СОДЕРЖАНИЕ

ВВЕДЕНИЕ

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

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, понижающих работоспособность сотрудников. Несмотря на то, что 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, управлять его работоспособностью и проводить профилактику проявления вышеописанных синдромов, приводящих к неутешительным последствиям.

Цель работы — реализовать и апробировать метод распознавания усталости оператора автоматизированного рабочего места по данным, приходящим с устройств взаимодействия пользователя с системой.

Для достижения поставленной цели необходимо:

- реализовать метод систематического распознавания усталости на автоматизированном рабочем месте;
- провести исследование разработанного метода путем сравнения количества успешных определений работоспособности пользователя.

1 Программная реализация метода систематического распознавания усталости на автоматизированном рабочем месте

1.1 Средства реализации программного обеспечения

При написании программного продукта был использован язык программирования Kotlin [5].

Данный выбор обусловлен следующими факторами:

- возможность запуска программного кода на любом устройстве, поддерживающем Java Virtual Machine;
- большое количество актуализируемой справочной литературы, связанной как я языком программирования Java, так и Kotlin;
- возможность интеграции программного кода в приложения для операционной системы Android.

При написании программного продукта использовалась среда разработки IntelliJ IDEA. Данный выбор обусловлен тем, что Kotlin является продуктом компании JetBrains, поставляющей данную среду разработки.

1.2 Выбор СУБД

1.2.1 Базы данных временных рядов

Базы данных временных рядов отличаются от статических баз данных тем, что содержат записи, в которых некоторые из атрибутов ассоциируются с временными метками. В качестве таких записей могут выступать данные мониторинга, биржевые данные о торгах или транзакции продаж. [6]

InfluxDB InfluxDB - это база данных временных рядов, предназначенная для обработки высокой нагрузки записи и запросов.

Основным назначением является хранение больших объемов данных с метками времени. Например, данные мониторинга, метрики приложений и данные датчиков интернета вещей.

В традиционной реляционной базе данных данные хранятся до тех пор, пока не будет принято решение об их удалении. Учитывая сценарии использования баз данных временных рядов, можно не хранить данные слишком долго: это или дорого, или они со временем теряют актуальность. [7]

Системы, подобные InfluxDB, могут удалять данные спустя определенное время, используя концепцию, называемую политикой хранения. Также поддерживается функционал отправки непрерывных запросов к оперативным данным для выполнения определенных операций. [7]

OpenTSDB OpenTSDB включает в себя “демона” временных рядов, а также набор утилит командной строки. Взаимодействие с OpenTSDB в первую очередь достигается путем запуска одного или нескольких независимых “демонов”.

“Демон” использует базу данных с открытым исходным кодом HBase или службу Google Bigtable для хранения и получения данных временных рядов. Схема данных высоко оптимизирована для быстрого объединения аналогичных временных рядов, чтобы минимизировать пространство хранения. Пользователям никогда не требуется прямой доступ к базовому хранилищу. Можно общаться с “демоном” через протокол telnet, HTTP API или простой встроенный графический интерфейс.

1.2.2 Реляционные базы данных

Реляционная база данных — это организованный по реляционной модели набор таблиц, в которых каждая ячейка этих таблиц имеет некоторое соответствующее описание. [8]

Использование реляционной модели предполагает возможность идентификации элементов по совокупности уникальных идентификаторов: имя столбца, первичный ключ. Для построения логической связи между строками и ячейками разных таблиц используются внешние ключи. [8]

Среди подобных СУБД, основанных на реляционных базах данных, пользуются популярностью Oracle и PostgreSQL.

Каждая из указанных СУБД имеет некоторые отличительные особенности. Так, например, PostgreSQL поддерживает вставки кода, написанного на языке программирования Python, в тело процедуры. Однако выделить среди данных особенностей важных для данной работы не представляется возможным.

Вывод

Базы данных временных рядов являются наиболее подходящими для решаемой задачи, так как они нацелены на хранение, извлечение и анализ боль-

шого количества статистических данных, в которых имеются временные метки.

Для организации хранения данных будет использоваться СУБД InfluxDB, так как она является одной из самых популярных среди известных баз данных временных рядов, а также по той причине, что поддержка данной СУБД все еще не прекращена на сегодняшний день.

1.3 Алгоритм и данные для кластеризации

В качестве используемого алгоритма кластеризации был выбран метод с-средних в силу того, что число кластеров заранее известно, а также задача рассматривает установку соответствия некоторого объекта (например, значения скорости печати) набору вещественных значений, показывающих степень отношения объекта к кластерам.

В качестве данных для кластеризации используются действия оператора автоматизированного рабочего места, производимые с использованием клавиатуры и мыши. Данные действия логируются, а затем направляются в базу данных для возможности переноса определенной модели поведения на другое автоматизированное место.

Действия пользователя, участвующие в построении модели, соотносятся с временем реакции, которое фиксируется каждые 10 минут, причем по времени реакции определяется, в каком состоянии в текущий момент времени находится организм оператора.

1.4 Сведения о модулях

Программное обеспечение состоит из модулей логирования действий оператора, обработки и анализа данных. Также определен модуль, который позволяет развернуть сервер для хранения данных пользователей с использованием базы данных InfluxDB.

1.4.1 Модуль логирования действий оператора

Данный модуль предназначен для записи информации о действиях оператора.

Библиотеки, используемые в модуле:

- Java Swing [9] — библиотека легковесных компонентов для реализации оконного интерфейса приложения;
- JNativeHook [10] — библиотека, предоставляющая средства перехвата прерываний, поступающих от клавиатуры и мыши.

Модуль состоит из четырех пакетов.

Пакет window

Данный пакет включает в себя абстрактный класс Window, предоставляющий родительский класс с определенными свойствами для всех окон реализуемого программного обеспечения. Реализация приведена в листинге 2 (приложение А, с. 27).

Пакет bigBrother

Данный пакет включает в себя класс BigBrotherWindow, который является реализацией класса Window и определяет функционал главного экрана приложения. Реализация приведена в листинге 3 (приложение А, с. 27).

Пакет loggers

Данный пакет включает в себя пакеты реализаций логирующих классов: KeyLogger (нажатия на клавиши клавиатуры), MouseLogger (нажатия на клавиши и движения), ReactionLogger (результаты пройденных тестов на реакцию). Реализация классов приведена в листингах 5 – 8 (приложение А, с. 27).

В пакете реализован класс ReactionTestWindow, предоставляющий интерфейс и логику определения реакции пользователя по нажатию на кнопку, появляющуюся в случайные моменты времени (от 2 до 10 секунд). Код приведен в листинге 9 (приложение А, с. 27).

Каждый логирующий класс локально создает текстовый файл, в который записывает в определенном формате собранные данные. Для исключения попыток изменения файла конкурирующими потоками в каждом из них представлена реализация очереди записи, которая переносится в файл при достижения размера в сотню записей либо по завершению работы приложения.

Пакет random

Данный пакет включает в себя класс, реализованный по шаблону “Одиночка”, предоставляющий доступ к классу Random, инициализированного отложено. Данный класс используется для получения данных о реакции пользователя. Реализация класса представлена в листинге 4 (приложение А, с. 27).

Также в модуле определен файл main.kt, который является точкой входа в приложения. Код файла представлен в листинге 1 (приложение А, с. 27).

На рисунке 1.1 представлена диаграмма классов модуля.

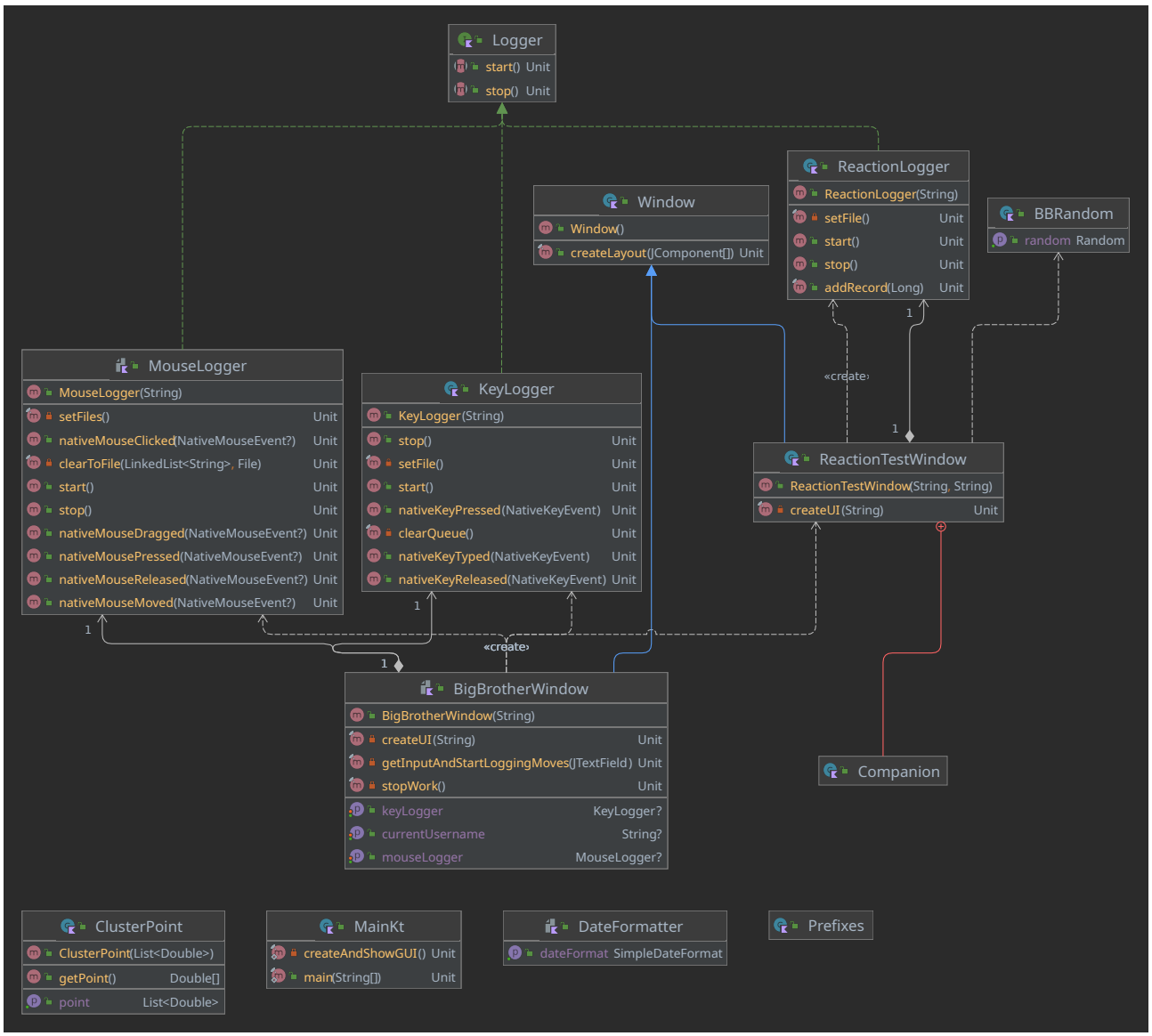

Рисунок 1.1 – Диаграмма классов модуля логирования.

Пакет window

Данный пакет включает в себя абстрактный класс Window, представляющий родительский класс с определенными свойствами для всех окон реализуемого программного обеспечения. Реализация приведена в листинге 2 (приложение А, с. 27).

1.4.2 Модуль обработки данных

Модуль синтаксического анализа данных предоставляет функции приведения записанных текстовых данных к определенным классам моделей данных.

Модуль обработки данных включает в себя два пакета: синтаксического анализа и приведения.

Пакет bbParser.models

Данный пакет включает в себя модели данных.

В листинге 10 (приложение А, с. 27) представлена реализация абстрактного класса модели, в листингах 11 – 13 (приложение А, с. 27) представлены примеры конкретных моделей.

Пакет bbParser.parsers

Данный пакет включает в себя синтаксические анализаторы для получаемых текстовых данных.

В листинге 14 (приложение А, с. 27) представлена реализация абстрактного класса синтаксического анализатора. В листинге 15 (приложение А, с. 27) представлен пример конкретного анализатора.

Пакет bbConverter

Данный пакет отвечает за получение требуемых характеристик по полученным данных, например, скорости печати.

В листинге 18 (приложение А, с. 27) представлена реализация абстрактного класса. В листинге 20 (приложение А, с. 27) и 19 (приложение А, с. 27) — примеры преобразователей.

На рисунке 1.2 представлена диаграмма классов модуля.

1.4.3 Модуль анализа данных

Данный модуль предназначен для анализа поступающей от оператора информации: кластеризации данных для модели и непосредственно текущей оценки усталости оператора.

Единственной библиотекой, используемой в модуле является The Commons Mathematics Library [11], из которой была взята реализация алгоритма кластеризации c-средних.

Пакет analyze.clusterization

Данный пакет включает в себя утилиты для кластеризации данных, в нем представлен абстрактный класс Clusterer и его реализация BbClusterer, предоставляющий функционал определения нечетких кластеров по переданным данным. Также в пакете приведен класс точки для кластеризации, используемая библиотекой The Commons Mathematics Library. Реализация приведена в

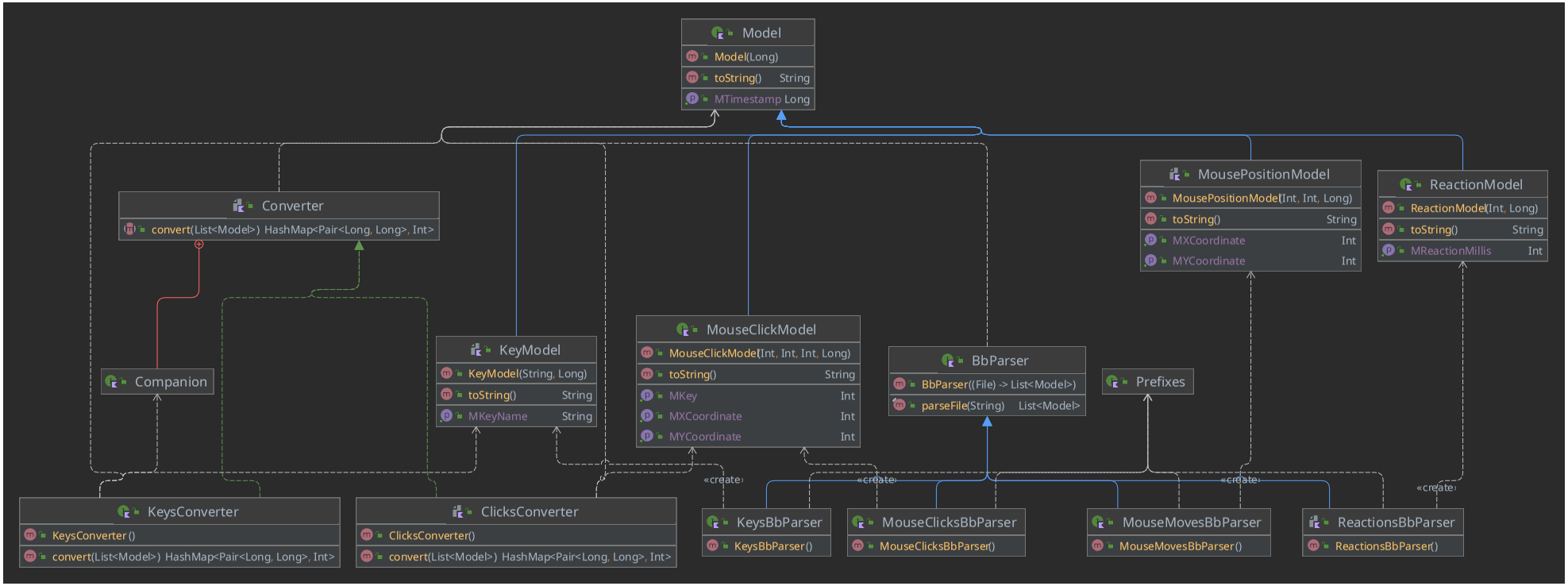

Рисунок 1.2 – Диаграмма классов модуля обработки данных.

листингах 22 — 24 (приложение А, с. 27).

Пакет analyze.analyzers Данный пакет включает в себя анализаторы, позволяющие построить модель и оценивать состояние оператора с использованием различных представлений данных и их методов обработки. В листингах 25 (приложение А, с. 27) и 26 (приложение А, с. 27) представлены абстрактный класс анализатора и его реализация для работы с файлами.

Пакет analyze.fileAnalyzer Данный пакет предоставляет реализацию класса, позволяющего полноценно создать модель по заданным файлам и определить состояния оператора. Реализация представлена в листинге 27 (приложение А, с. 27).

На рисунке 1.3 представлена диаграмма классов модуля.

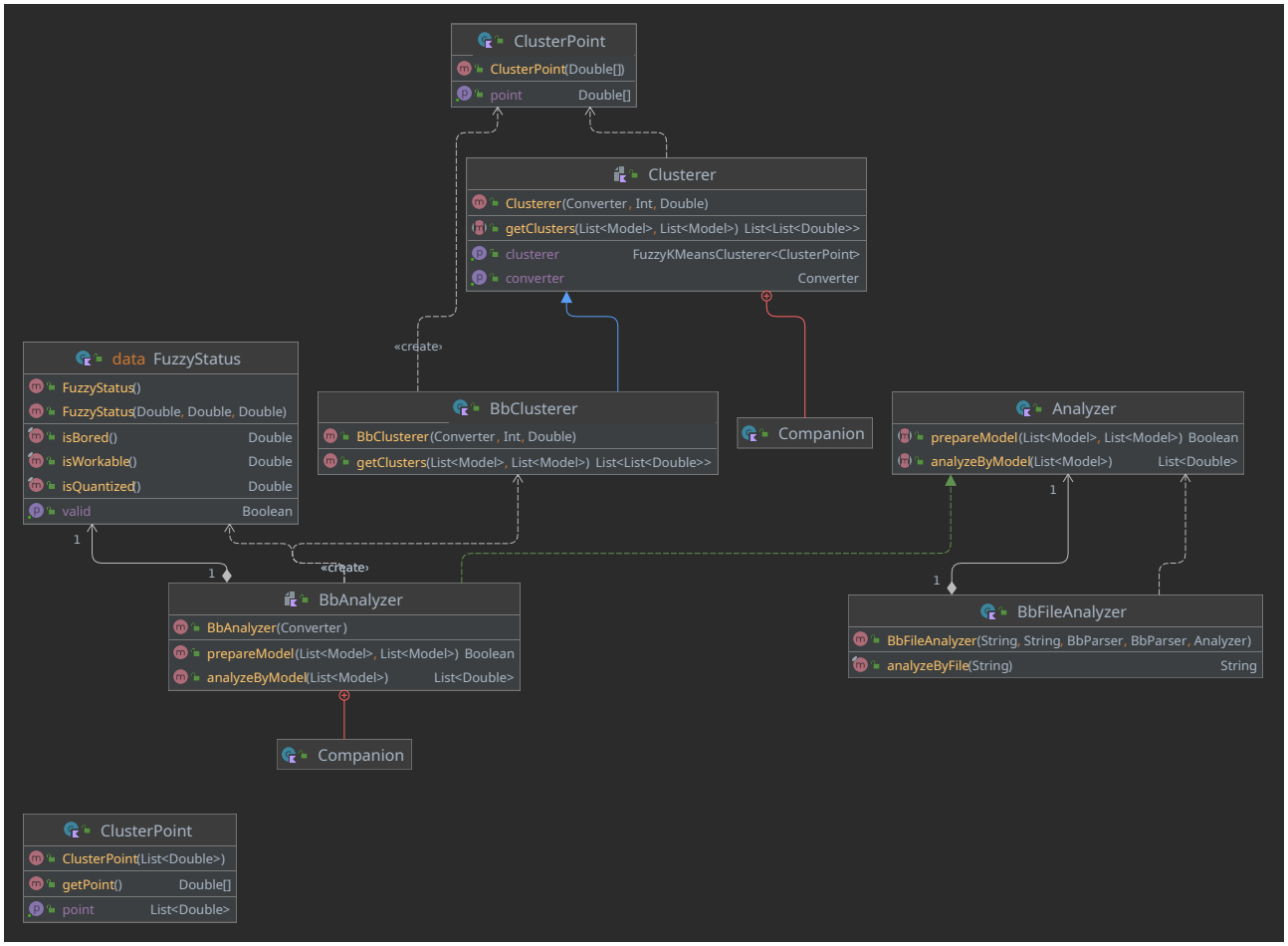

Рисунок 1.3 – Диаграмма классов модуля анализа данных.

1.4.4 Модуль серверного приложения

В данном модуле описана серверная составляющая хранилища данных, полученных от оператора. Главным назначением является хранение данных, позволяющих построить модель для пользователя в случае их утраты в локальном хранилище.

Данный модуль логически разделен на следующие части:

- пакет доступа к данным;
- пакет бизнес-логики;
- пакет реализации протокола;
- пакет клиента;
- пакет сервера.

Пакет доступа к данным

Данный пакет включает в себя реализацию двух классов: CharRepositoryImpl, основанного на шаблоне проектирования “Репозиторий”, и InfluxDAO, основаного на шаблоне проектирования “Объект доступа к данным”. Объект доступа к данным позволяет сделать репозиторий независимым от реализации исполнения запросов к базе данных. Данный объект использует InfluxDB-client-kotlin [12] для запросов на внесение и чтение записей, однако отдельный функционал реализован через отправку HTTP-запросов напрямую к серверу InfluxDB с использованием OkHttp3 [13].

Пакет бизнес-логики

Данный пакет включает в себя множество сущностей, фигурирующих между слоями клиент-серверной архитектуры.

Пакет реализации протокола

Данный пакет включает в себя класс YDVP, который может представлять собой YDVP-запрос или YDVP-ответ, единственным отличием для них будет интерпретация абстрактного класса YdvvpStartingLine, от которого наследуются классы YdvpStartingLineRequest и YdvpStartingLineRespone. Данный пакет также содержит класс YdvpParser, который предоставляет функционал обработки приходящих YDVP-запросов.

YDVP — собственный протокол приложения, основанный на версии HTTP 1.1.

Пакет клиента

Данный пакет включает в себя класс InfluxServiceClient, который позволяет подключиться к удаленному серверу, направлять ему YDVP-запросы и получать ответы.

Пакет сервера

Данный пакет включает в себя все необходимое для запуска сервера на заданном порту устройства.

В листинге 29 (приложение Б, с. 50) представлено описание класса сервера приложения. Данный класс инициализирует классы для инъекции зависимостей в модули, используемые сервисами и контроллерами, а также передает приходящих клиентов обработчик в отдельном потоке, таким образом достигается возможность одновременной обработки запросов от нескольких клиентов.

В листинге 30 (приложение Б, с. 50) представлено описание класса обработчика запроса клиента. Данный класс включает в себя необходимый для навигации по ресурсам контроллер, а также типовые объекты YDVP-ответов и методы обработки запроса клиента.

В листингах 31 и 32 (приложение Б, с. 50) представлен пример реализации взаимодействия пользователя с сервером.

На рисунках 1.4 — 1.7 представлены диаграммы классов компонентов модуля.

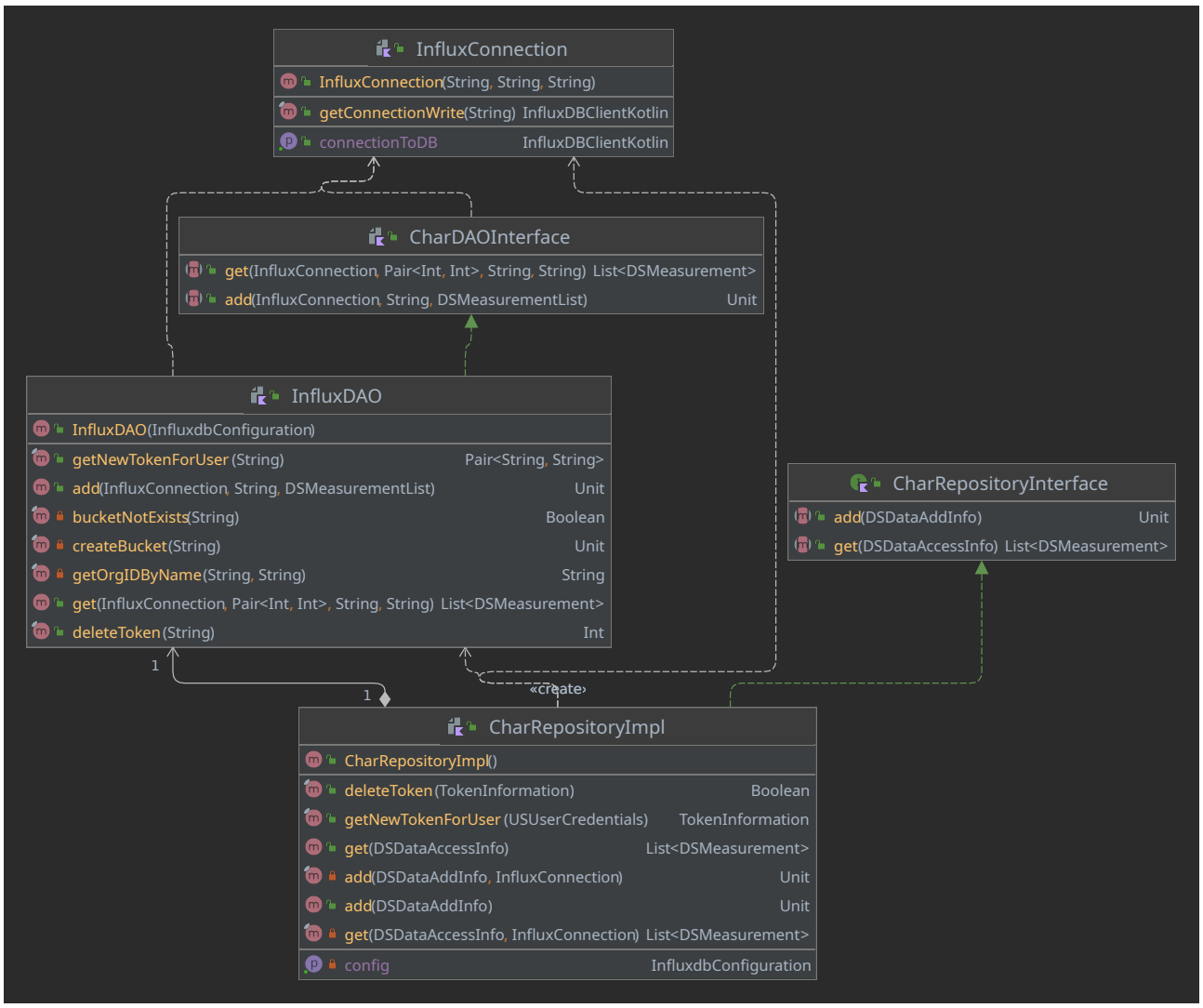

Рисунок 1.4 – Диаграмма классов слоя доступа к данным приложения.

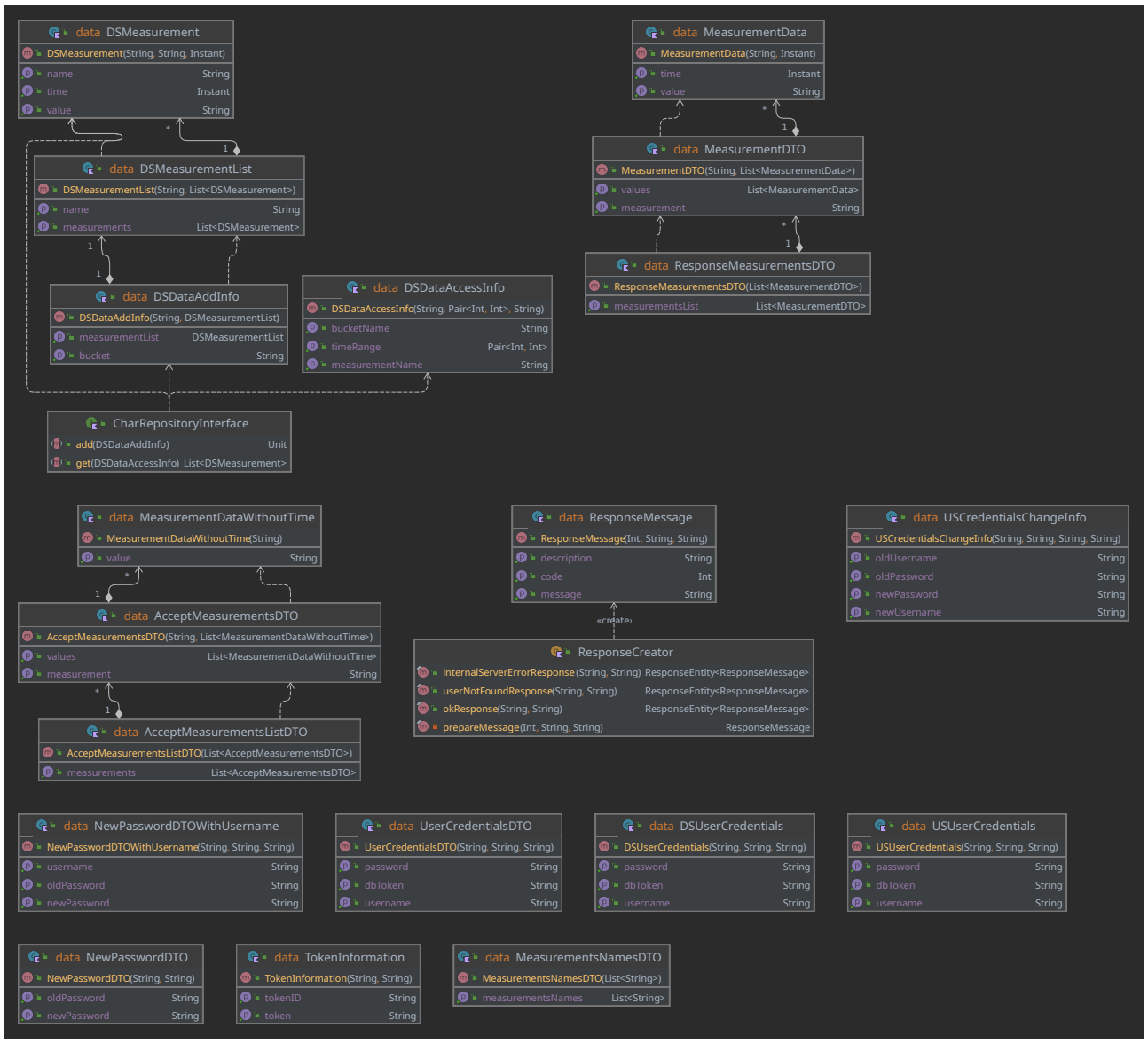

Рисунок 1.5 – Диаграмма классов бизнес-логики приложения.

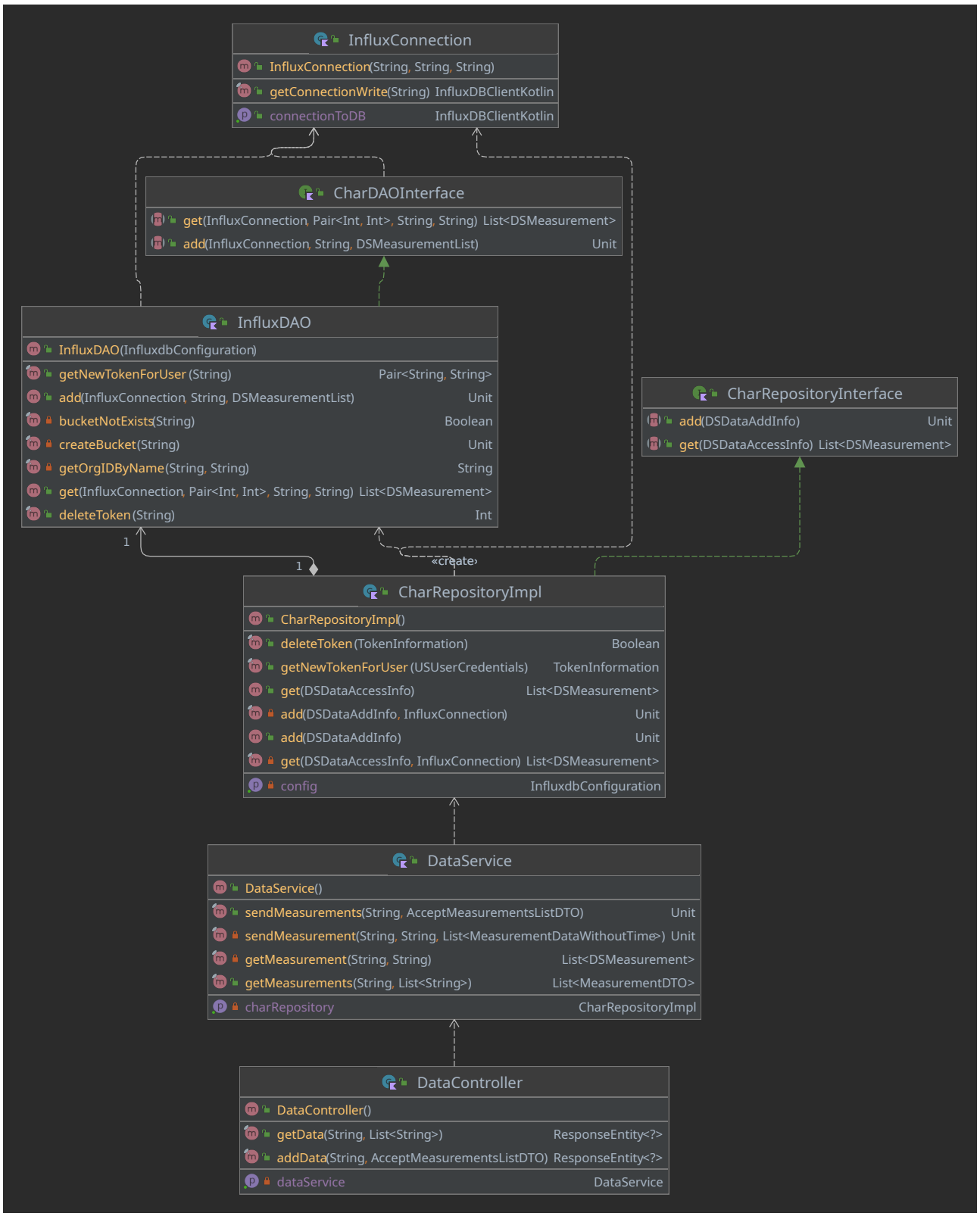

Рисунок 1.6 – Диаграмма классов, показывающая связь контроллера и слоя доступа к данным.

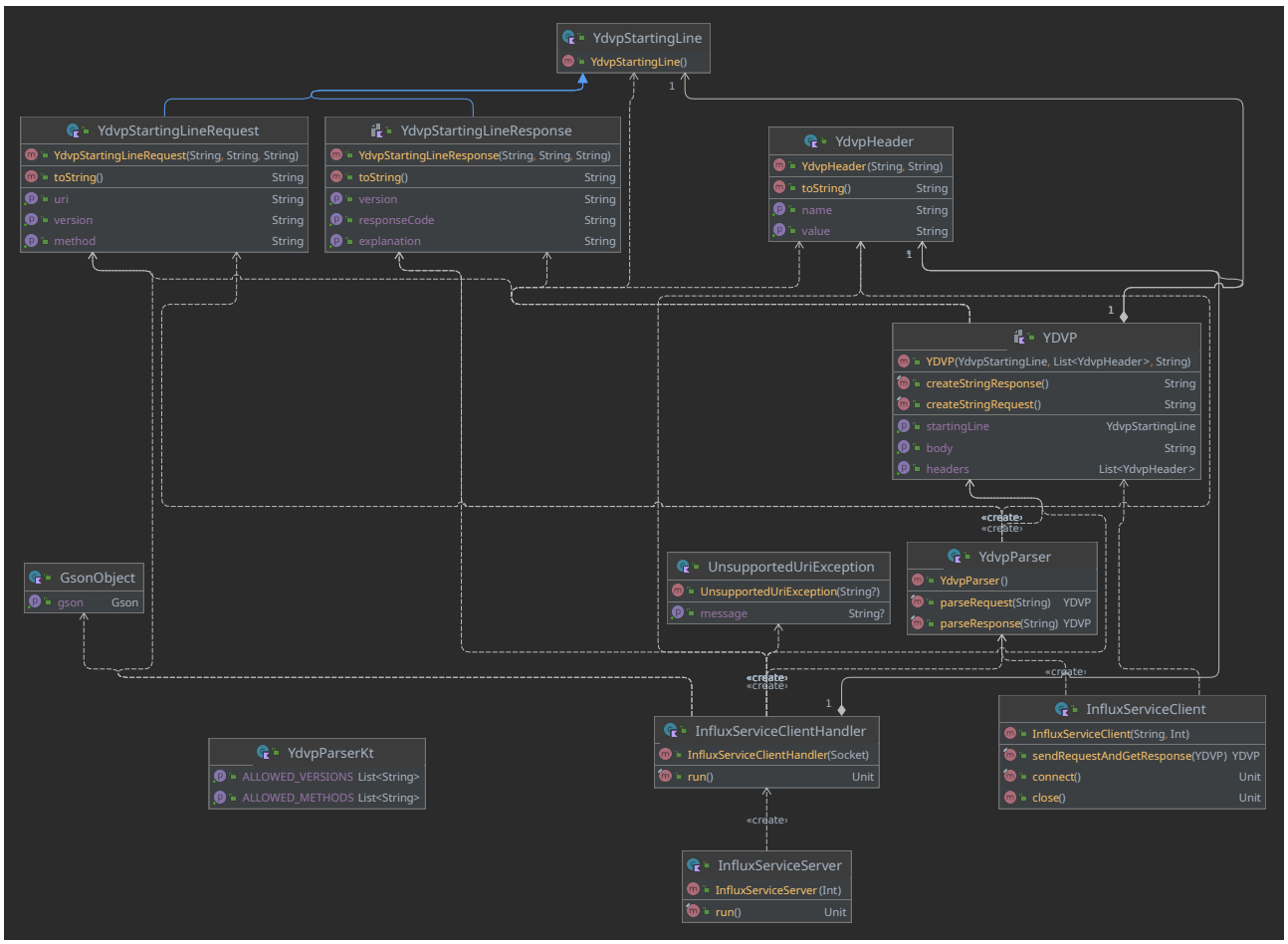

Рисунок 1.7 – Диаграмма классов серверной части приложения.

1.4.5 Пример использования модулей обработки и анализа данных

В листинге 28 представлен пример определения текущего состояния пользователя.

На рисунках 1.8 – 1.10 представлены все возможные результаты выполнения программы.

```
/home/trvehazzk3r/.jdks/corretto-15.0.2/bin/java ...
Quantized


Process finished with exit code 0
```

Рисунок 1.8 – Результат выполнения, сообщающий о том, что состояние пользователя не установлено.

```
/home/trvehazzk3r/.jdks/corretto-15.0.2/bin/java ...
Bored


Process finished with exit code 0
```

Рисунок 1.9 – Результат выполнения, сообщающий о том, что пользователь устал.

```
/home/trvehazzk3r/.jdks/corretto-15.0.2/bin/java ...
Workable


Process finished with exit code 0
```

Рисунок 1.10 – Результат выполнения, сообщающий о том, что пользователь работоспособен.

1.4.6 Пример использования серверной части приложения

В листингах 31 (приложение А, с. 27) и 32 (приложение А, с. 27) представлен пример реализации взаимодействия пользователя с сервером.

На рисунках 1.11 и 1.12

```
/home/trvehazzk3r/Downloads/jdk-11.0.12/bin/java ...
Server started on port 6666
Accepted client on /127.0.0.1:6666 from /127.0.0.1:55082
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by retrofit2.Platform (file:/home/trvehazzk3r/.gradle/caches/modules-2/files-2.1/
WARNING: Please consider reporting this to the maintainers of retrofit2.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Returned to client:
YDVP/0.1 200 OK
Server: 127.0.0.1

{"code":200,"message":"Measurements were carefully sent","description":"We know all about you now \u003e:c"}
```

Рисунок 1.11 – Результат выполнения запроса на стороне сервера.

Рисунок 1.12 – Результат выполнения запроса на стороне клиента.

Вывод

В качестве средства реализации был выбран язык программирования Kotlin, использовалась среда разработки IntelliJ IDEA.

В качестве используемой базы данных была выбрана база данных временных рядов, так как она нацелена на хранение, извлечение и анализ большого количества статистических данных. В качестве СУБД было решено использовать InfluxDB в силу отсутствия аналогов, а также по причине того, что поддержка данной СУБД все еще не прекращена на сегодняшний день.

В качестве алгоритма был выбран метод с-средних в силу того, что число кластеров заранее известно, а также задача рассматривает установку соответствия некоторого объекта набора вещественных значений, показывающих степень отношения объекта к кластерам.

Было определено, что в качестве данных для кластеризации используются действия оператора автоматизированного рабочего места, производимые с использованием клавиатуры и мыши.

Были приведены сведения и особенности модулей логирования действий оператора, анализа данных, серверного приложения.

Также были приведены примеры, которые показали возможные пути использования реализованных компонентов.

2 Апробирование метода систематического распознавания усталости на автоматизированном рабочем месте

2.1 Технические характеристики

Технические характеристики ЭВМ, на котором выполнялись исследования:

- операционная система: Manjaro Linux (5.13.19-2);
- оперативная память: 16 гигабайт;
- процессор: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz.

2.2 Сравнение количества успешных определений работоспособности пользователя путем варьирования фактора нечеткости

2.2.1 Предварительные условия

В данном исследовании используется набор данных, полученных от студента, выполняющего письменную работу. Во время сбора данных объекту исследования было разрешено отвлекаться на отдых, выполняя действия развлекательного характера в сети Интернет.

Полные данные, в силу их объема, приведены быть не могут. В листингах 33 (приложение В, с. 58) и 34 (приложение В, с. 58) приведена часть полученных по пользователю данных.

В качестве варьируемого параметра принимается критерий нечеткости при проведении кластеризации методом с-средних.

По тесту на реакцию было определено, что пользователю требуется отдых по данным, направляемым анализатору.

2.2.2 Сравнение количества успешных определений работоспособности по клавиатуре

В таблице 1 представлены результаты определений работоспособности пользователя по заданной выборке с использованием различных факторов не четкости. Для каждого значения фактора проводилось 10 определений состояния.

Таблица 1 – Количество определений состояния системой в зависимости от значения критерия нечеткости.

Значение критерия нечеткости	Количество определений состояния		
	Неопределен	Устал	Работоспособен
1.5	10	0	0
2.0	10	0	0
2.5	10	0	0
3.0	10	0	0
3.5	10	0	0
4.0	10	0	0
4.5	0	10	0
5.0	0	10	0
5.5	0	10	0
6.0	0	10	0
6.5	0	10	0
7.0	0	10	0
7.5	0	10	0
8.0	0	10	0
8.5	0	10	0
9.0	0	10	0
9.5	0	10	0
10.0	0	10	0

Полученные данные позволяют определить, что наибольшей точности можно добиться при использовании значения критерия нечеткости больше или равного 4.5, так как в таком случае шанс успешного определения состояния пользователя составляет 100%.

Вывод

В результате проведенного исследования было определено, что на заданной выборке наиболее точными являются факторы нечеткости, лежащие на отрезке от 4.5 до 10.0, так как при их использовании шанс распознать истинное состояние пользователя составляет 100%.

2.2.3 Сравнение количества успешных определений работоспособности по мыши

В таблице 2 представлены результаты определений работоспособности пользователя по заданной выборке с использованием различных факторов не четкости. Для каждого значения фактора проводилось 10 определений состоя-

ния.

Таблица 2 – Количество определений состояния системой в зависимости от значения критерия нечеткости.

Значение критерия нечеткости	Количество определений состояния		
	Неопределен	Устал	Работоспособен
1.5	0	10	0
2.0	0	10	0
2.5	0	10	0
3.0	0	10	0
3.5	0	10	0
4.0	10	0	0
4.5	8	2	0
5.0	6	4	0
5.5	9	1	0
6.0	9	1	0
6.5	10	0	0
7.0	10	0	0
7.5	10	0	0
8.0	0	10	0
8.5	10	0	0
9.0	0	10	0
9.5	0	10	0
10.0	0	10	0

Полученные данные позволяют определить, что наибольшей точности можно добиться при использовании значений критерия нечеткости 1.5 – 3.5, 8.0, 9.0 – 10.0, так как при их использовании шанс распознать истинное состояние пользователя составляет 100%.

Вывод

В результате проведенного исследования было определено, что на заданной выборке наиболее точными являются факторы нечеткости 1.5 – 3.5, 8.0, 9.0 – 10.0, так как при их использовании шанс распознать истинное состояние пользователя составляет 100%. Причем критерий нечеткости 5.0 позволил с шансом в 40% распознать опасное состояние оператора, в то время как оставшиеся критерии имели шанс в 20% и 10%.

Вывод

Исследование в области сравнения времени исполнения запросов в базы данных показало преимущество скорости исполнения запросов в базу данных InfluxDB над скоростью исполнения запросов в базу данных Postgres с использованием языка программирования Kotlin. Результаты показали, что время исполнения запросов в InfluxDB при количестве записей в таблице от 10 тысяч до 28 тысяч практически можно приравнять к константе. При этом в среднем InfluxDB позволяет получить ответ в ≈ 4.53 раза быстрее, чем Postgres.

Исследования в области сравнения количества успешных определений работоспособности пользователя показали, что при распознавании усталости с использованием клавиатуры при варьировании фактора нечеткости было получено $\approx 67\%$ верных результатов, что на 13% больше, чем при использовании мыши с точностью 54%. Таким образом, на заданной выборке более эффективным устройством для распознавания усталости оказалась клавиатура. При этом для клавиатуры наиболее точными факторами нечеткости являются значения, лежащие на отрезке от 4.5 до 10.0, которые позволили с точностью в 100% определить истинное состояние оператора. Для мыши наиболее точными факторами нечеткости являются 1.5–3.5, 8.0, 9.0–10.0.

ЗАКЛЮЧЕНИЕ

Был реализован метод систематического распознавания усталости на рабочем месте. Реализация включила в себя три модуля: логирования действий оператора, анализа данных и серверного приложения. Были приведены особенности реализации каждого модуля, диаграммы классов.

Было проведено исследование в области сравнения количества успешных определения работоспособности пользователя путем варьирования
фактора нечеткости. В результате было определено, что при определении усталости с использованием клавиатуры на заданной выборке при варьировании фактора нечеткости было получено $\approx$ 67% верных результатов, что на 17% больше, чем при использовании мыши. Эффективным устройством для распознавания усталости в данных испытаниях была признана клавиатура.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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. М.: 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. Kotlin language specification [Электронный ресурс]. Режим доступа: https://kotlinlang.org/spec/introduction.html (дата обращения 09.10.2020).

6. Шабельников А.Н. Шабельников В.А. Поиск аномалий в технических базах данных временных рядов // Известия ЮФУ. Технические науки. 2008. № 4.

7. TProger : Знакомство с InfluxDB и базами данных временных рядов [Электронный ресурс]. Режим доступа: https://tproger.ru/translations/influxdb-guide/ (дата обращения 01.04.2020).

8. Н.Р. Булахов. Основы реляционных баз данных // Вестник науки и образования. 2019. Т. 76, № 22-2.

9. Документация Oracle - Java Swing [Электронный ресурс]. Режим доступа: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html (дата обращения 16.05.2022).

10. Официальный репозиторий GitHub проекта JNativeHook [Электронный ресурс]. Режим доступа: https://github.com/kwhat/jnativehook (дата обращения 16.05.2022).

11. Apache Commons: официальный сайт [Электронный ресурс]. Режим доступа: https://commons.apache.org/proper/commons-math/ (дата обращения 20.05.2022).

12. InfluxDB Client Kotlin: официальный репозиторий [Электронный ресурс]. Режим доступа: https://github.com/influxdata/influxdb-client-java/tree/master/client-kotlin (дата обращения 20.05.2022).

13. OkHttp: официальная страница [Электронный ресурс]. Режим доступа: https://square.github.io/okhttp/ (дата обращения 20.05.2022).

ПРИЛОЖЕНИЕ А

Иходный код модулей обработки и анализа данных

Листинг 1: Файл main.kt

```
 1 import bigBrother.BigBrotherWindow
 2 import java.awt.EventQueue
 3
 4 private fun createAndShowGUI() {
 5     val frame = BigBrotherWindow("BB Неактивен()")
 6     frame.isVisible = true
 7 }
 8
 9 fun main(args: Array<String>) {
10     EventQueue.invokeLater(::createAndShowGUI)
11 }
```

Листинг 2: Файл Window.kt

```
 1 package window
 2
 3 import javax.swing.GroupLayout
 4 import javax.swing.ImageIcon
 5 import javax.swing.JComponent
 6 import javax.swing.JFrame
 7
 8 open class Window: JFrame() {
 9
10     init {
11         iconImage = ImageIcon(this.javaClass.getResource("/bb.
               png")).image
12     }
13
14     fun createLayout(vararg components: JComponent) {
15         val gl = GroupLayout(contentPane)
16         contentPane.layout = gl
17
18         gl.autoCreateContainerGaps = true
19
20         val horizontalGroup = gl.createParallelGroup(
```

```
            GroupLayout.Alignment.CENTER)
21          components.forEach { horizontalGroup.addComponent(it) }
22
23          gl.setHorizontalGroup(horizontalGroup)
24
25
26          val verticalGroup = gl.createSequentialGroup()
27          components.forEach { verticalGroup.addComponent(it) }
28
29          gl.setVerticalGroup(verticalGroup)
30
31          pack()
32      }
33 }
```

Листинг 3: Файл BigBrotherWindow.kt

```
 1 package bigBrother
 2
 3 import loggers.keyLogger.KeyLogger
 4 import loggers.mouseLogger.MouseLogger
 5 import loggers.reactionTest.ReactionTestWindow
 6 import window.Window
 7 import java.io.File
 8 import java.lang.Exception
 9 import javax.swing.*
10
11 class BigBrotherWindow(title: String) : Window() {
12
13     var currentUsername: String? = null
14
15     var mouseLogger: MouseLogger? = null
16     var keyLogger: KeyLogger? = null
17
18     init {
19         try {
20             UIManager.setLookAndFeel(UIManager.
                   getSystemLookAndFeelClassName())
21         } catch (ex: Exception) {
22         }
23
24         File("${System.getProperty("user.dir")}/data").mkdir()
```

```
25
26          createUI(title)
27      }
28
29      private fun getInputAndStartLoggingMoves(textField:
            JTextField) {
30          currentUsername = textField.text.trim()
31
32          mouseLogger = MouseLogger(currentUsername!!)
33          keyLogger = KeyLogger(currentUsername!!)
34
35          mouseLogger?.start()
36          keyLogger?.start()
37      }
38
39      private fun stopWork() {
40          mouseLogger!!.stop()
41          keyLogger!!.stop()
42      }
43
44      private fun createUI(title: String) {
45
46          setTitle(title)
47
48          val input = JTextField("ФамилияИмя__Ваш( факультет)xxx-
                ")
49          input.horizontalAlignment = JTextField.CENTER
50
51          val goButton = JButton("Начать слежку")
52          goButton.addActionListener {
53              getInputAndStartLoggingMoves(input)
54              JOptionPane.showMessageDialog(
55                  this,
56                  "Наблюдаю зажизнедеятельностью ",
57                  "Изменение статуса",
58                  JOptionPane.INFORMATION_MESSAGE
59              )
60              setTitle("BB активен()")
61          }
62
63          val stopButton = JButton("Остановить слежку")
```

```
64          stopButton.addActionListener {
65              stopWork()
66              JOptionPane.showMessageDialog(
67                  this,
68                  "Закончил наблюдениезажизнедеятельностью  ",
69                  "Изменение статуса",
70                  JOptionPane.INFORMATION_MESSAGE
71              )
72              setTitle("BB неактивен()")
73          }
74
75          val checkReactionButton = JButton("Проверить реакцию")
76          checkReactionButton.addActionListener {
77              val frame = ReactionTestWindow("Тест реакции",
                    currentUsername ?: input.text.trim())
78              frame.isVisible = true
79          }
80
81          createLayout(input, goButton, stopButton,
                checkReactionButton)
82
83          defaultCloseOperation = JFrame.EXIT_ON_CLOSE
84          setSize(400, 200)
85          setLocationRelativeTo(null)
86      }
87 }
```

Листинг 4: Файл BBRandom.kt

```
1 package random
2
3 import java.util.*
4
5 object BBRandom {
6     val random by lazy { Random() }
7 }
```

Листинг 5: Файл Logger.kt

```
1 package loggers
2
3 interface Logger {
4     fun start()
```

```
5
6     fun stop()
7 }
```

Листинг 6: Файл KeyLogger.kt

```
1 package loggers.keyLogger
2
3 import com.github.kwhat.jnativehook.GlobalScreen
4 import com.github.kwhat.jnativehook.NativeHookException
5 import com.github.kwhat.jnativehook.keyboard.NativeKeyEvent
6 import loggers.Logger
7 import com.github.kwhat.jnativehook.keyboard.NativeKeyListener
8 import java.io.File
9 import java.util.LinkedList
10 import kotlin.system.exitProcess
11
12 class KeyLogger(username: String) : Logger, NativeKeyListener {
13     private val mPathToFile = "${System.getProperty("user.dir")
          }/data/${username}_Keys.txt"
14
15     private var mFile = File(mPathToFile)
16
17     @Volatile
18     private var queueToWrite = LinkedList<String>()
19
20     private fun setFile() {
21         if (!mFile.exists()) mFile.createNewFile()
22     }
23
24     override fun start() {
25         try {
26             if (!GlobalScreen.isNativeHookRegistered())
                  GlobalScreen.registerNativeHook()
27         } catch (ex: NativeHookException) {
28             System.err.println("There was a problem registering
                   the native hook.")
29             exitProcess(1)
30         }
31
32         setFile()
33
```

```
34          GlobalScreen.addNativeKeyListener(this)
35      }
36
37      private fun clearQueue() {
38          queueToWrite.forEach {
39              mFile.appendText(it)
40          }
41          queueToWrite.clear()
42      }
43
44      override fun stop() {
45          clearQueue()
46
47          try {
48              GlobalScreen.removeNativeKeyListener(this)
49              GlobalScreen.unregisterNativeHook()
50          } catch (e: Exception) {
51          }
52      }
53
54      override fun nativeKeyPressed(e: NativeKeyEvent) {
55          if (!e.isActionKey) {
56              queueToWrite.add(
57                  "key=${NativeKeyEvent.getKeyText(e.keyCode)},"
                        + " timestamp=${System.currentTimeMillis()}\
                        n"
58              )
59              if (queueToWrite.size == 100) {
60                  clearQueue()
61              }
62          }
63      }
64
65      override fun nativeKeyReleased(e: NativeKeyEvent) {
66      }
67
68      override fun nativeKeyTyped(e: NativeKeyEvent) {
69      }
70  }
```

Листинг 7: Файл MouseLogger.kt

```
 1 package loggers.mouseLogger
 2
 3 import com.github.kwhat.jnativehook.GlobalScreen
 4 import com.github.kwhat.jnativehook.NativeHookException
 5 import com.github.kwhat.jnativehook.mouse.NativeMouseEvent
 6 import com.github.kwhat.jnativehook.mouse.
     NativeMouseInputListener
 7 import loggers.Logger
 8 import java.io.File
 9 import java.util.*
10 import kotlin.system.exitProcess
11
12 class MouseLogger(username: String) : Logger,
     NativeMouseInputListener {
13
14     private val mPathToFileForMoves = "${System.getProperty("
         user.dir")}/data/${username}_Mouse_Moves.txt"
15     private val mPathToFileForClicks = "${System.getProperty("
         user.dir")}/data/${username}_Mouse_Clicks.txt"
16
17     @Volatile
18     private var mFileForMoves = File(mPathToFileForMoves)
19
20     @Volatile
21     private var mFileForClicks = File(mPathToFileForClicks)
22
23     @Volatile
24     private var queueOfMoves = LinkedList<String>()
25
26     @Volatile
27     private var queueOfClicks = LinkedList<String>()
28
29     private fun setFiles() {
30         if (!mFileForMoves.exists()) mFileForMoves.
             createNewFile()
31         if (!mFileForClicks.exists()) mFileForClicks.
             createNewFile()
32     }
33
34     private fun clearToFile(queue: LinkedList<String>, file:
```

```
          File) {
35          queue.forEach {
36              file.appendText(it)
37          }
38          queue.clear()
39      }
40
41      override fun start() {
42          try {
43              if (!GlobalScreen.isNativeHookRegistered())
44                  GlobalScreen.registerNativeHook()
45          } catch (ex: NativeHookException) {
46              System.err.println("There was a problem registering
                    the native hook.")
47              exitProcess(1)
48          }
49
50          setFiles()
51
52          GlobalScreen.addNativeMouseListener(this)
53          GlobalScreen.addNativeMouseMotionListener(this)
54      }
55
56      override fun stop() {
57          clearToFile(queueOfClicks, mFileForClicks)
58          clearToFile(queueOfMoves, mFileForMoves)
59
60          try {
61              GlobalScreen.removeNativeMouseListener(this)
62              GlobalScreen.removeNativeMouseMotionListener(this)
63              GlobalScreen.unregisterNativeHook()
64          } catch (e: Exception) {
65          }
66      }
67
68      override fun nativeMouseClicked(e: NativeMouseEvent?) {
69          queueOfClicks.add(
70              "x=${e!!.x}, y=${e.y}, key=${e.button}," +
71                      " timestamp=${System.currentTimeMillis()}\n
                           "
72          )
```

```
 73
 74         if (queueOfClicks.size == 100) {
 75             clearToFile(queueOfClicks, mFileForClicks)
 76         }
 77     }
 78
 79     override fun nativeMousePressed(p0: NativeMouseEvent?) {
 80
 81     }
 82
 83     override fun nativeMouseReleased(p0: NativeMouseEvent?) {
 84
 85     }
 86
 87     override fun nativeMouseMoved(e: NativeMouseEvent?) {
 88         queueOfMoves.add(
 89             "x=${e!!.x}, y=${e.y}," +
 90                     " timestamp=${System.currentTimeMillis()}\n
                        "
 91         )
 92
 93         if (queueOfMoves.size == 100) {
 94             clearToFile(queueOfMoves, mFileForMoves)
 95         }
 96     }
 97
 98     override fun nativeMouseDragged(p0: NativeMouseEvent?) {
 99
100     }
101
102 }
```

Листинг 8: Файл ReactionLogger.kt

```
 1 package loggers.reactionTest
 2
 3 import loggers.Logger
 4 import java.io.File
 5
 6 class ReactionLogger(username: String) : Logger {
 7
 8     private val mPathToFile = "${System.getProperty("user.dir")
```

```
        }/data/${username}_Reactions.txt"

    private val mFile = File(mPathToFile)

    private fun setFile() {
        if (!mFile.exists()) mFile.createNewFile()
    }

    override fun start() {
        setFile()
    }

    fun addRecord(resultInMillis: Long) {
        mFile.appendText(
            "reaction_time=${resultInMillis}, " +
                    "timestamp=${System.currentTimeMillis()}\n"
        )
    }

    override fun stop() {}
}
```

Листинг 9: Файл ReactionTestWindow.kt

```
package loggers.reactionTest

import random.BBRandom
import window.Window
import java.awt.Font
import java.lang.Thread.sleep
import javax.swing.*
import kotlin.concurrent.thread

class ReactionTestWindow(title: String, username: String) :
    Window() {

    @Volatile
    private var testButtonPressed = false

    @Volatile
    private var reactionTimestamp = 0L

```

```
18      private var reactionsTotal = 0L
19
20      private var reactionLogger: ReactionLogger
21
22      init {
23          reactionLogger = ReactionLogger(username)
24
25          reactionLogger.start()
26
27          createUI(title)
28      }
29
30      private fun createUI(title: String) {
31
32          setTitle(title)
33
34          val testButton = JButton("Жать сюда!")
35          testButton.font = Font("Arial", Font.PLAIN, 50)
36          testButton.addActionListener {
37              reactionTimestamp = System.currentTimeMillis()
38              testButtonPressed = true
39          }
40          testButton.isVisible = true
41
42          val spacer = JLabel(" ")
43          spacer.font = Font("Arial", Font.PLAIN, 50)
44
45          val startButton = JButton("Начать тест")
46          startButton.addActionListener {
47              startButton.isVisible = false
48              thread {
49                  var startTime: Long
50                  for (i in 0 until NUMBER_OF_TESTS) {
51                      sleep((3 + BBRandom.random.nextInt(8).
                            toLong()) * 1000)
52                      testButton.isVisible = true
53                      startTime = System.currentTimeMillis()
54                      while (!testButtonPressed) {
55                          sleep(20)
56                      }
57                      reactionsTotal += reactionTimestamp -
```

```
                        startTime
58                              testButton.isVisible = false
59                      testButtonPressed = false
60                  }
61
62                  this.isVisible = false
63
64                  reactionLogger.addRecord(reactionsTotal /
                        NUMBER_OF_TESTS)
65              }
66          }
67
68
69          createLayout(testButton, spacer, startButton)
70
71          testButton.isVisible = false
72
73          setLocationRelativeTo(null)
74      }
75
76      companion object {
77          private const val NUMBER_OF_TESTS = 10
78      }
79 }
```

Листинг 10: Файл Model.kt

```
 1 package bbParser.models
 2
 3 import java.util.*
 4
 5 abstract class Model(val mTimestamp: Long) {
 6
 7     override fun toString(): String {
 8         return "timestamp=$mTimestamp]"
 9     }
10 }
```

Листинг 11: Файл KeyModel.kt

```
 1 package bbParser.models
 2
 3 import dateFormat.DateFormatter
```

```
 4
 5 class KeyModel(
 6     val mKeyName: String,
 7     timestamp: Long
 8 ): Model(timestamp) {
 9
10     override fun toString(): String {
11         return "[key=$mKeyName; " + super.toString()
12     }
13 }
```

Листинг 12: Файл MouseClickModel.kt

```
 1 package bbParser.models
 2
 3 class MouseClickModel(
 4     val mXCoordinate: Int,
 5     val mYCoordinate: Int,
 6     val mKey: Int,
 7     timestamp: Long
 8 ): Model(timestamp) {
 9
10     override fun toString(): String {
11         return "[x=$mXCoordinate; y=$mYCoordinate; key=$mKey; "
               + super.toString()
12     }
13 }
```

Листинг 13: Файл ReactionModel.kt

```
 1 package bbParser.models
 2
 3 class ReactionModel(
 4     val mReactionMillis: Int,
 5     timestamp: Long
 6 ): Model(timestamp) {
 7
 8     override fun toString(): String {
 9         return "[reaction_time=$mReactionMillis; " + super.
              toString()
10     }
11 }
```

Листинг 14: Файл BbParser.kt

```
1 package bbParser.parsers
2
3 import bbParser.models.Model
4 import java.io.File
5
6 abstract class BbParser(private val parseFun: (File) -> List<
    Model>) {
7
8     fun parseFile(path: String): List<Model> {
9         val file = File(path)
10        return if (file.exists() && file.canRead()) parseFun(
              file) else listOf()
11    }
12 }
```

Листинг 15: Файл KeysBbParser.kt

```
1 package bbParser.parsers
2
3 import bbParser.models.KeyModel
4 import bbParser.prefixes.Prefixes
5
6 class KeysBbParser : BbParser(
7     { file ->
8         file.readLines().map { line ->
9             val strValues = line.split(',')
10            KeyModel(
11                strValues[0].trim().removePrefix(Prefixes.KEY),
12                strValues[1].trim().removePrefix(Prefixes.
                      TIMESTAMP).toLong()
13            )
14        }
15    }
16 )
```

Листинг 16: Файл MouseClicksBbParser.kt

```
1 package bbParser.parsers
2
3 import bbParser.models.MouseClickModel
4 import bbParser.prefixes.Prefixes
5
```

```
 6 class MouseClicksBbParser : BbParser(
 7     { file ->
 8         file.readLines().map { line ->
 9             val strValues = line.split(',')
10             MouseClickModel(
11                 strValues[0].trim().removePrefix(Prefixes.
                       X_COORDINATE).toInt(),
12                 strValues[1].trim().removePrefix(Prefixes.
                       Y_COORDINATE).toInt(),
13                 strValues[2].trim().removePrefix(Prefixes.KEY).
                       toInt(),
14                 strValues[3].trim().removePrefix(Prefixes.
                       TIMESTAMP).toLong()
15             )
16         }
17     }
18 )
```

Листинг 17: Файл ReactionsBbParser.kt

```
 1 package bbParser.parsers
 2
 3 import bbParser.models.ReactionModel
 4 import bbParser.prefixes.Prefixes
 5
 6 class ReactionsBbParser : BbParser(
 7     { file ->
 8         file.readLines().map { line ->
 9             val strValues = line.split(',')
10             ReactionModel(
11                 strValues[0].trim().removePrefix(Prefixes.
                       REACTION).toInt(),
12                 strValues[1].trim().removePrefix(Prefixes.
                       TIMESTAMP).toLong()
13             )
14         }
15     }
16 )
```

Листинг 18: Файл Converter.kt

```
 1 package bbConverter
 2
```

```
 3 import bbParser.models.Model
 4 import java.util.Date
 5 import kotlin.collections.HashMap
 6
 7 interface Converter {
 8     fun convert(models: List<Model>): HashMap<Pair<Long, Long>,
         Int>
 9
10     companion object {
11         const val MILLIS_IN_MINUTE = 1000 * 60
12     }
13 }
```

Листинг 19: Файл ClicksConverter.kt

```
 1 package bbConverter
 2
 3 import bbParser.models.Model
 4 import bbParser.models.MouseClickModel
 5 import kotlin.math.sqrt
 6
 7 @Suppress("UNCHECKED_CAST")
 8 class ClicksConverter : Converter {
 9
10     override fun convert(clicks: List<Model>): HashMap<Pair<
          Long, Long>, Int> {
11         val out = HashMap<Pair<Long, Long>, Int>()
12
13         val sortedClicks = (clicks as List<MouseClickModel>).
             sortedBy { it.mTimestamp }
14
15         var prevClick: MouseClickModel
16         var curClick: MouseClickModel
17         for (i in 1 until sortedClicks.size) {
18             prevClick = sortedClicks[i - 1]
19             curClick = sortedClicks[i]
20             out[Pair(prevClick.mTimestamp, curClick.mTimestamp)
                 ] =
21                 sqrt(
22                     ((curClick.mXCoordinate - prevClick.
                         mXCoordinate) *
23                             (curClick.mXCoordinate - prevClick.
```

```
                                    mXCoordinate) +
24                                (curClick.mYCoordinate - prevClick.
                                    mYCoordinate) *
25                                (curClick.mYCoordinate - prevClick.
                                    mYCoordinate)).toDouble()
26                  ).toInt()
27          }
28
29          return out
30      }
31  }
```

Листинг 20: Файл KeysConverter.kt

```
 1 package bbConverter
 2
 3 import bbParser.models.KeyModel
 4 import bbParser.models.Model
 5 import kotlin.collections.HashMap
 6
 7 @Suppress("UNCHECKED_CAST")
 8 class KeysConverter : Converter {
 9
10     override fun convert(keys: List<Model>): HashMap<Pair<Long,
           Long>, Int> {
11         val out = hashMapOf<Pair<Long, Long>, Int>()
12
13         val sortedKeys = (keys as List<KeyModel>).sortedBy { it
               .mTimestamp }
14
15         var currentTimestamp = sortedKeys.first().mTimestamp
16         var passedKeys = 1
17         for (i in 1 until sortedKeys.size) {
18             passedKeys++
19             if (sortedKeys[i].mTimestamp - currentTimestamp >
                   Converter.MILLIS_IN_MINUTE) {
20                 out[Pair(currentTimestamp, sortedKeys[i - 1].
                       mTimestamp)] = passedKeys
21                 passedKeys = 1
22                 currentTimestamp = sortedKeys[i].mTimestamp
23             }
24         }
```

```
25
26          out[Pair(currentTimestamp, sortedKeys.last().mTimestamp
                )] = passedKeys
27
28          return out
29      }
30  }
```

Листинг 21: Файл FuzzyStatus.kt

```
 1 package analyze.models
 2
 3
 4 data class FuzzyStatus(
 5      val isQuantized: Double = 0.0,
 6      val isBored: Double = 0.0,
 7      val isWorkable: Double = 0.0
 8 ) {
 9
10      fun isValid(): Boolean {
11          return (isQuantized > 0.0 || isBored > 0.0 ||
                    isWorkable > 0.0) &&
12                  (isQuantized != isBored && isBored !=
                        isWorkable && isQuantized != isWorkable)
13      }
14 }
```

Листинг 22: Файл Clusterer.kt

```
 1 package analyze.clusterization
 2
 3 import bbConverter.Converter
 4 import bbParser.models.Model
 5 import org.apache.commons.math3.ml.clustering.
       FuzzyKMeansClusterer
 6
 7 abstract class Clusterer(val converter: Converter, k: Int = 3,
       fuzziness: Double = 5.0) {
 8
 9      val clusterer = FuzzyKMeansClusterer<ClusterPoint>(k,
            fuzziness)
10
11      abstract fun getClusters(fDimension: List<Model>,
```

```
        sDimension: List<Model>): List<List<Double>>
12
13      companion object {
14          const val MILLIS_IN_MINUTE = 1000 * 60
15      }
16  }
```

Листинг 23: Файл ClusterPoint.kt

```
 1  package analyze.clusterization
 2
 3  import org.apache.commons.math3.ml.clustering.Clusterable
 4
 5  class ClusterPoint(private val point: DoubleArray) :
        Clusterable {
 6
 7      override fun getPoint(): DoubleArray {
 8          return point
 9      }
10  }
```

Листинг 24: Файл BbClusterer.kt

```
 1  package analyze.clusterization
 2
 3  import bbConverter.Converter
 4  import bbParser.models.Model
 5  import bbParser.models.ReactionModel
 6
 7  class BbClusterer(converter: Converter, k: Int = 3, fuzziness:
        Double = 5.0) : Clusterer(converter, k, fuzziness) {
 8
 9      override fun getClusters(fDimension: List<Model>, reactions
           : List<Model>): List<List<Double>> {
10          val converted = converter.convert(fDimension)
11          reactions as List<ReactionModel>
12
13          val clusterable = mutableListOf<ClusterPoint>().apply {
14              reactions.forEach { reaction ->
15                  converted.entries.filter { filIt ->
16                      reaction.mTimestamp - filIt.key.first in
                            (0..(10 * MILLIS_IN_MINUTE))
17                  }.sortedBy { it.key.first }.forEach { innerIt
```

```
                    ->
18                   add(
19                       ClusterPoint(
20                           doubleArrayOf(
21                               reaction.mReactionMillis.
                                    toDouble(),
22                               (innerIt.value / (innerIt.key.
                                    second - innerIt.key.first).
                                    toDouble() * 1000 * 60)
23                           )
24                       )
25                   )
26               }
27           }
28       }
29
30       clusterer.cluster(clusterable)
31       return clusterer.clusters.map { listOf(it.center.point
             [0], it.center.point[1]) }
32   }
33 }
```

Листинг 25: Файл Analyzer.kt

```
 1 package analyze.analyzers
 2
 3 import bbParser.models.Model
 4
 5 interface Analyzer {
 6
 7     fun prepareModel(fDim: List<Model>, sDim: List<Model>):
         Boolean
 8
 9     fun analyzeByModel(values: List<Model>): List<Double>
10 }
```

Листинг 26: Файл BbAnalyzer.kt

```
 1 package analyze.analyzers
 2
 3 import analyze.clusterization.BbClusterer
 4 import analyze.models.FuzzyStatus
 5 import bbConverter.Converter
```

```
 6 import bbParser.models.Model
 7 import kotlin.math.abs
 8
 9 class BbAnalyzer(private val converter: Converter) : Analyzer {
10
11     private var fuzzyStatus = FuzzyStatus()
12
13     override fun prepareModel(fDim: List<Model>, reactions:
           List<Model>): Boolean {
14         val gotCenters = BbClusterer(converter).getClusters(
               fDim, reactions)
15
16         val centers = gotCenters.sortedBy { it.first() }.map {
               it[1] }
17
18         fuzzyStatus = FuzzyStatus(centers[0], centers[1],
               centers[2])
19
20         return true
21     }
22
23     override fun analyzeByModel(values: List<Model>): List<
           Double> {
24         if (!fuzzyStatus.isValid()) return emptyList()
25
26         var average = 0.0
27         converter.convert(values).entries.sortedBy { it.key.
               first }
28             .takeLast(NUMBER_OF_RECORDS_TO_BE_TAKEN)
29             .forEach { average += it.value }
30
31         average /= NUMBER_OF_RECORDS_TO_BE_TAKEN
32
33         return listOf(
34             abs(average - fuzzyStatus.isQuantized),
35             abs(average - fuzzyStatus.isBored),
36             abs(average - fuzzyStatus.isWorkable)
37         )
38     }
39
40     companion object {
```

```
41         const val NUMBER_OF_RECORDS_TO_BE_TAKEN = 5
42     }
43 }
```

Листинг 27: Файл BbFileAnalyzer.kt

```
 1 package analyze.fileAnalyzer
 2
 3 import analyze.analyzers.Analyzer
 4 import bbParser.parsers.BbParser
 5
 6 class BbFileAnalyzer(
 7     modelPathFileFDimension: String,
 8     modelPathFileSDimension: String,
 9     private val mMainParser: BbParser,
10     slaveParser: BbParser,
11     private val mAnalyzer: Analyzer
12 ) {
13
14     init {
15         val fDim = mMainParser.parseFile(
              modelPathFileFDimension)
16         val sDim = slaveParser.parseFile(
              modelPathFileSDimension)
17
18         mAnalyzer.prepareModel(fDim, sDim)
19     }
20
21     fun analyzeByFile(filepath: String): String {
22         val dim = mMainParser.parseFile(filepath)
23
24         val verdictList = mAnalyzer.analyzeByModel(dim)
25
26         return when (verdictList.indexOf(verdictList.minOrNull
              ())) {
27             0 -> "Quantized"
28             1 -> "Bored"
29             else -> "Workable"
30         }
31     }
32 }
```

Листинг 28: Файл analyze/main.kt

```
 1 package analyze
 2
 3 import analyze.analyzers.BbAnalyzer
 4 import bbConverter.KeysConverter
 5 import bbParser.parsers.KeysBbParser
 6 import bbParser.parsers.ReactionsBbParser
 7 import analyze.fileAnalyzer.BbFileAnalyzer
 8
 9 fun main() {
10
11     val fileAnalyzer = BbFileAnalyzer(
12         "${System.getProperty("user.dir")}/dataExample/
              test_Keys.txt",
13         "${System.getProperty("user.dir")}/dataExample/
              test_Reactions.txt",
14         KeysBbParser(), ReactionsBbParser(), BbAnalyzer(
              KeysConverter())
15     )
16
17     println(fileAnalyzer.analyzeByFile("${System.getProperty("
           user.dir")}/dataExample/test_Keys.txt"))
18 }
```

ПРИЛОЖЕНИЕ Б

Исходный код серверной части

Листинг 29: Реализация класса InluxServiceServer

```
        startKoin {
            modules(module {
                single { InfluxdbConfiguration() }

                single { CharRepositoryImpl() }

                single { DataService() }

                single { DataController() }
            })
        }
    }

    private val serverSocket = ServerSocket(socketPort)

    fun run() {
        getRuntime().addShutdownHook(Thread {
            println("Server on port ${serverSocket.localPort}
               stopped")
        })

        if (!serverSocket.isBound || serverSocket.isClosed) {
            throw SocketException("Server socket is already in
               use")
        }

        println("Server started on port ${serverSocket.
           localPort}")

        while (true) {
            val clientSocket = serverSocket.accept()
            thread {
                InfluxServiceClientHandler(clientSocket).run()
            }
        }
```

```
33     }
34 }
```

Листинг 30: Реализация класса InfluxServiceClientHadler

```
 1 class InfluxServiceClientHandler(private val clientSocket:
     Socket) {
 2     private val ydvpVersion = "0.1"
 3     private val defaultHeader = YdvpHeader("Server", "127.0.0.1
         ")
 4
 5     private val controller by inject<DataController>(
         DataController::class.java)
 6
 7     private fun anyResponse(code: String, explanation: String,
         body: Any): YDVP {
 8         return YDVP(
 9             YdvpStartingLineResponse(
10                 ydvpVersion,
11                 code,
12                 explanation
13             ),
14             listOf(defaultHeader),
15             GsonObject.gson.toJson(body)
16         )
17     }
18
19     private val badRequestResponse by lazy {
20         YDVP(
21             YdvpStartingLineResponse(ydvpVersion, "400", "BAD
                 REQUEST"),
22             listOf(defaultHeader)
23         )
24     }
25     private val internalServerErrorResponse by lazy {
26         YDVP(
27             YdvpStartingLineResponse(ydvpVersion, "500", "
                 INTERNAL SERVER ERROR"),
28             listOf(defaultHeader)
29         )
30     }
31
```

```
32      private val methodNotAllowed by lazy {
33          YDVP(
34              YdvpStartingLineResponse(ydvpVersion, "405", "
                    METHOD NOT ALLOWED"),
35              listOf(defaultHeader)
36          )
37      }
38
39      private val notFoundResponse by lazy {
40          YDVP(
41              YdvpStartingLineResponse(ydvpVersion, "404", "NOT
                    FOUND"),
42              listOf(defaultHeader)
43          )
44      }
45
46      private fun prepareUri(uri: String): List<String> {
47          val parsedUri = uri.split("/").toMutableList()
48
49          if (parsedUri[0] != "")
50              throw UnsupportedUriException("URI format error")
51
52          parsedUri.removeAt(0)
53          return parsedUri
54      }
55
56      private fun controllerPostMethod(uri: String, body: String)
            : YDVP {
57          val parsedUri = prepareUri(uri)
58
59          return when (parsedUri.first()) {
60              "data" -> {
61                  if (parsedUri.size < 2)
62                      throw UnsupportedUriException("Not enough
                            inline arguments")
63                  val response = controller.addData(
64                      parsedUri[1],
65                      GsonObject.gson.fromJson(body,
                            AcceptMeasurementsListDTO::class.java)
66                  )
67
```

```
68                anyResponse(response.statusCodeValue.toString()
                      , response.statusCode.name, response.body)
69            }
70            else -> throw UnsupportedUriException("Unsupported
                  URI")
71        }
72    }
73
74    private fun controllerGetMethod(uri: String, body: String):
          YDVP {
75        val parsedUri = prepareUri(uri)
76
77        return when (parsedUri.first()) {
78            "data" -> {
79                if (parsedUri.size < 2)
80                    throw UnsupportedUriException("Not enough
                          inline arguments")
81                val response =
82                    controller.getData(parsedUri[1], GsonObject
                          .gson.fromJson(body, Array<String>::
                          class.java).toList())
83
84                anyResponse(response.statusCodeValue.toString()
                      , response.statusCode.name, response.body)
85            }
86            else -> throw UnsupportedUriException("Way not
                  found")
87        }
88    }
89
90    private fun controllerWayByMethod(ydvpRequest: YDVP): YDVP
          {
91        ydvpRequest.startingLine as YdvpStartingLineRequest
92        val method = ydvpRequest.startingLine.method
93        val uri = ydvpRequest.startingLine.uri
94
95        val body = ydvpRequest.body
96
97        return try {
98            when (method) {
99                "GET" -> controllerGetMethod(uri, body)
```

```
100                 "POST" -> controllerPostMethod(uri, body)
101                 else -> methodNotAllowed
102             }
103         } catch (exc: NullPointerException) {
104             badRequestResponse
105         } catch (exc: UnsupportedUriException) {
106             notFoundResponse
107         } catch (exc: Exception) {
108             println("EXCEPTION")
109             println(exc.javaClass)
110             println(exc.localizedMessage)
111             internalServerErrorResponse
112         }
113     }
114
115     fun run() {
116         println("Accepted client on ${clientSocket.
                localSocketAddress} from ${clientSocket.
                remoteSocketAddress}")
117
118         val bufferedReader = BufferedReader(InputStreamReader(
                clientSocket.getInputStream()))
119
120         var gotRequest = bufferedReader.readLine() + "\n"
121         while (bufferedReader.ready())
122             gotRequest += bufferedReader.readLine() + "\n"
123
124         val clientOut = PrintWriter(clientSocket.
                getOutputStream(), true)
125
126         val ydvpRequest = try {
127             YdvpParser().parseRequest(gotRequest)
128         } catch (exc: Exception) {
129             clientOut.println(badRequestResponse)
130
131             return
132         }
133
134         val response = controllerWayByMethod(ydvpRequest).
                createStringResponse()
135         println("Returned to client:\n$response")
```

```
136         clientOut.println(response)
137     }
138 }
139
140 class InfluxServiceServer(socketPort: Int) {
141     init {
```

Листинг 31: Пример организации серверной части

```
 1 package examples.helloexample
 2
 3 import server.InfluxServiceServer
 4
 5 fun main() {
 6     val server = InfluxServiceServer(6666)
 7
 8     server.run()
 9 }
```

Листинг 32: Пример организации клиентской части

```
 1 package examples.helloexample
 2
 3 import client.InfluxServiceClient
 4 import domain.dtos.AcceptMeasurementsDTO
 5 import domain.dtos.AcceptMeasurementsListDTO
 6 import domain.dtos.MeasurementDataWithoutTime
 7 import domain.dtos.ResponseMeasurementsDTO
 8 import gson.GsonObject
 9 import protocol.YDVP
10 import protocol.YdvpHeader
11 import protocol.YdvpStartingLineRequest
12 import protocol.YdvpStartingLineResponse
13 import java.net.ConnectException
14
15 val measurementsToSend = AcceptMeasurementsListDTO(
16     listOf(
17         AcceptMeasurementsDTO(
18             "pulse", listOf(
19                 MeasurementDataWithoutTime("30"),
20                 MeasurementDataWithoutTime("40")
21             )
22         ),
```

```
23          AcceptMeasurementsDTO(
24              "botArterialPressure", listOf(
25                  MeasurementDataWithoutTime("40"),
26                  MeasurementDataWithoutTime("50")
27              )
28          ),
29          AcceptMeasurementsDTO(
30              "topArterialPressure", listOf(
31                  MeasurementDataWithoutTime("80"),
32                  MeasurementDataWithoutTime("90")
33              )
34          )
35      )
36  )
37
38  fun sendTest() {
39      val client = InfluxServiceClient("localhost", 6666)
40
41      client.use {
42          try {
43              client.connect()
44          } catch (exc: ConnectException) {
45              println("Server is dead")
46              return
47          }
48
49          client.sendRequestAndGetResponse(
50              YDVP(
51                  YdvpStartingLineRequest("POST", "/data/TestUser
                        ", "0.1"),
52                  listOf(YdvpHeader("Host", "127.0.0.1")),
53                  GsonObject.gson.toJson(measurementsToSend)
54              )
55          )
56      }
57  }
58
59  fun getTest() {
60      val client = InfluxServiceClient("localhost", 6666)
61
62      client.use {
```

```
63          try {
64              client.connect()
65          } catch (exc: ConnectException) {
66              println("Server is dead")
67              return
68          }
69
70          client.sendRequestAndGetResponse(
71              YDVP(
72                  YdvpStartingLineRequest("GET", "/data/TestUser"
                        , "0.1"),
73                  listOf(YdvpHeader("Host", "127.0.0.1")),
74                  GsonObject.gson.toJson(listOf("pulse", "
                        botArterialPressure"))
75              )
76          )
77      }
78 }
79
80 fun main() {
81      sendTest()
82 //     getTest()
83 }
```

ПРИЛОЖЕНИЕ В

Примеры полученных от пользователя данных

Листинг 33: Первые 40 строк данных анализа действий с клавиатурой

```
 1 key=G, timestamp=1652607722497
 2 key=T, timestamp=1652607722677
 3 key=H, timestamp=1652607722806
 4 key=D, timestamp=1652607722933
 5 key=F, timestamp=1652607723101
 6 key=Z, timestamp=1652607723242
 7 key=Slash, timestamp=1652607723511
 8 key=Space, timestamp=1652607723627
 9 key=F, timestamp=1652607723781
10 key=D, timestamp=1652607724434
11 key=F, timestamp=1652607724589
12 key=Y, timestamp=1652607724757
13 key=U, timestamp=1652607725089
14 key=F, timestamp=1652607725218
15 key=H, timestamp=1652607725448
16 key=L, timestamp=1652607725629
17 key=Slash, timestamp=1652607725834
18 key=Space, timestamp=1652607725925
19 key=E, timestamp=1652607726066
20 key=Semicolon, timestamp=1652607726156
21 key=T, timestamp=1652607726260
22 key=Space, timestamp=1652607726361
23 key=D, timestamp=1652607726631
24 key=S, timestamp=1652607726734
25 key=H, timestamp=1652607726876
26 key=B, timestamp=1652607727017
27 key=C, timestamp=1652607727094
28 key=J, timestamp=1652607727210
29 key=D, timestamp=1652607727326
30 key=S, timestamp=1652607727466
31 key=D, timestamp=1652607727554
32 key=F, timestamp=1652607727747
33 key=K, timestamp=1652607727888
34 key=F, timestamp=1652607727978
35 key=C, timestamp=1652607728131
```

```
36 key=M, timestamp=1652607728209
37 key=Space, timestamp=1652607728310
38 key=Y, timestamp=1652607728477
39 key=F, timestamp=1652607728541
40 key=Space, timestamp=1652607728632
```

Листинг 34: Первые 40 строк данных анализа действий с мышью

```
 1 x=994, y=570, key=1, timestamp=1652607697196
 2 x=993, y=618, key=1, timestamp=1652607698841
 3 x=884, y=553, key=1, timestamp=1652607700152
 4 x=1005, y=503, key=1, timestamp=1652607707841
 5 x=1150, y=454, key=1, timestamp=1652607708860
 6 x=2848, y=1, key=1, timestamp=1652607710663
 7 x=3165, y=129, key=1, timestamp=1652607712493
 8 x=3063, y=478, key=1, timestamp=1652607714309
 9 x=1990, y=692, key=1, timestamp=1652607717712
10 x=2665, y=596, key=1, timestamp=1652607718976
11 x=2834, y=294, key=1, timestamp=1652607744520
12 x=2094, y=198, key=1, timestamp=1652607745173
13 x=3076, y=472, key=1, timestamp=1652607746519
14 x=1980, y=676, key=1, timestamp=1652607756266
15 x=2540, y=607, key=1, timestamp=1652607761777
16 x=2789, y=392, key=1, timestamp=1652607767999
17 x=2511, y=379, key=1, timestamp=1652607775627
18 x=2772, y=475, key=1, timestamp=1652607836448
19 x=870, y=340, key=1, timestamp=1652607842495
20 x=13, y=549, key=1, timestamp=1652607845704
21 x=2658, y=483, key=1, timestamp=1652607848005
22 x=2744, y=586, key=1, timestamp=1652607905944
23 x=2710, y=604, key=1, timestamp=1652607907428
24 x=2657, y=425, key=1, timestamp=1652607908124
25 x=2094, y=288, key=1, timestamp=1652607914433
26 x=3696, y=461, key=1, timestamp=1652607918059
27 x=2142, y=431, key=1, timestamp=1652607919348
28 x=3388, y=546, key=1, timestamp=1652607920515
29 x=3381, y=563, key=1, timestamp=1652607921828
30 x=3575, y=444, key=1, timestamp=1652607923673
31 x=3575, y=444, key=1, timestamp=1652607925103
32 x=3385, y=741, key=1, timestamp=1652607927127
33 x=2656, y=417, key=1, timestamp=1652607928988
34 x=2642, y=593, key=1, timestamp=1652607929821
```

```
35 x=2116, y=117, key=1, timestamp=1652607991163
36 x=1950, y=51, key=1, timestamp=1652607994926
37 x=2004, y=268, key=1, timestamp=1652608003053
38 x=3724, y=463, key=1, timestamp=1652608011495
39 x=893, y=572, key=1, timestamp=1652608014592
40 x=2627, y=520, key=1, timestamp=1652608016682
```