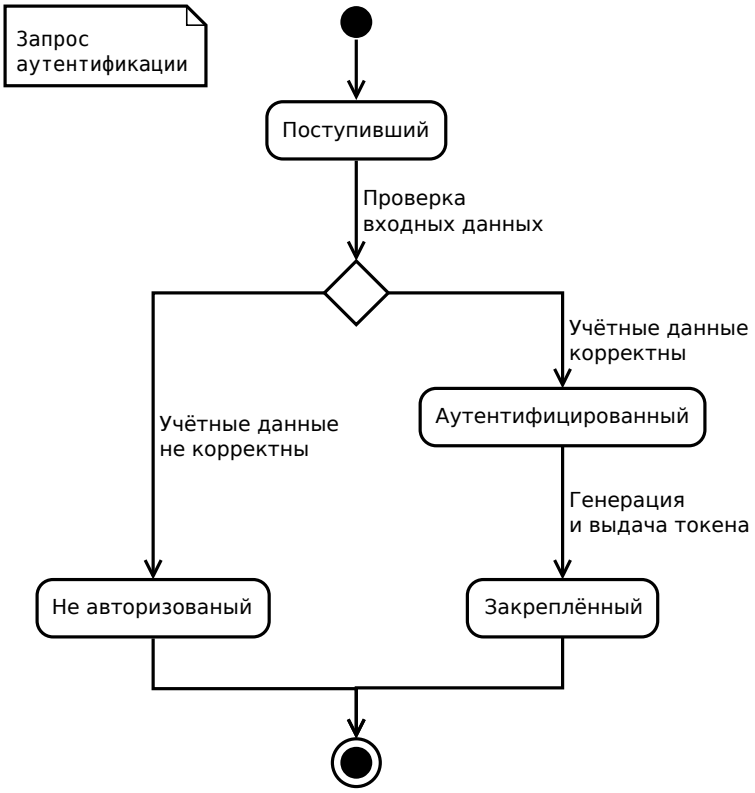

Рисунок 6. Диаграмма состояний аутентификации

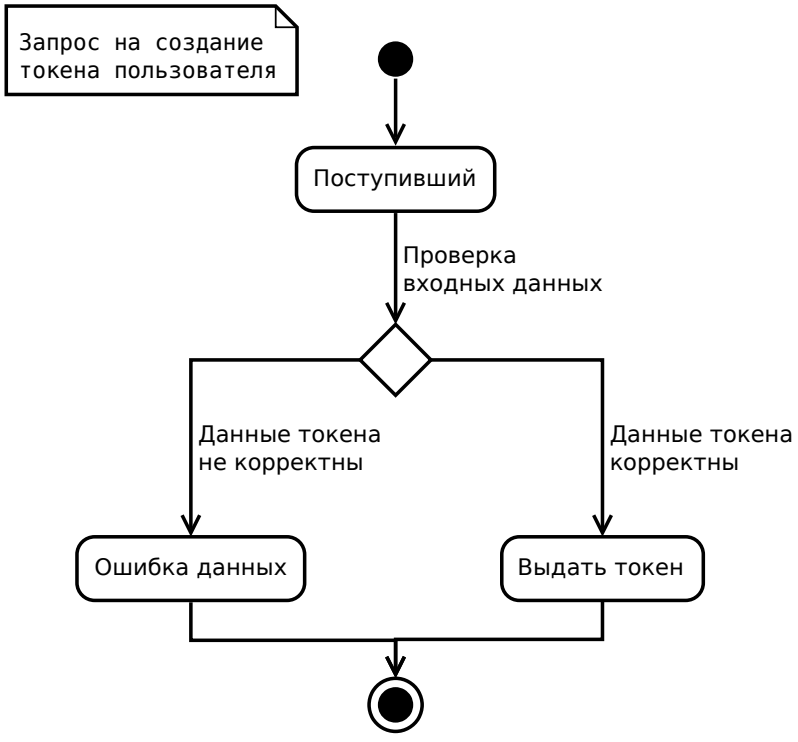

Рисунок 7. Диаграмма состояний создания токена

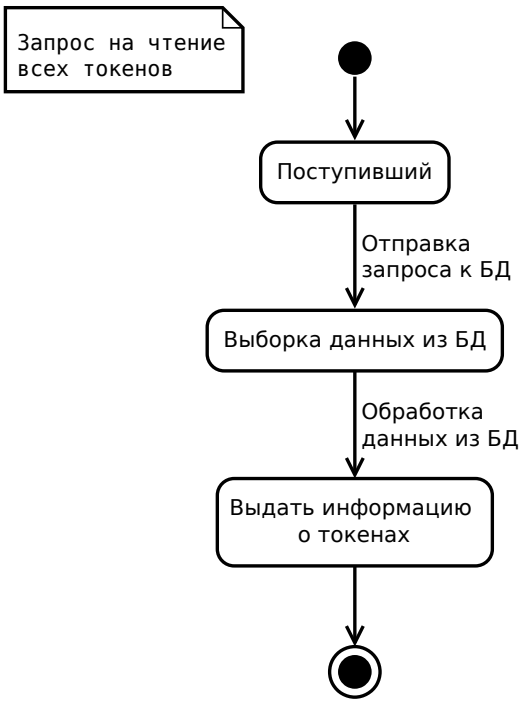

Рисунок 8. Диаграмма состояний чтения всех токенов

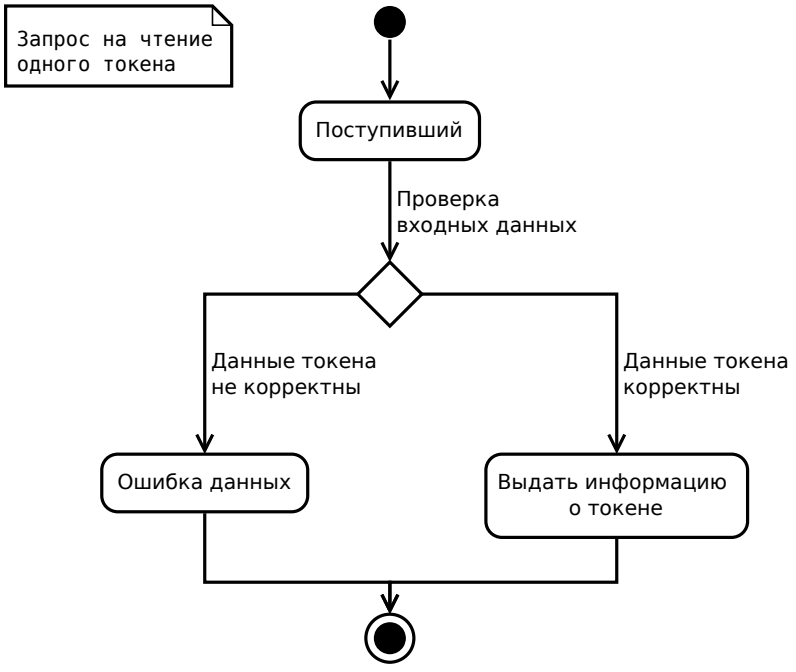

Рисунок 9. Диаграмма состояний чтения одного токена

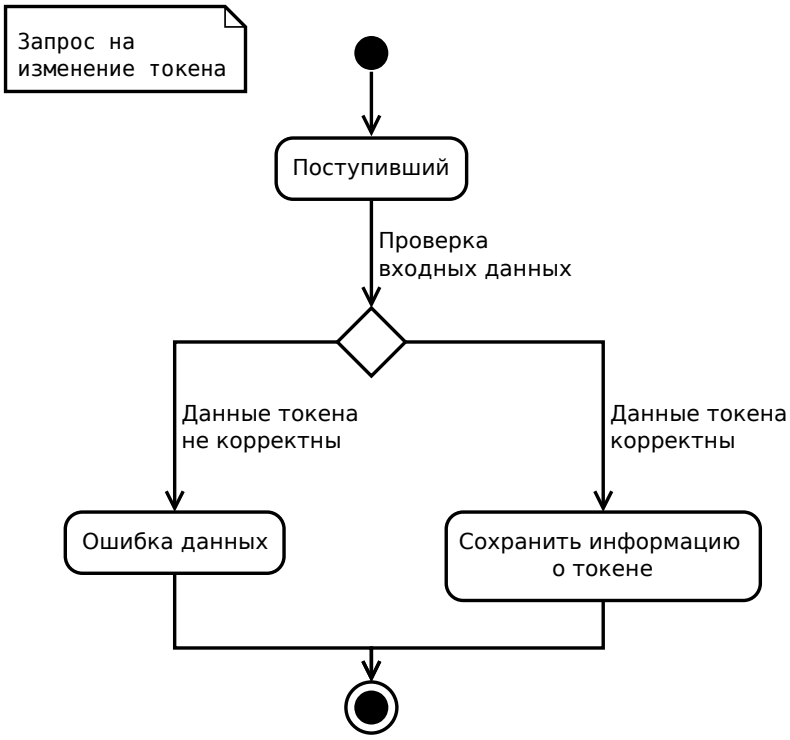

Рисунок 10. Диаграмма состояний изменения токена

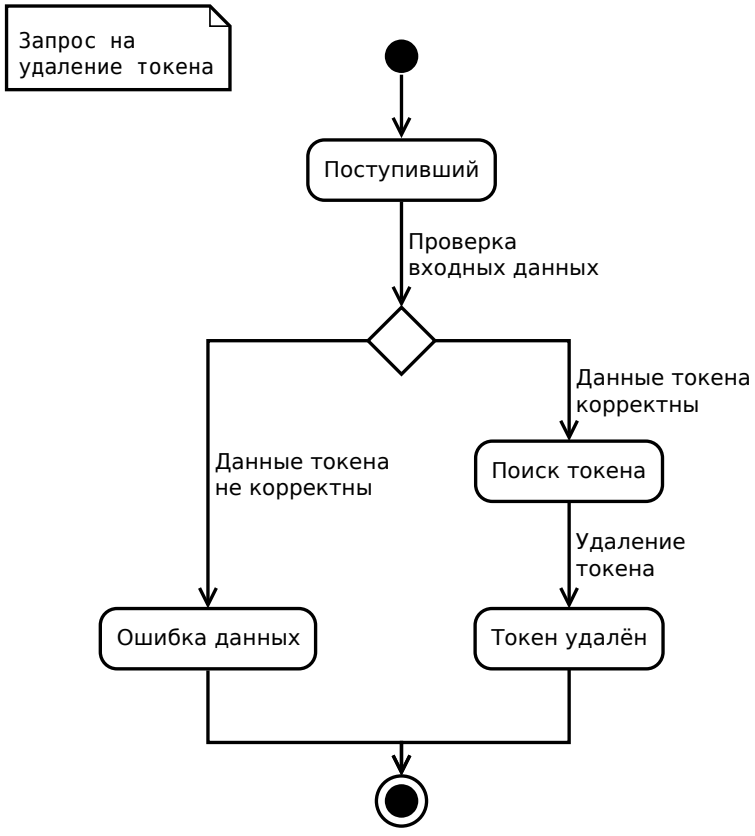

Рисунок 11. Диаграмма состояний удаления токена

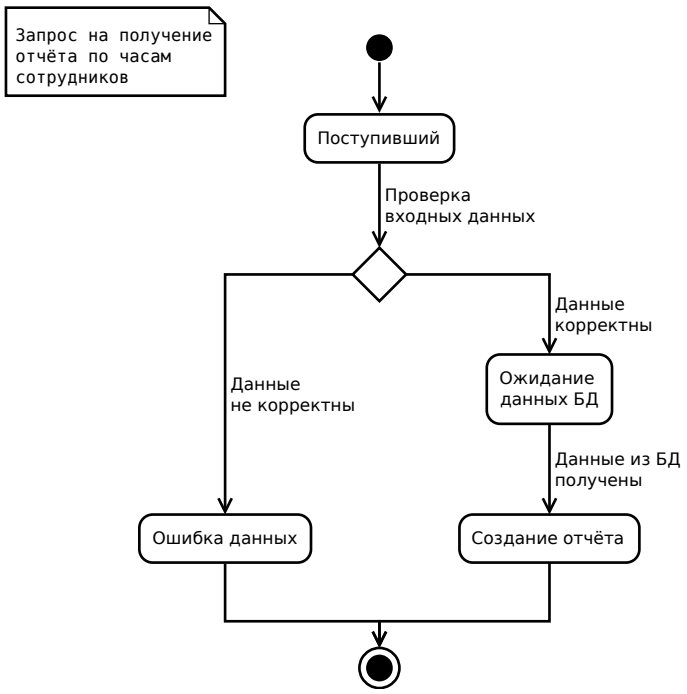

Рисунок 12. Диаграмма состояний получения отчёта по часам

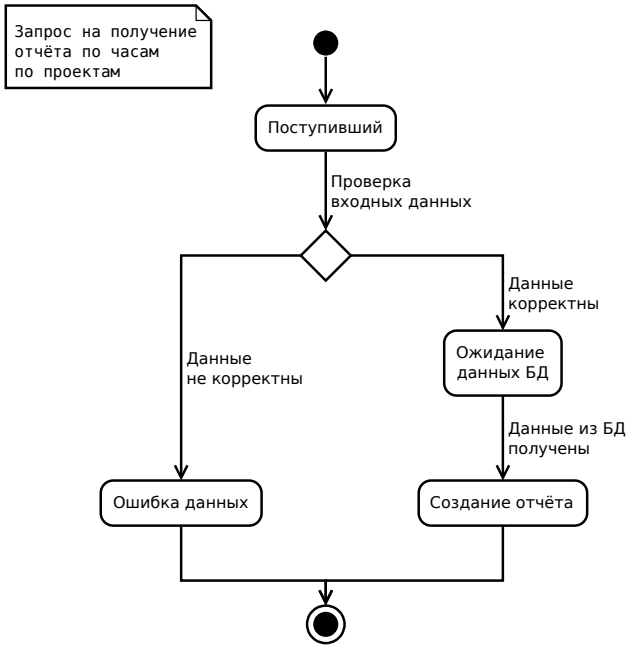

Рисунок 13. Диаграмма состояний получения отчёта по проектам

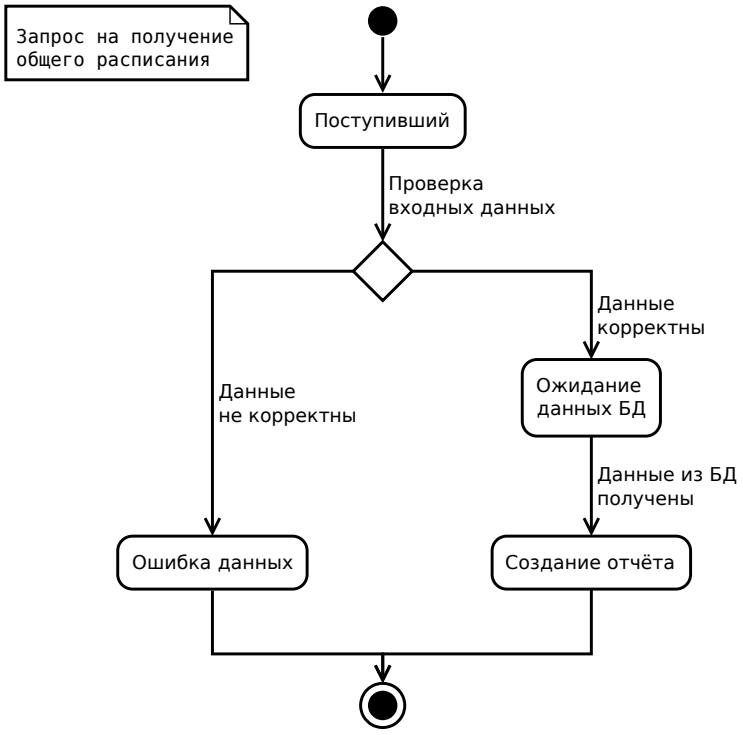

Рисунок 14. Диаграмма состояний получения общего расписания

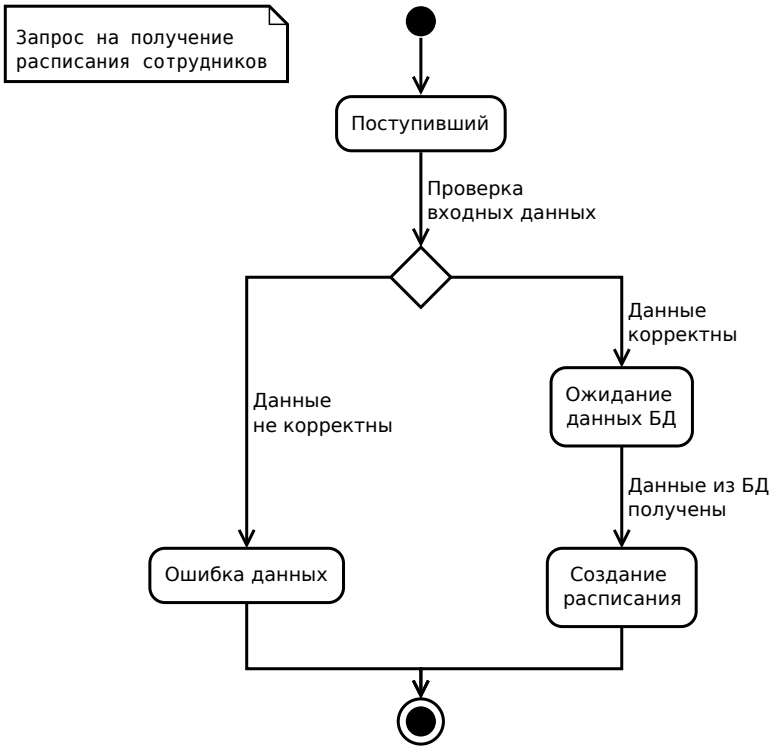

Рисунок 15. Диаграмма состояний получения расписаний сотрудников

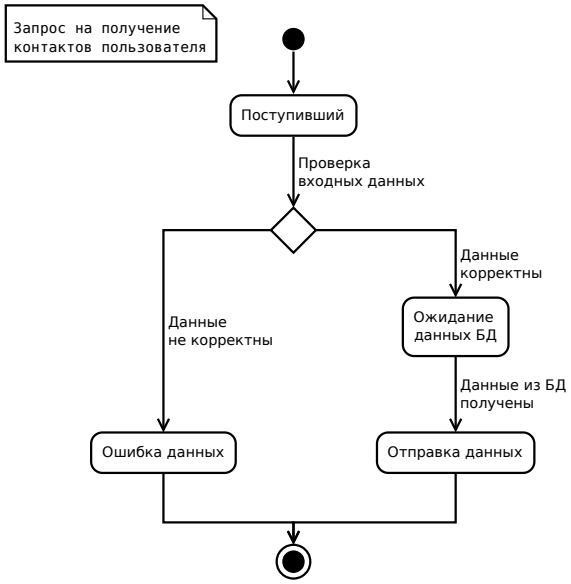

Рисунок 16. Диаграмма состояний получения контактов пользователя

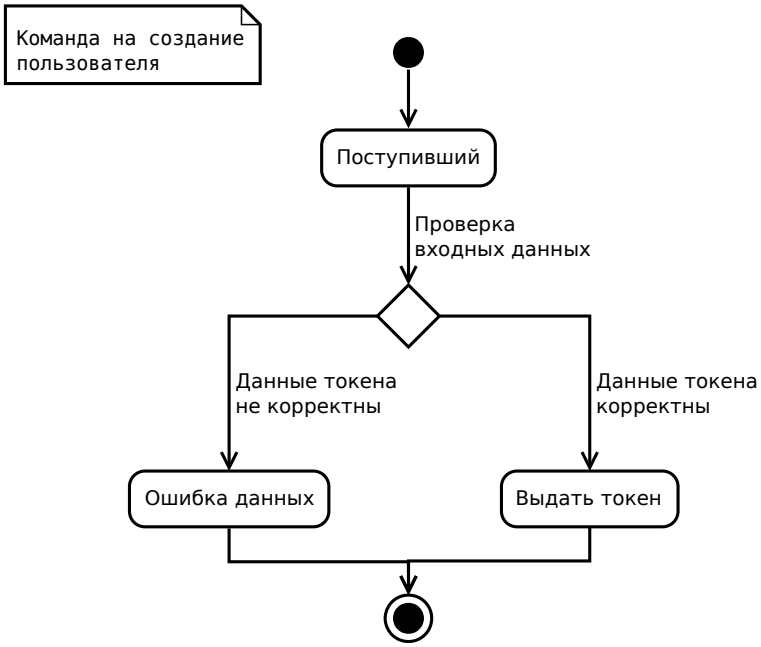

Рисунок 17. Диаграмма состояний создания пользователя

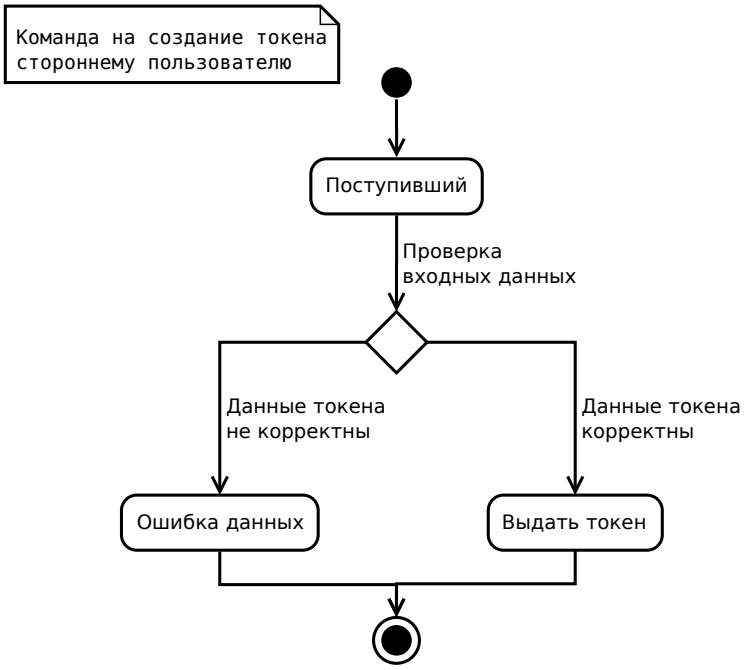

Рисунок 18. Диаграмма состояний создания токена администратором

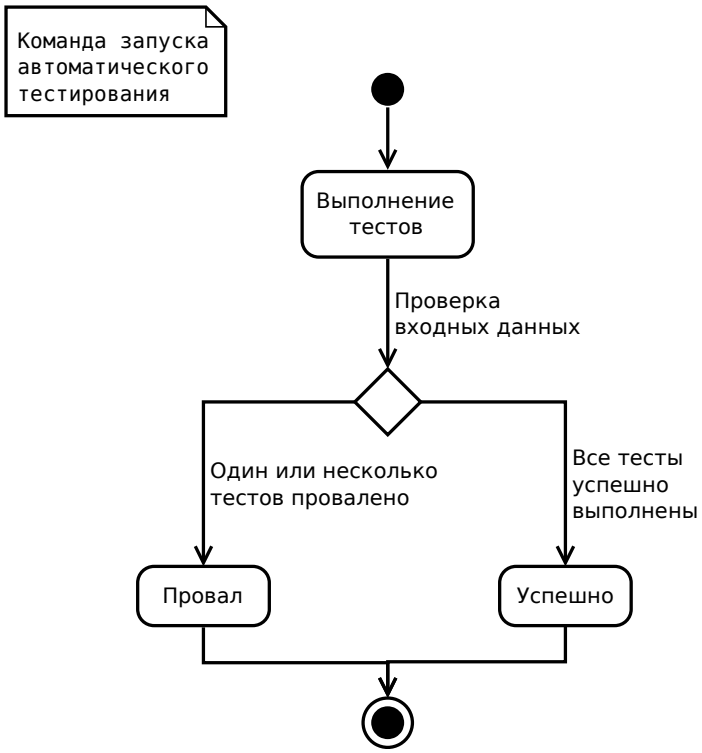

Рисунок 19. Диаграмма состояний выполнения автоматических тестов

2.4 Схемы данных

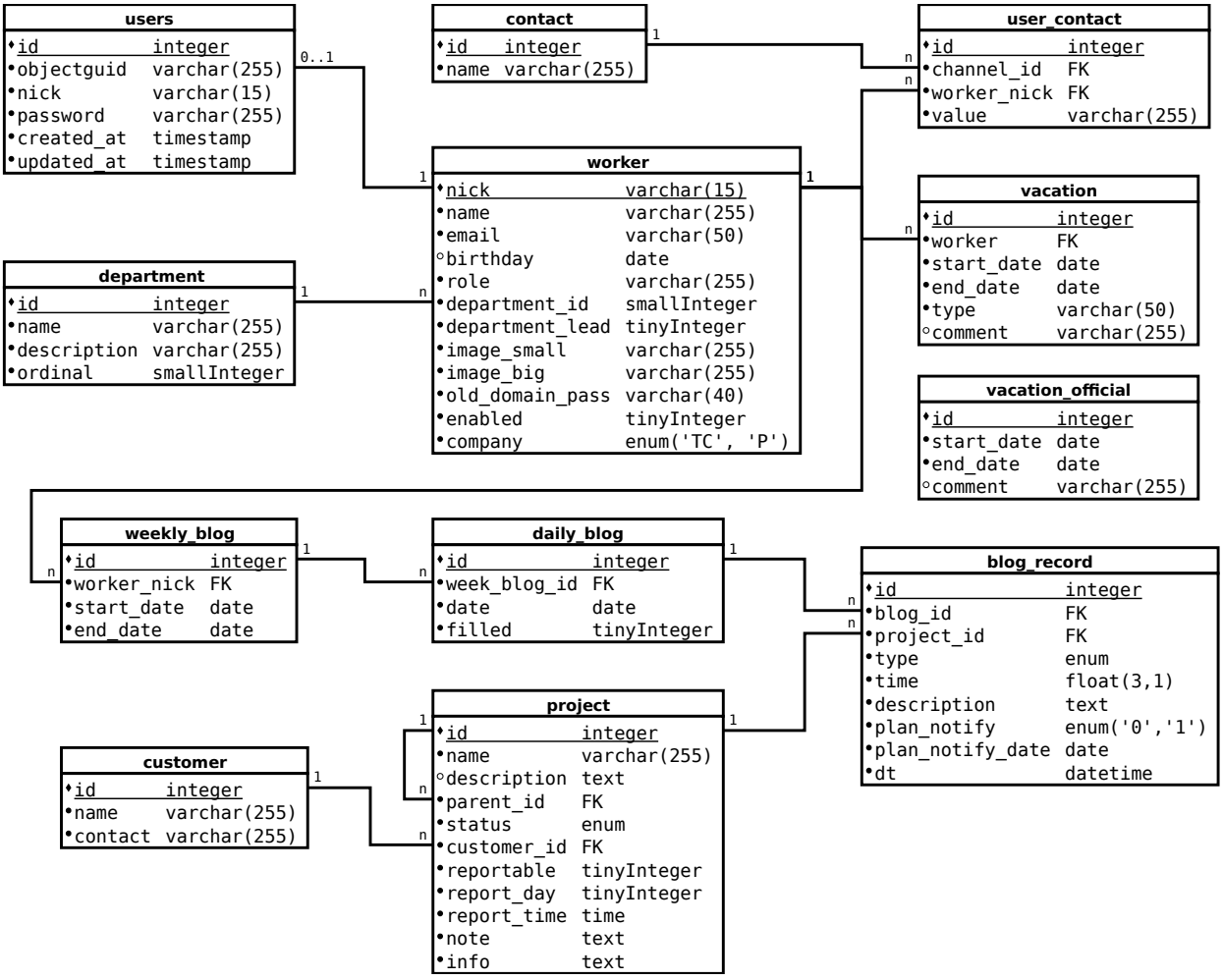

Рисунок 20. Структура данных

Таблица 1 — Users

Название данных	Тип поля	Описание
id	integer	Идентификатор пользователя
objectguid	varchar(255)	Идентификатор LDAP записи
nick	varchar(15)	Ник сотрудника
password	varchar(255)	Пароль пользователя
created_at	timestamp	Дата и время создания пользователя
updated_at	timestamp	Дата и время обновления пользователя

Таблица 2 — Department

Название данных	Тип поля	Описание
id	integer	Идентификатор группы
name	varchar(255)	Название группы
description	varchar(255)	Описание группы
ordinal	smallInteger	Порядок отображения в web интерфейсе

Таблица 3 — Workers

Название данных	Тип поля	Описание
nick	varchar(15)	Ник сотрудника
name	varchar(255)	ФИО сотрудника
email	varchar(50)	Электронная почта
birthday	date	День рождения
role	varchar(255)	Роль
department_id	smallInteger	Внешний ключ на рабочую группу
department_lead	tinyInteger	Внешний ключ на лидера рабочей группы
image_small	varchar(255)	Путь файла на малое изображение профиля
image_big	varchar(255)	Путь файла на большое изображение профиля
old_domain_pass	varchar(40)	Хеш пароля сотрудника
enabled	tinyInteger	Флаг активированного сотрудника
company	enum('TC', 'P')	Подразделение компании

Таблица 4 — Contact

Название данных	Тип поля	Описание
id	integer	Идентификатор группы контактов
name	varchar(255)	Название группы контактов

Таблица 5 — User_contact

Название данных	Тип поля	Описание
id	integer	Идентификатор группы контактов
channel_id	integer	Внешний ключ на группу контактов
worker_nick	varchar(15)	Внешний ключ на сотрудника
value	varchar(255)	Строковое значение контакта

Таблица 6 — Vacation_official

Название данных	Тип поля	Описание
id	integer	Идентификатор записи выходных
start_date	date	Начало периода
end_date	date	Конец периода
comment	varchar(255)	Комментарий

Таблица 7 — Vacation

Название данных	Тип поля	Описание
id	integer	Идентификатор записи выходных
worker	varchar(15)	Внешний ключ на сотрудника
start_date	date	Начало периода
end_date	date	Конец периода
type	varchar(50)	Тип
comment	varchar(255)	Комментарий

Таблица 8 — Weekly_blog

Название данных	Тип поля	Описание
id	integer	Идентификатор записи рабочей недели
worker_nick	varchar(15)	Внешний ключ на сотрудника
start_date	date	Начало периода
end_date	date	Конец периода

Таблица 9 — daily_blog

Название данных	Тип поля	Описание
id	integer	Идентификатор дня
week_blog_id	integer	Внешний ключ на спринт
date	date	Дата дня
filled	tinyInteger	Флаг своевременного заполнения дня

Таблица 10 — Customer

Название данных	Тип поля	Описание
id	integer	Идентификатор заказчика
name	varchar(255)	Имя / название компании заказчика
contact	varchar(255)	Контакт для связи с заказчиком

Таблица 11 — Project

Название данных	Тип поля	Описание
id	integer	Идентификатор проекта
name	varchar(255)	Название проекта
description	text	Описание проекта
parent_id	integer	Родительский проект (если есть)
status	enum(…)	Статус проекта
customer_id	integer	Внешний ключ на заказчика проекта
reportable	tinyInteger	Флаг для запроса отчёта
report_day	tinyInteger	День недели для отправки отчёта
report_time	time	Время, в которое нужно отправить отчёт
note	text	Текст для уведомления о собрании
info	text	Дополнительная информация по проекту

Таблица 12 — blog_record

Название данных	Тип поля	Описание
id	integer	Идентификатор блога
blog_id	integer	Внешний ключ на день
project_id	integer	Внешний ключ на проект
type	enum(…)	Тип выполняемых работ
time	float(3,1)	Затраченное время в часах
description	text	Описание выполненной работы
plan_notify	enum('0', '1')	Флаг отправки уведомления
plan_notify_date	date	День для отправки уведомления
dt	datetime	Временная метка создания блога

2.5 Прототип спецификации API

2.5.1 Запросы с аутентификацией

Запросы, требующие аутентификации, должны быть отправлены с http заголовком запроса "X-Auth-Key", содержащим токен доступа.

Все запросы, требующие токен доступа, помечены как требующий аутентификации.

Токен доступа можно получить по маршрутам POST /api/login (по нику и паролю пользователя) и POST /api/token (используя токен пользователя).

В случае если запрос пришёл без токена, токен устарел или удалён, то будет выдан следующий ответ:

Код 401 — неаутентифицирован

Тело ответа:

```
{
    "message": "Unauthenticated."
}
```

2.5.2 Токены

Создание токена

Требует аутентификации.

Создание токена для внешнего сервиса.

Запрос: POST api/token.

Параметры тела:

1. name строка — название токена

Пример ответа:

Код 200 — хорошо.

```
{
   "data": {
      "name": "report-service",
      "token": "{YOUR_AUTH_KEY}"
   }
}
```

Чтение токенов пользователя

Требует аутентификации.

Чтение всех токенов, что принадлежат аутентифицированному пользователю.

Запрос: GET api/token.

Пример ответа:

Код 200 — хорошо.

```
{
   "data": {
      "tokens": [
         {
```

```
            "name": "login",
            "abilities": [
                "*"
            ],
            "last_used_at": "2022-04-01T11:17:50.000000Z"
        },
        {
            "name": "report-service",
            "abilities": [
                "*"
            ],
            "last_used_at": null
        }
    ]
  }
}
```


Чтение токена пользователя

Требует аутентификации.

Чтение указанного токена у аутентифицированного пользователя.

Запрос: GET api/token/{token_name}

Параметры URL:

token_name строка — название токена пользователя

Пример ответа:

Код 200 — хорошо.


```
{
  "data": {
    "name": "report-service",
    "abilities": [
```

```
            "*"
        ],
        "last_used_at": null
    }
}
```


Редактирование токена

Требует аутентификации.

Редактирование указанного токена у аутентифицированного пользователя.

Запрос: PUT api/token/{token_name}.

Параметры URL:

token_name строка — название токена пользователя

Параметры тела запроса:

1. name строка — название токена
2. abilities массив строк, опционально — список прав доступа, предоставляемых токену

Пример ответа:

Код 200 — хорошо.


```
{
    "data": {
        "name": "report-service",
        "abilities": [
            "report:getShort",
            "report:getByProject"
        ],
        "last_used_at": null
    }
}
```

Удаление токена

Требует аутентификации.

Запрос: DELETE api/token/{token_name}.

Параметры URL:

token_name строка — название токена пользователя

Пример ответа:

Код 204 — хорошо.

Без тела ответа.

2.5.3 Пользователи

Аутентификация

Аутентификация пользователей. По нику и паролю пользователя выдаётся токен с именем "login", который используется для дальнейшего взаимодействия с api.

Токен "login" является полноценным токеном api и может быть также отредактирован. При создании токена, ему предоставляются все права пользователя.

Если токен "login" был ранее создан, то он будет удалён и создан заново.

Запрос: POST api/login.

Пример ответа:

Код 200 — хорошо.

```
{
   "data": {
      "name": "login",
      "token": "{YOUR_AUTH_KEY}"
```

```
    }
}
```


2.5.4 Отчёт по часам

Короткий отчёт

Требует аутентификации.

Получение короткого отчёта о часах сотрудников за определённый период дней.

Простая сумма времени, указанного в блогах, сгруппированные по сотрудникам.

Запрос: GET api/report/hours/short.

Get параметры запроса:

1. start_date строка, опционально — Дата начала периода отчёта. Включительно. Должен быть корректной датой.

2. end_date строка, опционально — Дата конца периода отчёта. Включительно. Должен быть корректной датой.

Пример ответа:

Код 200 — хорошо.


```
{
   "data": {
      "count_reports": 2,
      "reports": [
         {
            "nick": "nherman",
            "name": "Myrna Toy II",
            "company": "TC",
            "total_hours": 4.7
```

```
            },
            {
                "nick": "sterling03",
                "name": "Oliver Daniel",
                "company": "TC",
                "total_hours": 8.200000000000001
            }
        ]
    }
}
```


Отчёт по проектам

Требует аутентификации.

Получение отчёта о часах сотрудников по проектам за определённый период дней.

Если на одном проекте работало несколько сотрудников, то создаётся несколько записей в массиве "reports" с одинаковым значением полей, относящихся к проекту.

Запрос: GET api/report/hours/project.

Get параметры запроса:

1. start_date строка, опционально — Дата начала периода отчёта. Включительно. Должен быть корректной датой.

2. end_date строка, опционально — Дата конца периода отчёта. Включительно. Должен быть корректной датой.

Пример ответа:

Код 200 — хорошо.


```
{
    "data": {
        "count_reports": 4,
```

```
    "reports": [
      {
        "nick": "nherman",
        "name": "Myrna Toy II",
        "company": "TC",
        "customer_id": 24,
        "customer_name": "est explicabo nesciunt",
        "project_id": 26,
        "project_name": "dolores neque quae",
        "total_hours": 4.7
      },
      {
        "nick": "sterling03",
        "name": "Oliver Daniel",
        "company": "TC",
        "customer_id": 28,
        "customer_name": "aut et voluptatem",
        "project_id": 12,
        "project_name": "dicta reiciendis asperiores",
        "total_hours": 4.4
      },
    ]
  }
}
```

2.5.5 Расписания

Общее расписание

Требует аутентификации.

Получение расписания рабочих дней компании.

Запрос: GET api/shedule/official.

Get параметры запроса:

1. start_date строка, опционально — Дата начала периода отчёта. Включительно. Должен быть корректной датой.

2. end_date строка, опционально — Дата конца периода отчёта. Включительно. Должен быть корректной датой.

Пример ответа:

Код 200 — хорошо.


```
{
   "data": {
      "2010-10-07": {
         "status": "holiday",
         "info": [
            "Official holiday"
         ]
      },
      "2010-10-08": {
         "status": "holiday",
         "info": [
            "Official holiday"
         ]
      },
      "2010-10-09": {
```

```
            "status": "work",
            "info": []
        }
    }
}
```


Расписание сотрудников

Требует аутентификации.

Получение расписания рабочих дней сотрудников.

Запрос: GET api/shedule/worker.

Get параметры запроса:

1. start_date строка, опционально — Дата начала периода отчёта. Включительно. Должен быть корректной датой.
2. end_date строка, опционально — Дата конца периода отчёта. Включительно. Должен быть корректной датой.

Пример ответа:

Код 200 — хорошо.


```
{
    "ilene.wolf": {
        "2010-10-07": {
            "status": "holiday",
            "info": [
                "Error possimus maiores eius est sit molestiae qui qui."
            ]
        },
        "2010-10-08": {
            "status": "work",
            "info": [
                "Error possimus maiores eius est sit molestiae qui qui.",
```

```
                "Provident delectus et atque sequi."
            ],
            "total_hours": 5
        },
        "2010-10-09": {
            "status": "work",
            "info": []
        }
    }
}
```


2.6 Тестирование программного продукта

Для обеспечения качественного тестирования, обеспечивающие проверку каждого обновления, были написаны автоматические тесты системы.

Данные тесты рекомендуется применять перед каждым закреплением состояния в системе контроля версий. Данные тесты должны применяться перед каждым объединением тематического ответвления разработки с главной веткой разработки. Это необходимо для качественного сохранения работоспособности приложения.

Таблица 13 — Test case

Название проекта	Разработка модуля «Интеграция» для корпоративной системы ООО «Томскксофт»
Номер версии	1.0
Имя тестера	Кулманаков Илья Владимирович
Даты тестирования	25.05.2022

Таблица 14 — Предоставление прав к ресурсам API

Номер тестового сценария	1
Приоритет теста	Высокий
Название теста	Предоставление прав к ресурсам API
Резюме испытания	Режимы доступа должны соответствовать требованиям.
Шаги тестирования	1. Проверить все маршруты без аутентификации 2. Проверить все маршруты с аутентификацией
Ожидаемый результат	Защита маршрутов соответствует требованиям
Фактический результат	Маршруты защищены в соответствии с требованиями.
Предпосылки	—
Статус	Успешно
Комментарий	1. Протестированы все маршруты без аутентификации и с аутентификацией.

Таблица 15 — Тест отчётов по простой сумме

Номер тестового сценария	2
Приоритет теста	Высокий
Название теста	Тест отчётов по простой сумме
Резюме испытания	Сумма часов, указанных в отчётах должна совпадать с суммой всех записей работ в БД.
Шаги тестирования	2. Создать несколько записей работ с известным количеством затраченного времени и сохранить в БД. 3. Запросить отчёт 4. Посчитать сумму часов в отчёте и сравнить с суммой заложенных значений.
Ожидаемый результат	Сумма часов совпадает
Фактический результат	Отчёты вернули корректный результат
Предпосылки	—
Статус	Успешно
Комментарий	1. Протестирован короткий отчёт рабочего времени. 2. Протестирован отчёт рабочего времени по сотрудникам.

Таблица 16 — Корректная выборка отчётов

Номер тестового сценария	3
Приоритет теста	Высокий
Название теста	Корректная выборка отчётов
Резюме испытания	Отчёты должны выбираться только по запрошенному периоду.
Шаги тестирования	1. Создать несколько записей в заданном периоде, посчитать сумму затраченных на работу часов. 2. Создать несколько записей вне заданного периода. 3. Посчитать сумму часов отчёта, запрошенного по заданному периоду. 4. Сравнить с заранее посчитанной суммой часов.
Ожидаемый результат	Сумма часов совпадает
Фактический результат	Отчёты вернули корректный результат
Предпосылки	В GET параметры запроса переданы начальная и конечная даты запрашиваемого периода.
Статус	Успешно
Комментарий	1. Протестирован короткий отчёт рабочего времени. 2. Протестирован отчёт рабочего времени по сотрудникам.

Таблица 17 — Корректная структура ответа API отчётов

Номер тестового сценария	4
Приоритет теста	Высокий
Название теста	Корректная структура ответа API отчётов
Резюме испытания	Отчёты должны соответствовать требованиям к структуре API
Шаги тестирования	1. Создать несколько записей. 2. Проверить структуру ответа.
Ожидаемый результат	Структура ответа должна быть корректной
Фактический результат	Отчёты вернули корректный результат
Предпосылки	—
Статус	Успешно
Комментарий	1. Протестирован короткий отчёт рабочего времени. 2. Протестирован отчёт рабочего времени по сотрудникам.

Таблица 18 — Получение общего расписания

Номер тестового сценария	5
Приоритет теста	Высокий
Название теста	Получение общего расписания
Резюме испытания	Расписание должно корректно отображать информацию о днях без ошибок.
Шаги тестирования	1. Создать информацию о выходных и сохранить в БД. 2. Получить расписание. 3. Проверить статусы дней, указанные в расписании.
Ожидаемый результат	Все дни являются рабочими, за исключением указанных дней в БД. При наложении диапазонов дней выходных, статус дней является выходными.
Фактический результат	Полученное расписание соответствует заложенным данным.
Предпосылки	—
Статус	Успешно
Комментарий	1. Расписание выбирается согласно запрошенному диапазону дней. 2. Расписание корректно отрабатывает указанные в БД диапазоны выходных дней. 3. Расписание отрабатывает наложение диапазонов выходных дней как выходные, а не конфликт.

Таблица 19 — Получение расписания работников

Номер тестового сценария	6
Приоритет теста	Высокий
Название теста	Получение расписания работников
Резюме испытания	Расписание должно корректно отображать информацию о днях без ошибок.
Шаги тестирования	1. Создать информацию о выходных днях компании и сохранить в БД. 2. Создать работника и назначить ему выходные. 3. Получить расписание. 4. Проверить расписание на соответствие заложенным данным.
Ожидаемый результат	Расписание соответствует заложенным данным.
Фактический результат	Полученное расписание соответствует заложенным данным.
Предпосылки	—
Статус	Успешно
Комментарий	1. Выходные сотруднику работают. 2. Возврат к работе назначается корректно. 3. Согласно с требованиями, наложение выходных и рабочих диапазонов дней возвращает статус дня как конфликтный.

Таблица 20 — Корректная структура ответа API расписаний

Номер тестового сценария	7
Приоритет теста	Высокий
Название теста	Корректная структура ответа API расписаний
Резюме испытания	Отчёты должны соответствовать требованиям к структуре API расписаний
Шаги тестирования	1. Наполнить БД тестовыми данными. 2. Запросить данные и проверить структуру ответа.
Ожидаемый результат	Структура ответа должна быть корректной
Фактический результат	Расписание вернуло корректный результат
Предпосылки	—
Статус	Успешно
Комментарий	1. Протестировано получение общего расписания. 2. Протестировано получение расписания работников.

3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

3.1 Расчёт затрат на разработку программы и решение задачи на ЭВМ

Основными компонентами затрат на разработку программы и решение задачи на ЭВМ являются затраты, связанные с оплатой труда специалистов на разработку программы, обслуживание и эксплуатацию ЭВМ в период отладки программы и решения задачи, то есть рассчитываются прямые и косвенные затраты.

При определении полной себестоимости программы учтены расходы по заработной плате, отчисления в социальные статьи и составлена калькуляция затрат в следующей последовательности:

- основная заработная плата персонала;
- дополнительная заработная плата персонала;
- отчисления во внебюджетные фонды;
- стоимость работ на ЭВМ;
- расчёт косвенных затрат на разработку программы.

При расчёте всех экономических показателей была составлена таблица 21, в которой указаны все этапы работы по разработке программы и решению задачи, исполнитель каждого этапа, трудоёмкость и стоимость исполнения.

Стоимость каждого этапа определена, исходя из оклада исполнителей и времени выполнения этапа.

Количество рабочих часов в месяце равно 168 часов, то есть 21 рабочий день в месяце по 8 часов.

Стоимость часа работы определяется по следующей формуле 1:

$$Ст.ч.=\frac{Оклад.}{К.р.ч} \quad (1)$$

где Оклад. – оклад, руб.;

Ст.ч. – стоимость часа работы;

К.р.ч. – количество рабочих часов в месяце, час.

Стоимость часа работы руководителя

Ст.ч.р = 58800/ 168 = 350 руб./час.

Стоимость часа работы программиста

Ст.ч.п = 15000 / 168 = 89,29 руб./час.

Таблица 21 — Этапы разработки

Наименование этапов работ	Исполнитель	Трудоёмкость, час	Плата за час, руб./час	Стоимость исполнения, руб.
Постановка задачи	Руководитель	9	350	3150
	Программист	13	89,29	1160,71
Изучение литературы	Программист	19	89,29	1696,43
Технический проект	Программист	23,5	89,29	2098,21
	Руководитель	3	350	1050
Эскизный проект	Программист	10	89,29	892,86
	Руководитель	3	350	1050
Написание кода	Программист	85	89,29	7589,29
Отладка программы Тестирование	Программист	29	89,29	7589,29
	Руководитель	8	350	2800
Оптимизация программы	Программист	50	89,29	4464,29
Оформление сопроводительной документации	Программист	20	89,29	1785,71
	Руководитель	1	350	350
Итого	Программист	249	89,29	22276,79
	Руководитель	24	350	8400

Основная заработная плата персонала рассчитывается по формуле 2:

$$ЗП = Сти * РК \quad (2)$$

где ЗП – основная заработная плата персонала;

Сти – стоимость исполнения из таблицы 21 для каждого исполнителя, руб.;

РК – районный коэффициент (1,3).

Основная заработная плата руководителя

$ЗП_{рук.}$ = 8400 × 1,3 = 10920 руб.

Основная заработная плата программиста

$ЗП_{пр.}$ = 22276,79 × 1,3 = 28959,82 руб.

Дополнительная заработная плата персонала рассчитывается по формуле 3:

$$ЗП_{доп} = ЗП * 0{,}10 \quad (3)$$

где $ЗП_{доп}$ – дополнительная заработная плата персонала;

ЗП – основная заработная плата персонала, руб.

Дополнительная заработная плата руководителя

ЗПдоп = 10920 × 0,1 = 1092руб.

Дополнительная заработная плата программиста

ЗПдоп = 28959,82 × 0,1 = 2895,98 руб.

Отчисления во внебюджетные фонды рассчитываются по формуле 4:

$$О_{сн} = (ЗП + ЗП_{доп}) \times 0{,}302 \quad (4)$$

где $ЗП_{доп}$ - дополнительная заработная плата;

ЗП – основная заработная плата;

$О_{сн}$ - отчисления во внебюджетные фонды (30,2%).

Отчисления во внебюджетные фонды составляют 30,2%, из них:

- в пенсионный фонд($О_{пф}$) отчисляется 22%;
- в фонд социального страхования($О_{сстр}$) – 2,9%;
- в фонд медицинского страхования($О_{мс}$) – 5,1%;
- страхование от несчастных случаев на производстве – 0,2%

Отчисления во внебюджетные фонды от заработной платы руководителя рассчитывается по формуле 4,

где $О_{пф}$ – отчисления в пенсионный фонд;

$О_{сстр}$ – отчисления в фонд социального страхования;

$О_{мс}$ – отчисления в фонд медицинского страхования;

$О_{сс}$ – отчисления во внебюджетные фонды..

$О_{пф}$= (10920+ 1092) × 0,22 = 2642,64 руб.;

$О_{сстр}$= (10920+ 1092) × 0,029 = 348,35 руб.;

$О_{мс}$= (10920+ 1092) × 0,051 = 612,61 руб.;

$О_{сс}$= (10920+ 1092) × 0,302 = 3627,62 руб.

Отчисления во внебюджетные фонды от заработной платы программиста рассчитывается по формуле 4:

$О_{пф}$= (28959,82+ 2895,98) × 0,22 = 7008,28 руб.;

$О_{сстр}$= (28959,82+ 2895,98) × 0,029 = 923,82 руб.;

$О_{мс}$= (28959,82+ 2895,98) × 0,051 = 1624,65 руб.;

$О_{сс}$= (28959,82+ 2895,98) × 0,302= 9620,45 руб.

Для расчёта стоимости работ на ЭВМ учтём амортизацию ЭВМ на период написания программы и расходы на электроэнергию, используемую при разработке программы.

Стоимость работ на ЭВМ рассчитывается по формуле 5:

$$C_{рм} = C_{мч} \times T_{м} \quad (5)$$

где $С_{мч}$ – стоимость машинного часа в рублях;

$С_{рм}$ – стоимость работ на ЭВМ;

$Т_{м}$ – общее время работы ЭВМ (час).

Стоимость работ на ЭВМ

$С_{рм}$ = 0,3 × 273,5 = 82,05 руб.

Расчёт косвенных расходов на разработку программы рассчитывается по формуле 6

$$Р_{к} = ЗП * К_{нр} \quad (6)$$

где $Р_{к}$– косвенные расходы на разработку программы;

ЗП – основная заработная плата персонала;

$К_{нр}$ - коэффициент накладных расходов (5-10%).

$Р_{к}$ = 39879,82 × 0,05 = 1993,99 руб.

Полная себестоимость программы приведена в таблице 22.

Таблица 22 — Смета затрат на разработку

Наименование статей расходов	Стоимость работ, руб.
Основная заработная плата	39879,98
Дополнительная заработная плата	3987,98
Отчисления во внебюджетные фонды	13248,08
Стоимость работ на ЭВМ	82,05
Косвенные расходы	1993,99
Итого	59191,92

3.1.1 Расчёт годовых затрат на эксплуатацию программы

Стоимость одного непосредственного решения на ЭВМ определяется по формуле 7:

$$C_{р.м} = C_{мч} \times T_р + ЗП_{о.п.} \times Q \times K_р \times K_{кр} \quad (7)$$

где $C_{р.м}$ – стоимость одного непосредственного решения на ЭВМ;

$C_{мч}$ – стоимость работы на ЭВМ за час (руб./час);

$T_р$ – время решения задачи на ЭВМ (час);

Q – трудоёмкость исполнителя (час);

$K_р$ – районный коэффициент (1,3);

$K_{кр}$ – коэффициент косвенных расходов (1,05);

$ЗП_{о.п}$ – заработная плата за час работника (руб./час).

Стоимость одного непосредственного решения на ЭВМ

$C_{р.м} = 2,31 \times 0,0025 + 0,01 \times 178,57 \times 1,3 \times 1,05 = 2,44$ руб.

Расчёт годовых затрат на эксплуатацию программы необходимо провести для последующего анализа эффективности данного программного продукта.

Готовые затраты на эксплуатацию программы рассчитываются по формуле 8:

$$C_{р.м.год} = N \times C_{р.м} + E_н \times C \quad (8)$$

где N – плотность потока заявок (заявок в год);

$С_{р.м.год}$ – годовые затраты на эксплуатацию программы;

$С_{р.м}$ – стоимость одного непосредственного решения на ЭВМ;

$Е_{н}$ – нормальный коэффициент сложности (0,2-0,6);

С – себестоимость разработки программы (итог таблицы 2).

Расчёт годовых затрат на эксплуатацию программы

$С_{р.м.год} = 150 \times 2,44 + 0,2 \times 58697,93 = 12106,08$ руб.

3.2 Расчёт экономического эффекта и определение срока окупаемости

Экономический эффект достигается при эксплуатации и характеризуется экономией времени работы специалиста, повышением производительности труда.

Для того чтобы определить экономическую эффективность проекта необходимо рассчитать затраты на эксплуатацию ранее употреблявшимся образом.

3.2.1 Расчёт годовых затрат на выполнение работ ранее употреблявшимся способом

Расходы на выполнение работ ранее употреблявшимся способом рассчитываются по формуле 9:

$$C_{р.сп} = ЗП_{сп} \times Т_{сп} \times К_{кр} \times К_{р} \quad (9)$$

где $ЗП_{сп}$ – заработная плата специалиста за час (руб./час);

$Т_{сп}$ – затраты времени специалиста на выполнение работ ранее употреблявшимся способом (ч);

$К_{р}$ – районный коэффициент (1,3);

$K_{кр}$ – коэффициент косвенных расходов (1,05).

Расходы на выполнение работ ранее употреблявшимся способом:

$C_{р.сп}$= 125 × 5 × 1,05 × 1,3 = 853,125 руб.

Зная стоимость всех работ по выполнению одной задачи, определим годовые расходы ранее употреблявшимся способом:

$$C_{р.сп.год}=N \times C_{р.сп} \quad (10)$$

где N – плотность потока заявок (заявок в год);

$C_{р.сп.год}$ – годовые расходы ранее употреблявшимся способом.

Расчёт годовых затрат на выполнение работ ранее употреблявшимся способом

$C_{р.сп.год}$ = 150 × 853,13 = 127968,75 руб.

Экономический эффект и срок окупаемости

Экономия рассчитывается по формуле:

$$Э_{год}=C_{р.сп.год}-C_{р.м.год} \quad (11)$$

где $Э_{год}$ – экономия в год;

$C_{р.сп.год}$ – годовые затраты на выполнение работ ранее употреблявшимся способом;

$C_{р.м.год}$ – годовые затраты на эксплуатацию программы.

Определение коэффициента экономической эффективности программы.

Коэффициента экономической эффективности показывает сколько на 1 руб. вложенных затрат в разработку и эксплуатацию, получаем экономии. Чем больше данное значение, тем эффективнее проект.

Данный коэффициент рассчитывается по формуле 12:

$$E_p=\frac{Э_{год}}{C+C_{р.м.год}} \quad (12)$$

Рассчитаем экономию, связанную с использованием разработки:

Э = $C_{р.сп.год}$ - $C_{р.м.год}$ = 127968,75 - 12204,88 = 115763,87

Рассчитаем экономическую эффективность программы:

$$E_p = \frac{Э_{год}}{C + C_{р.м.год}} = \frac{115763,87}{127968,75 + 59191,92} = 0,62$$

Экономический эффект показывает, что на 1 вложенный рубль в разработку и эксплуатацию программы, получаем 0,62 рублей экономии. Так как проект не предполагает коммерциализации, мы не можем посчитать его коммерческую эффективность, но в результате внедрения программы облегчается труд специалиста, снижаются затраты времени на решение задач.

Срок окупаемости программы рассчитываем исходя из экономии. То есть благодаря экономии за какой период времени окупятся затраты на разработку и внедрение программы.

$$T_{ок} = \frac{1}{E_p} \quad (13)$$

Рассчитаем срок окупаемости программы:

$$T_{ок} = \frac{1}{0,62} = 1,62$$

Таким образом, программа окупится через 1,62 года.

Выводы. На основании проведённых расчётов себестоимости и экономического эффекта можно сделать следующие выводы. Результаты технико-экономического обоснования свидетельствуют об экономической эффективности проекта. За счёт снижения эксплуатационных затрат проект окупится через 19,4 месяца.

Заключение

В ходе выполнения дипломного проекта было разработано приложение для интеграции сервисов с данными компании Томсксофт.

Данное приложение является первой версией и закладывает основу для развития.

Реализованный функционал

В ходе выполнения работы был разработан следующий функционал системы:

1. Авторизация;
2. CRUD токенов;
3. Формирование отчётных данных по часам сотрудников и по часам сотрудников по проектам;
4. Получение расписания рабочих дней компании;
5. Получение расписания рабочих дней сотрудников компании;
6. Получение контактов сотрудников;
7. Автоматические тесты разрабатываемой системы.

Перечень использованной литературы

1. Eloquent: Getting Started [Электронный ресурс] / Laravel. — 2022. — Режим доступа: https://laravel.com/docs/9.x/eloquent. Дата обращения: 10.04.2022.

2. Eloquent · Начало работы [Электронный ресурс] / Laravel Russian Community. — 2022. — Режим доступа: https://laravel.su/docs/8.x/eloquent. Дата обращения: 10.04.2022.

3. Руководство по PHP [Электронный ресурс] / php.net. — 2022. — Режим доступа: https://www.php.net/manual/ru/index.php. Дата обращения: 10.04.2022.

4. Пакет Laravel Sanctum [Электронный ресурс] / Laravel Russian Community. — 2022. — Режим доступа: https://laravel.su/docs/8.x/sanctum. Дата обращения: 10.04.2022.

5. Eloquent Power Joins [Электронный ресурс] / Kirschbaum // README.md — 2022. — Режим доступа: https://github.com/kirschbaum-development/eloquent-power-joins. Дата обращения: 16.05.2022.

6. Endpoint metadata [Электронный ресурс] / Scribe. — 2022. — Режим доступа: https://scribe.knuckles.wtf/laravel/documenting/metadata. Дата обращения: 23.05.2022.

7. Выбираем Yii2 или laravel [Электронный ресурс] / Хабр // dastanaron_dev. — 2022. — Режим доступа: https://habr.com/ru/post/353434/. Дата обращения: 06.06.2022.