Оглавление

ВВЕДЕНИЕ

Согласно исследованиям [1], на момент 2019 года 28% сотрудников постоянно или достаточно часто чувствовали себя угнетёнными под давлением рабочих обязанностей, а 48% страдали синдромом эмоционального выгорания время от времени.

На сегодняшний день проблема эмоционального выгорания касается не только самих работников, но и компаний, которые при утрате контроля над ситуацией вынуждены увольнять сотрудников под предлогом неисполнения ими обязанностей. [2]

Причиной синдрома может быть и физическое, и эмоциональное истощение вследствие увеличения нагрузки на работе, а также количества возлагаемых обязанностей на сотрудника. [3]

Синдром хронической усталости также является одним из факторов, понижающим работоспособность сотрудников. Несмотря на то, что истинная этиология данного заболевания до конца не раскрыта, одним из возможных факторов появления данного синдрома приписывают высокой нагрузке как умственной, так и физической. [4]

Усталость негативно влияет на производительность труда, а также на психологическое и физическое состояние человека. В условиях современной цифровизации медицины становится реальным учитывать индивидуальные особенности организма, управление его работоспособностью и проведение профилактики проявления вышеописанных синдромов, приводящих к неутешительным последствиям.

Цель работы – спроектировать и реализовать серверное приложение для доступа к базе данных, предназначенной для хранения информации о действиях и характеристиках, необходимых для определения усталости пользователей автоматизированного рабочего места (АРМ).

Для достижения поставленной цели потребуется:

1) формализовать задачу;
2) определить требуемую функциональность;
3) проанализировать существующие решения;
4) описать протокол взаимодействия клиента и сервера;
5) разработать приложение для решения поставленной задачи.

1. Аналитический раздел

В данном разделе формализуется задача, приводится требуемая функциональность разрабатываемого приложения, проводится анализ существующих решений.

1.1 Формализация задачи

Необходимо реализовать клиент-серверное приложение для доступа к базе данных, предназначенной для хранения информации о действиях и характеристиках, необходимых для определения усталости пользователей АРМ. Данная потребность связана с тем, что при работе с InfluxDB на ЯП Kotlin на текущий момент отсутствуют библиотеки, отвечающие полноте функционала, который может понадобится при проводимых работах.

В качестве решения поставленной задачи поставщики СУБД предлагают разработчику реализовать собственное серверное приложение, которое будет обрабатывать запросы, используя обращения к API развёрнутой СУБД.

К возможностям, которые должен предоставлять сервер, отнесены:

- внесение данных;
- получение данных;
- проверка существования хранилища для пользователя;
- создание хранилищ для новых пользователей.

1.2 Существующие решения

1.2.1 The Kotlin InfluxDB 2.0 Client

The Kotlin InfluxDB 2.0 Client [5] - это клиент, который предоставляет возможность производить запросы и запись в InfluxDB 2.0 с использованием ЯП Kotlin. Данная библиотека поддерживает асинхронные запросы с использованием Kotlin Coroutines.

На данный момент решение поддерживает следующий функционал:

- запись в базу данных;
- чтение базы данных с использованием стандартного языка InfluxQL;
- чтение базы данных с использованием языка Flux.

Данным решением не поддерживается следующий требуемый функционал:

- проверка существования хранилища для пользователя;
- создание хранилищ для новых пользователей.

1.2.2 InfluxDB v2 API

InfluxDB поддерживает обращение к InfluxDB v2 API [6]. InfluxDB API предоставляет способ взаимодействия с базой данных с использованием HTTP-запросов и ответов, включающих в своё тело данные в формате JSON, HTTP аутентификации, а также с поддержкой токенов JWT и базовой аутентификации.

Предоставляемый данным интерфейсом функционал полон и непосредственно используется в реализации Web-клиента данной СУБД. К недостатку использования данного метода взаимодействия относятся формирование множественных HTTP-запросов и потребность в обработке ответов.

Вывод

Среди рассмотренных существующих решений отсутствуют примеры удобной реализации, отвечающей полноте функционала, которую можно было бы использовать при написании приложений на ЯП Kotlin.

1.3 Понятие ”клиент-серверного”приложения

Программное обеспечение архитектуры ”клиент-сервер”состоит из двух частей: программного обеспечения сервера и программного обеспечения пользователя – клиента. Программа-клиент выполняется на компьютере пользователя и посылает запросы к программе-серверу, которая работает на компьютере общего доступа. Основная обработка данных производится мощным сервером, а на компьютер пользователя возвращаются только результаты выполнения запроса. В такой архитектуре сервер называется сервером баз данных. [7]

Иными словами, данная архитектура определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций, и клиенты, потребители данных функций.

Каждое приложение, опирающееся на архитектуру ”клиент-сервер определяет собственные или использует имеющиеся правила взаимодействия между клиентов и сервером, которые называются протоколом обмена или протоколом взаимодействия. [8]

1.4 HTTP

Протокол HTTP реализует клиент-серверную технологию, которая предполагает наличие множества клиентов, инициирующих соединение и посы-

лающих запрос, а также множества серверов, получающих запросы, выполняющих требуемые действия и возвращающих клиентам результат. [9]

В данном протоколе главным объектом обработки является ресурс, который в клиентском запросе записан в URI (Uniform Resource Identifier). В качестве ресурса выступают файлы, хранящиеся на сервере. HTTP позволяет определить в запросе и ответе способ представления ресурса по различным параметрам. [9]

К преимуществам протокола HTTP относят [9]:

- простоту;
- расширяемость;
- распространённость;
- документация на различных языках.

К недостаткам данного протокола относят [9]:

- отсутствие ”навигации”;
- отсутствие поддержки распределенных действий.

Каждое HTTP-сообщение состоит из трех частей [10]:

- стартовой строки (определяющей тип сообщения);
- заголовков (характеризующих тело сообщения, параметры передачи и т.д.);
- тела (данные сообщения).

Версия протокола 1.1 включает в себя требование наличия заголовка Host. Сами заголовки представляют собой строки, содержащие разделенную двоеточием пару парамтера и значения. [10]

Стартовая строка включает в себя [10]:

- метод (тип запроса, одно слово заглавными буквами);
- путь к запрашиваемому ресурсу;
- версию используемого протокола.

Метод указывает на основную операцию над ресурсом. Среди наиболее часто используемых методов можно выделить [10]:

- GET – запрос содержимого указанного ресурса;
- POST – передача пользовательских данных заданному ресурсу;
- PUT – загрузка содержимого запроса по указанному пути;
- PATCH – PUT, применимый к фрагменту ресурса;
- DELETE – удаление указанного ресурса.

В качестве ответа клиенту сервер также направляет код состояния, по которому тот узнает о результатах выполнения запроса. Выделяют 5 классов состояний [10]:

- 1xx – информационный (информирование о процессе передачи);
- 2xx – успех (информирование о случаях успешного принятия и обработки запроса);
- 3xx – перенаправление (сообщение о том, что для успешного выполнения операции потребуется выполнить другой запрос);
- 4xx – ошибка клиента (указание на ошибки со стороны клиента);
- 5xx – ошибка сервера (информирование о неудачном выполнении операции на стороне сервера).

Вывод

В разделе была предоставлена формализация задачи: реализация клиент-серверного приложения для доступа к базе данных InfluxDB с предоставлением возможности внесения и получения данных, проверки существования хранилища для пользователя и создания хранилищ для новых пользователей. Была предоставлена информация о существующих решениях, которые могут использоваться при решении задачи. Сделаны выводы о том, что среди рассмотренных продуктов отсутствуют примеры удобной в использовании реализации, отвечающей полноте функционала. Также было предоставлено определение понятия ”клиент-серверного”приложения и приведена информация о наиболее часто используемом протоколе взаимодействия, используемого в подобных приложениях.

2. Конструкторский раздел

2.1 Диаграмма вариантов использования

2.2 Блок-схема работы сервера

На рисунке 2.1 предоставлена блок-схема начала работы сервера и алгоритма обработки запросов пользователей.

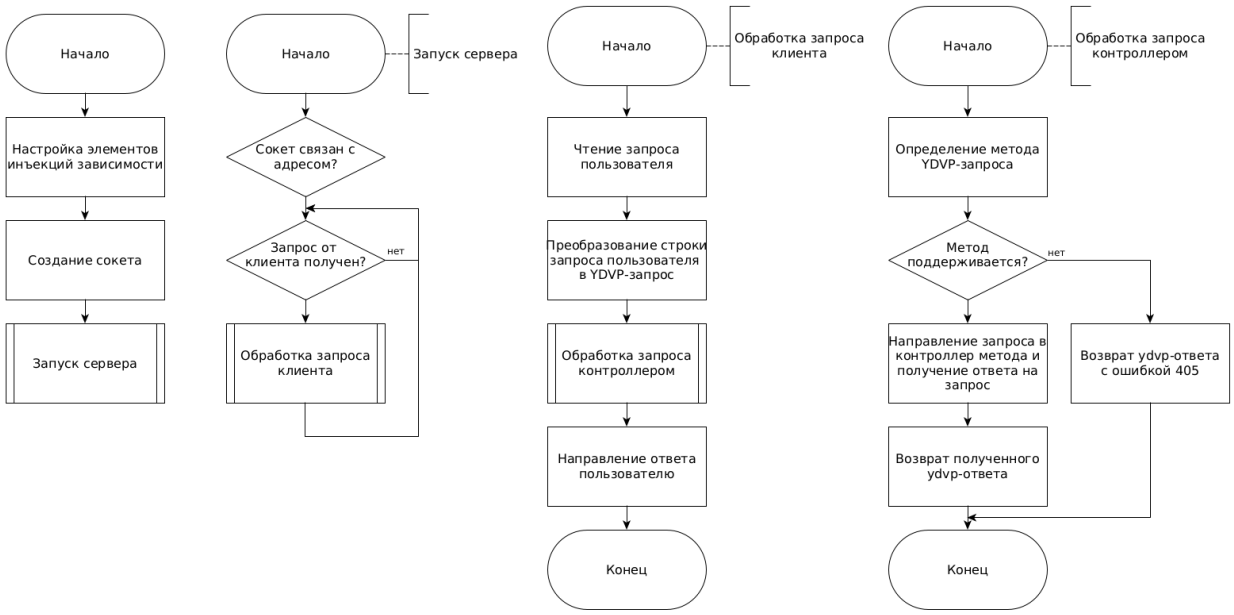

Рис. 2.1: Блок-схема работы сервера приложения.

2.3 Описание YDVP-протокола

На основе существующего стандарта HTTP в данном подразделе описывается структура реализуемого YDVP-протокола.

YDVP-запрос включает в себя три компонента:

- стартовую строку (определяющей тип сообщения);
- заголовки (характеризующих тело сообщения, параметры передачи и т.д.);
- тело (данные сообщения).

И так далее по списку, грубо говоря, только копируем... Обязательно говорим о том, что в нашем протоколе потребуется только поддержка GET и POST методов, а также пара слов о том, что текущая версия 0.1

3. Технологический раздел

3.1 Структура и состав классов

На рисунках 3.1 – 3.4 предоставлены диаграммы классов компонентов приложения.

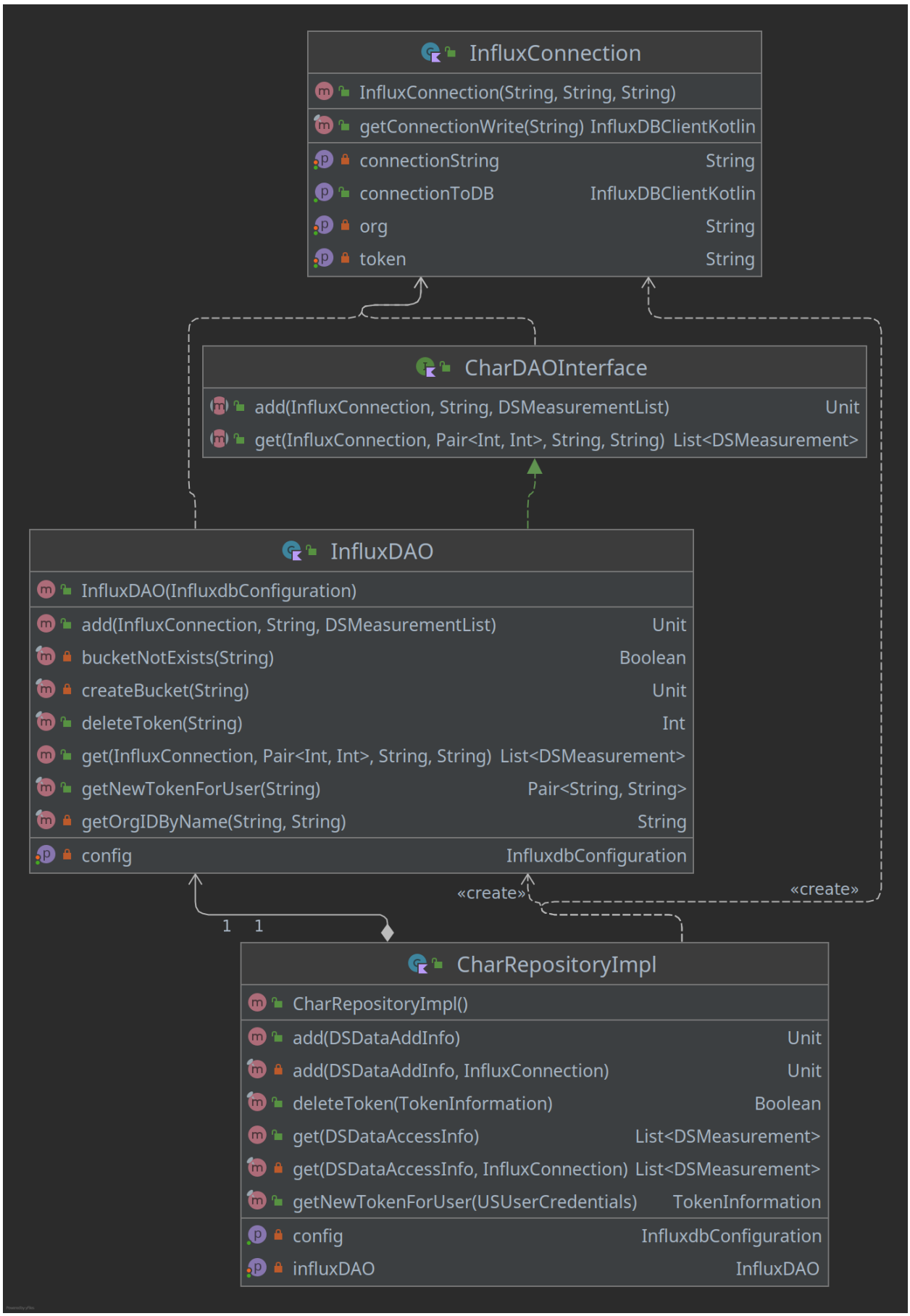

Рис. 3.1: Диаграмма классов слоя доступа к данным приложения.

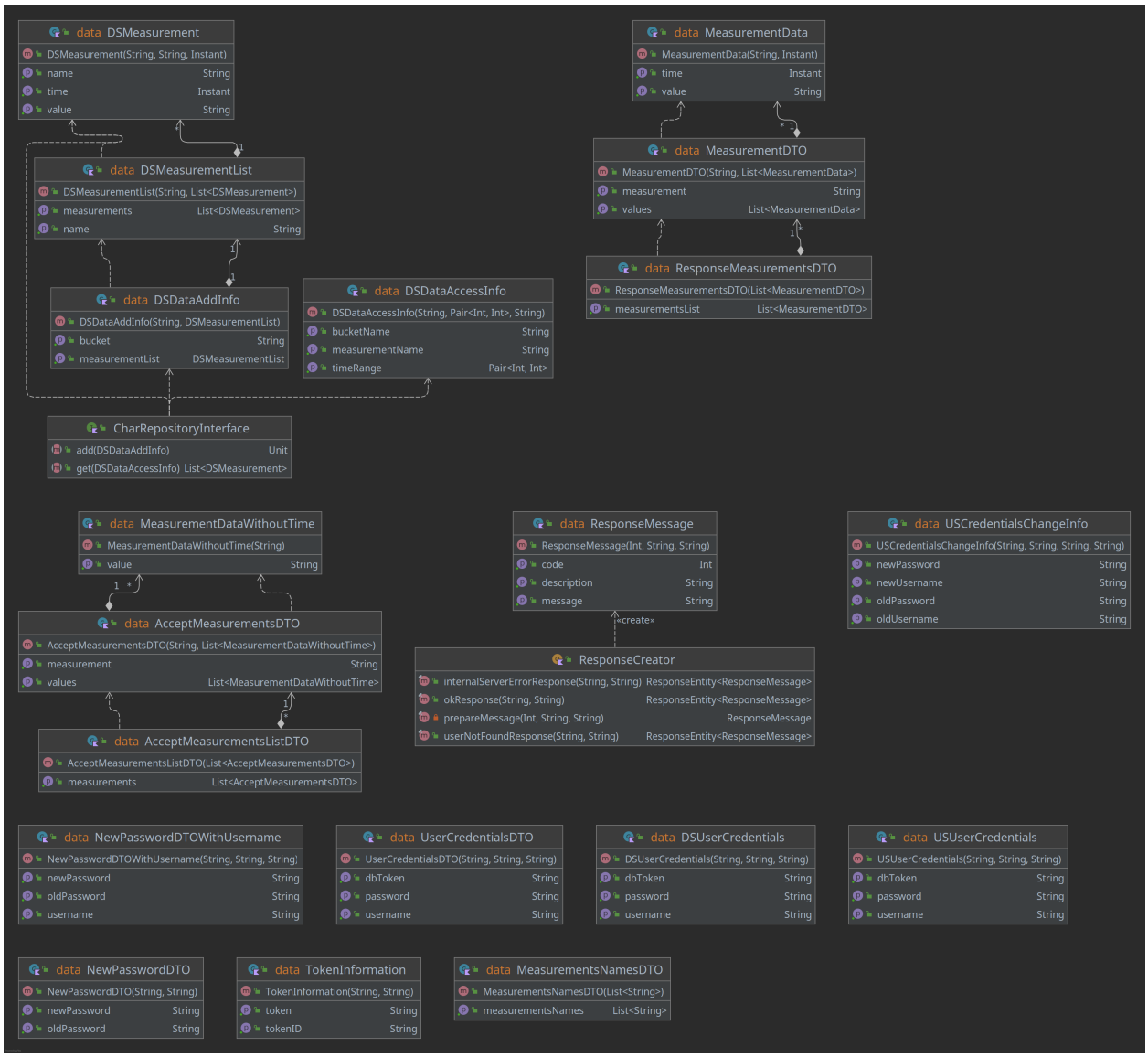

Рис. 3.2: Диаграмма классов бизнес-логики приложения.

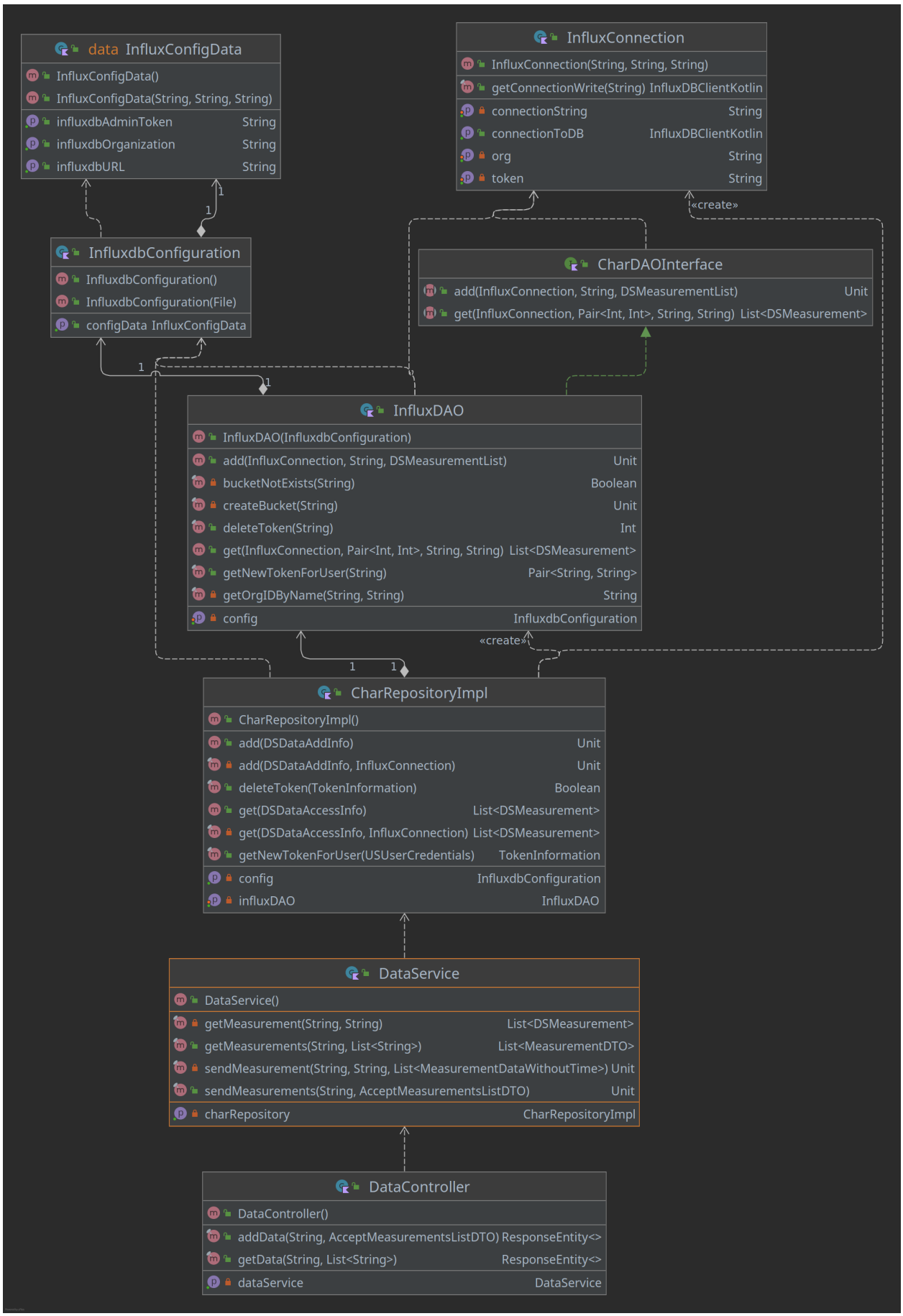

Рис. 3.3: Диаграмма классов, показывающая связь контроллера и слоя доступа к данным.

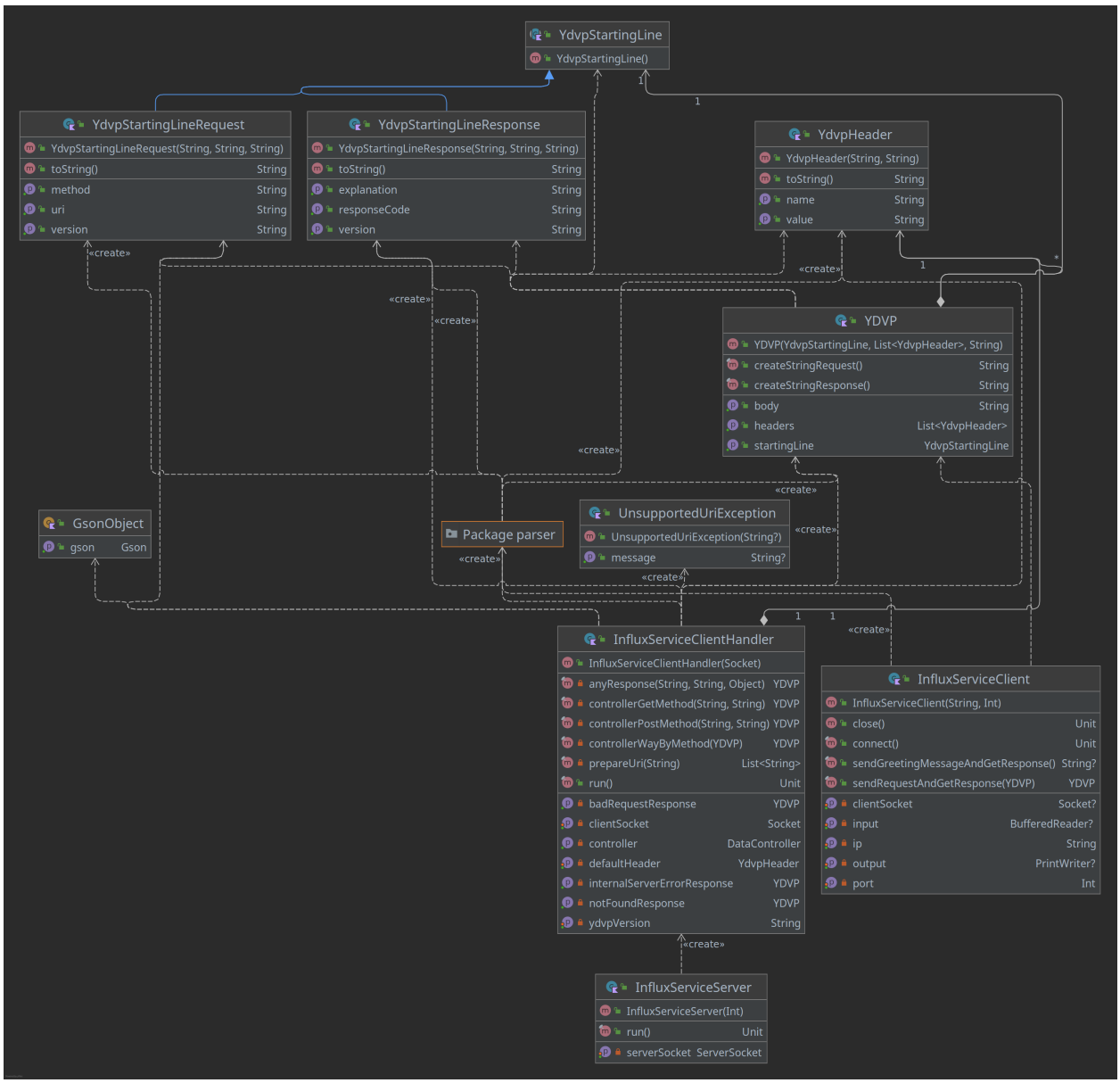

Рис. 3.4: Диаграмма классов серверной части приложения.

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Список литературы

1. Gallup. Employee Burnout: Causes and Cures // Gallup. 2020. p. 32.

2. Moss J. Burnout Is About Your Workplace, Not Your People [Электронный ресурс]. Режим доступа: https://hbr.org/2019/12/burnout-is-about-your-workplace-not-your-people (дата обращения 27.03.2021).

3. Г.А. Макарова. Синдром эмоционального выгорания. Просвящение, 2009. с. 432.

4. Е.А. Пигарова А.В. Плещева. Синдром хронической усталости: современные представления об этиологии // Ожирение и метаболизм. 2010. с. 13.

5. The Kotlin InfluxDB 2.0 Client Github [Электронный ресурс]. Режим доступа: https://github.com/influxdata/influxdb-client-java/tree/master/client-kotlin (дата обращения 13.12.2021).

6. Influx Data: InfluxDB v2 API [Электронный ресурс]. Режим доступа: https://docs.influxdata.com/influxdb/v2.1/reference/api/ (дата обращения 13.12.2021).

7. Гайнанова Р.Ш. Широкова О.А. Создание клиент-серверных приложений // Вестник Казанского технологического университета. 2017. № 9. С. 79–84.

8. Учебно-методические материалы для судентов кафедры АСОИУ: архитектура клиент-сервер [Электронный ресурс]. Режим доступа: https://www.4stud.info/networking/lecture5.html (дата обращения 13.12.2021).

9. Бондаренко Т.В. Федотов Е.А. Бондаренко А.В. Разработка http сервера // ИВД. 2018. Т. 49, № 2.

10. Официальная документация HTTP/1.1 [Электронный ресурс]. Режим доступа: https://datatracker.ietf.org/doc/html/rfc2616 (дата обращения 13.12.2021).