2018 deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática ieeta instituto de engenharia electrónica e telemática de aveiro

Alex Miranda dos Santos
Andreia de Castro Ferreira
João Renato Pinto Limas
Pedro Alexandre Santos Fernandes

SENSITIVE SPACES

Relatório de Projeto em Informática, no âmbito da Licenciatura em Engenharia Informática, da Universidade de Aveiro, sob orientação do docente José Maria Amaral Fernandes e coorientação do docente Ilídio Fernando de Castro Oliveira.

Agradecimentos

Gostaríamos de agradecer ao nosso orientador de projeto, José Maria Amaral Fernandes, ao nosso coorientador, Ilídio Fernando de Castro Oliveira e ao docente José Moreira, regente da unidade curricular de Projeto em Informática do corrente ano letivo, toda a disponibilidade que sempre demonstraram ao longo deste semestre e sempre se mostraram dispostos a ajudar sempre que foi necessário para que fosse possível derrubar os obstáculos que foram surgindo no desenvolvimento do projeto. De salientar também todos os nossos colegas e restantes professores que direta ou indiretamente contribuíram para a realização deste projeto.

Resumo

O projeto *Sensitive Spaces* visa avaliar a prestação de um utilizador através de *input* de teclado, *webcam*, áudio e pulsação e fornecer *feedback* com o intuito de melhorar a sua produtividade. A integração prevê uma utilização futura com equipas com capacidade de *cloud computing*.

Esta avaliação é feita através de sensores apropriados e pretende quantificar o uso intensivo das teclas do computador, bem como, a sua prestação e quantidade de erros obtidos durante a compilação, informando o utilizador quando deve fazer uma pausa e descansar.

Palavras-chave:

Cloud computing;

Container;

Docker;

Elasticsearch;

ELK;

Eye Tracker;

GUI;

Kafka;

KeyLogger;

Kibana;

Logstash;

Native Key Listener;

Noise Sensor;

Pulse Sensor;

Raspberry;

Sensores;

Tempo real;

Webcam.

Índice

Índice de Figuras

Abreviações

BPM	Batimentos por minuto
dB	Decibel
DETI	Departamento de Eletrónica Telecomunicações e Informática
ELK	Elasticsearch, Logstash, Kibana
GUI	Graphical User Interface
IDE	Integrated Development Environment
IEETA	Institute of Electronics and Informatics Engineering of Aveiro
I/O	Input/Output
IP	Internet Protocol
JSON	JavaScript Object Notation
LEI	Licenciatura em Engenharia Informática
TCP	Transmission Control Protocol
UA	Universidade de Aveiro
UI	User Interface
SPI	Serial Peripheral Interface

Enquadramento

Este projeto foi proposto aos discentes no ano letivo 2017/2018, como parte integrante da avaliação da unidade curricular de Projeto em Informática, inserido no terceiro ano da Licenciatura em Engenharia Informática, da Universidade de Aveiro.

O projeto *Sensitive Spaces* visa avaliar a prestação de um utilizador através de *input* de teclado, *webcam*, áudio e pulsação e fornecer *feedback* com o intuito de melhorar a sua produtividade. A integração prevê uma utilização futura com equipas com capacidade de *cloud computing*.

Esta avaliação é feita através de sensores apropriados e pretende quantificar o uso intensivo das teclas do computador, bem como, a sua prestação e quantidade de erros obtidos durante a compilação, informando o utilizador quando deve fazer uma pausa e descansar.

O projeto foi divulgado no Students@DETI e a experiência contou com cenário que envolvia uma secretária e um computador com uma webcam e micro incorporados, permitindo que o utilizador se sentisse confortável. Havia também um sensor que era aplicado num dedo de forma a medir os batimentos cardíacos. Todos os dados registados são transmitidos em tempo real para um *dashboard*.

Capítulo 1

Introdução

Este relatório contará com cinco capítulos, que se encontram divididos adequadamente de modo a permitir uma melhor perceção do processo de desenvolvimento.

O primeiro capítulo começa por contextualizar, fazendo uma breve introdução ao funcionamento do projeto. Depois, no segundo capítulo é descrito um estudo sobre o Estado da Arte de projetos concorrentes/semelhantes à solução desenvolvida, bem como as tecnologias que foram utilizadas. Estas serão explicadas de forma detalhada, expondo noções e conhecimento necessários para a compreensão deste projeto. No terceiro capítulo, é descrito o processo de levantamento dos requisitos, quer funcionais, quer não funcionais, tendo como objetivo a modelação realizada para o funcionamento e gestão do sistema. Por conseguinte, será descrita a arquitetura implementada e explicados os diagramas representativos da análise de modelação. No quarto capítulo, encontra-se explicitado a implementação efetuada no projeto, tanto a nível de *software* como de *hardware*. As configurações necessárias para o funcionamento do sistema também serão esclarecidas detalhadamente, expondo também as componentes de *software* desenvolvidas pelos elementos do grupo, para a sua utilização e visualização por parte do utilizador e observador.

Motivação

O DETI, em conjunto com o IEETA, tem vindo a desenvolver imensos projetos inovadores. Para este em específico, existe a possibilidade de colaboração com o Departamento de Educação e Psicologia, também da Universidade de Aveiro, onde o curso de Psicologia irá participar na análise dos resultados obtidos na experiência.

Contextualização

Este projeto surgiu com o intuito de aplicar o crescente uso de tecnologias, fornecendo um melhor ambiente ao utilizador alertando-o para que não cometa erros progressivos, através de um *feedback* do seu estado psicológico, que é lido através das teclas que o programador pressiona durante a experiência, informando-o quando deve fazer uma pausa. Para este caso, o utilizador, recebe uma mensagem de alerta que o informa quando deve parar para descansar. Os dados registados são enviados em tempo real para um *dashboard*.

Figura 1 – Logótipo *Sensitive Spaces*

Capítulo 2

Neste capítulo será feita uma abordagem do estado da arte e das tecnologias usadas.

Estado da Arte

Eye Tracking [1] é o processo de medição da posição dos olhos ou o movimento destes em relação à cabeça. O *Eye Tracker* é o dispositivo que executa este processo. São usados para o estudo de sistemas visuais, principalmente a nível de psicologia e de *marketing*. Existem vários tipos de medição do movimento do olho. Um tipo de *eye tracking* é uma espécie de lente de contacto que, colocado no olho deteta o movimento do olho, dando assim a sua posição. O segundo tipo utiliza potências elétricas medidas por um tipo de elétrodos [2] em redor dos olhos. Os olhos possuem um campo potencial elétrico constante que é detetado pelos elétrodos, dando assim a posição dos olhos. O terceiro tipo consiste em reflexão de luz. A luz, normalmente infravermelha é refletida pelo olho, que é depois captada por uma câmara de vídeo. Esta solução normalmente usa o reflexo corneano e o centro da pupila como recursos para rastrear o movimento ao longo do tempo.

Para este projeto adaptou-se a solução do terceiro tipo por ser a menos evasiva para o utilizador. Consiste numa *webcam* que capta os olhos do utilizador. Tornou-se a solução ideal para este projeto, pois permite ver se o utilizador se encontra em frente ao computador e se este se encontra focado na sua tarefa.

A solução deste tipo de implementação é gratuita e garante grande compatibilidade entre os equipamentos. No entanto, é necessário haver um fundo branco nas costas do utilizador, para que não ocorram erros na captação dos olhos.

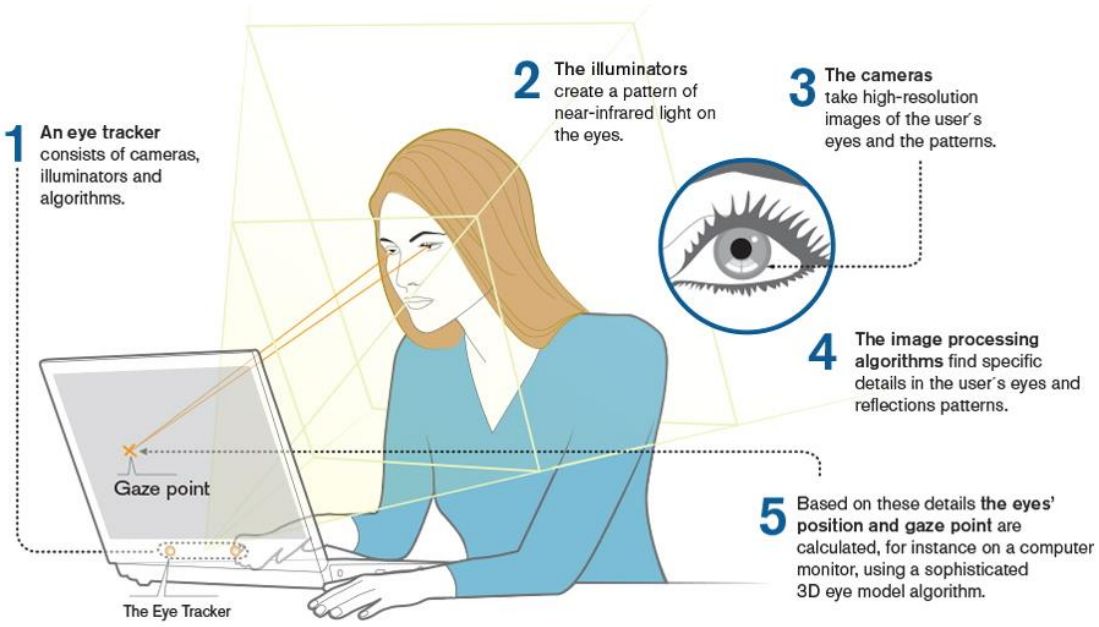

Figura 2 – Ilustração do *Eyetracker*

Um *Keylogger* [3] é um *software* instalado no computador cujo objetivo é fazer um registo de tudo o que é pressionado no teclado. Este tipo de software é utilizado principalmente nas empresas para que seja possível monitorizar o que os funcionários fazem nas suas próprias máquinas. No entanto, este software pode ser utilizado com outro propósito, nomeadamente, gravar senhas, números de cartão e entre outros tipos de fraude. Este tipo de software além de ser gratuito, é de fácil implementação e apresenta todos os dados necessários.

Figura 3 – Ilustração do *Keylogger*

Com o objetivo de avaliar a intensidade e a frequência das teclas utilizadas pelo utilizador fez-se a implementação de um *keylogger* usando um *KeyListener* [4]. Caso se verifique que o backspace tenha sido utilizado frequentemente, representa que o programador tenha cometido vários erros e que os esteja a apagar sucessivamente.

O *Pulse Sensor* é um sensor de frequência cardíaca *plug-and-play*. É um dispositivo de monitorização pessoal não invasivo que permite medir a frequência cardíaca em tempo real. Dispositivos deste género têm sido utilizados frequentemente no dia-a-dia das pessoas, como por exemplo, nos *smartwatches*, *running shoes entre outros*. Não necessita de conhecimento técnico para poder ser utilizado. É uma solução barata e simples para projetos que necessitem de incorporar facilmente dados de frequência cardíaca ao vivo.

Figura 4 – Ilustração do Pulse Sensor

No âmbito do projeto *Sensitive Spaces* é utilizado para monitorizar o utilizador e correlacionar com outros dados a fim de estabelecer uma linha de base e verificar existência algumas variações significativas.

O *Noise Sensor* é um sensor de decibéis captados através de um microfone. Através deste sensor é possível determinar o ruído do meio ambiente em que a medição é feita.

Neste projeto, este sensor é útil para determinar se o utilizador se encontra num ambiente com muito ruído e se esse fator afetar ou não a sua *performance*.

Tecnologias utilizadas

As tecnologias utilizadas para desenvolver o projeto não foram lecionadas durante as temáticas da LEI, pelo que, foram exploradas no decorrer do projeto.

Docker

O *Docker* [5] é uma tecnologia de virtualização ao nível do sistema operativo que possibilita o lançamento das aplicações de forma agnóstica e autocontida, isto é, é independente do sistema operativo que ocorre inerentemente e inclui todas as dependências necessárias.

Designamos de instâncias autocontida os *containers*, sendo que os *containers* apenas precisam de do *Docker Engine* [6] para poderem ser utilizados.

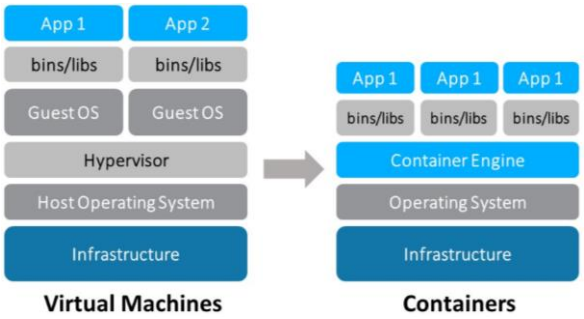

Figura 5 – Arquitetura dos *containers* do *Docker* [7]

Optou-se por usar o *docker* por ser altamente eficiente e ter uma performance confiável e porque os containers do *Docker* são mais leves e mais rápidos. Este tipo de tecnologia é produtivo por ser facilmente integrável em plataformas de integração contínua. É compatível com a maior parte das distribuições para *Linux*, *MacOS* e *Windows*.
O *container* é iniciado executando uma imagem, sendo que esta imagem faz parte de um pacote executável que inclui tudo o que é necessário para executar uma aplicação, ou seja, funciona como uma instância de tempo de execução.
Os mecanismos de volumes foram usados para a garantia de persistência dos dados que foram gerados através do container *docker*. Estes mecanismos são ideais para efetuar cópias de segurança e partilhar os dados entre os containers.
Para a criação de multi-containers existe uma maneira mais simples para a gestão dos containers, fazendo uso do Docker compose [8], que foi instalado através da distribuição GNU Linux.

Figura 6 – Logo do *Docker Compose* [9]

Kafka

O *Apache Kafka* [10] é uma plataforma de transmissão de dados distribuída, semelhante a uma fila de mensagens, cujo o seu principal objetivo é providenciar um fluxo de dados em tempo real. O *kafka* suporta a entrega de mensagens de baixa latência e oferece a garantia de tolerância a falhas na presença de falhas na máquina. Tem a capacidade de lidar com um grande número de diversos *consumers*. É uma tecnologia eficiente e rápida a garante a persistência de todos os dados gravados no disco, ou seja, significa que as gravações são direcionadas para a cache da RAM. No *kafka*, a comunicação entre os clientes e os servidores é feita com um protocolo TCP agnóstico em linguagem simples, de alto desempenho.

Os *producers* [11] enviam os dados para o broker do *kafka*, isto é, cada vez que um *producer* enviar uma mensagem para um broker, este broker simplesmente irá anexar a na fila de mensagens como se fosse anexada a uma partição. Os *consumers* [12] leem os dados do broker e o lo*gstash* [13] codificará os eventos adicionando um *timestamp* e um *hostname*, como será descrito a seguir.

ELK

ELK [14] é o acrónimo que junta três mecanismos de *open-source*: *Elasticsearch*, *Logstash* e *Kibana* e fornece um servidor de log centralizado conveniente.

O *Elasticsearch* é altamente escalável e fornece uma interface *RESTfull* de pesquisa e análise de dados. É capaz de solucionar bastantes use cases, permitindo armazenar, pesquisar e analisar grandes volumes de dados rapidamente e em tempo quase real. Dentro da stack o elasticsearch serve de base de dados, onde podemos guardar e indexar os dados.

O *Logstash* é um *pipeline* de processamento de dados do lado do servidor que insere os dados de várias fontes simultaneamente, os transforma e envia para um "*stash*", como o *elasticsearch*. É o motor central do fluxo de dados do *ELK* para conseguir unificar os dados independentemente do seu formato. Este processamento em tempo real é especialmente eficiente quando associado ao *elasticsearch* e ao Kibana. O *logstash* recebe os dados e mantém esses dados no *elasticsearch*. Assim que os dados estejam no *elasticsearch*, já é possível analisá-los.

O *Kibana* permite que os utilizadores pesquisem, visualizem e interajam com os dados em tempo real com gráficos e tabelas no *elasticsearch*. Este mecanismo é uma plataforma de análise e visualização dos dados, também projetada para trabalhar com o *elasticsearch*. O *kibana* facilita a compreensão de grandes volumes de dados.

Portainer.io

O *Portainer* é um gestor simples e leve que de uma forma direta, através de um *GUI*, permite a gestão de vários ambientes do *Docker Portainer*. Desta forma, é possível gerir *containers*, imagens, volumes, redes e ainda aceder facilmente aos *logs* e consola dos *containers*.

Capítulo 3

Requisitos do Sistema e Modelação

Nesta secção é descrito o processo de levantamento dos requisitos, quer funcionais, quer não funcionais, tendo como objetivo a modelação realizada para o funcionamento e gestão do sistema.

No desenvolvimento do projeto, foram usadas as linguagens de *Java* para as partes do *keylogger* e do *GUI* e *Python* para o *pulse sensor*, *noise sensor* e *eye tracker*.

Requisitos funcionais

Todo o sistema preparado antecipadamente e adequadamente é capaz de monitorizar o utilizador que participar na experiência. Toda esta experiência será realizada numa sala do DETI, tendo como cenário uma secretária, um computador e uma *webcam*, de modo a permitir que o utilizador fique confortável, como se demonstra na figura seguinte.

Figura 7 – Ilustração ambiente do utilizador

O utilizador senta-se em frente ao computador que lhe é disponibilizado pela equipa responsável. Terá também acesso a um pulse sensor para poder registar o seu batimento cardíaco. Imediatamente, o *raspberry* lê os dados e envia em tempo real para a plataforma *kibana*. Todo este processo terá partido de sensores existentes, como um *raspberry pi*, uma webcam, um pulse sensor e um micro, onde a partir daqui são testadas as mais diversas reações do utilizador.

Requisitos não funcionais

A persistência dos dados será no elasticsearch e estes serão apresentados imediatamente no *kibana*.

Atores

Para este projeto temos dois atores principais: o cliente e o observador.

O cliente representa o programador, ou seja, o utilizador que irá participar na experiência. Este senta-se em frente ao computador, conecta-se ao *raspberry* e começa a programar. A partir daqui o pulse sensor começa a registar as diversas reações.

Do outro lado, temos o observador que será a equipa de psicologia responsável por receber e analisar os dados obtidos na experiência.

Casos de Uso

As próximas duas figuras demonstram os casos de usos dos dois tipos de entidade representadas no sistema.

Client: Utilizador que fará a experiência

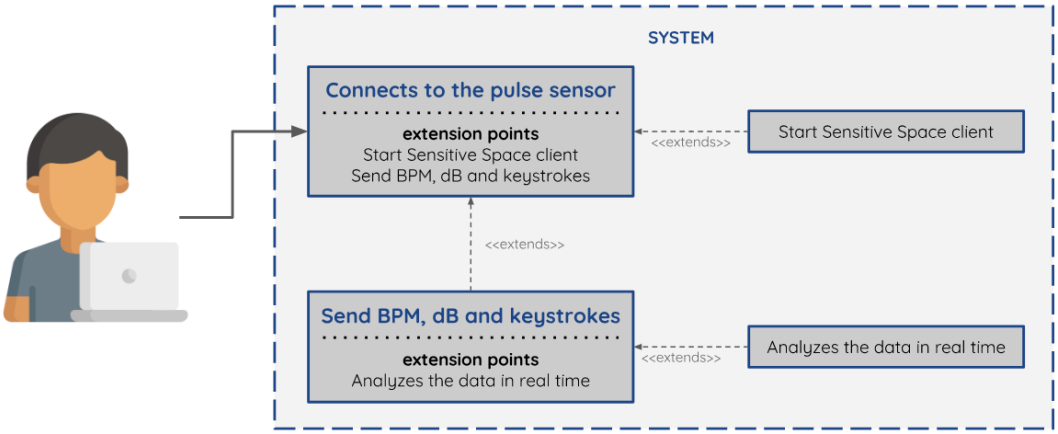

Figura 8 – Caso de uso do utilizador

Observer: Analisa os dados

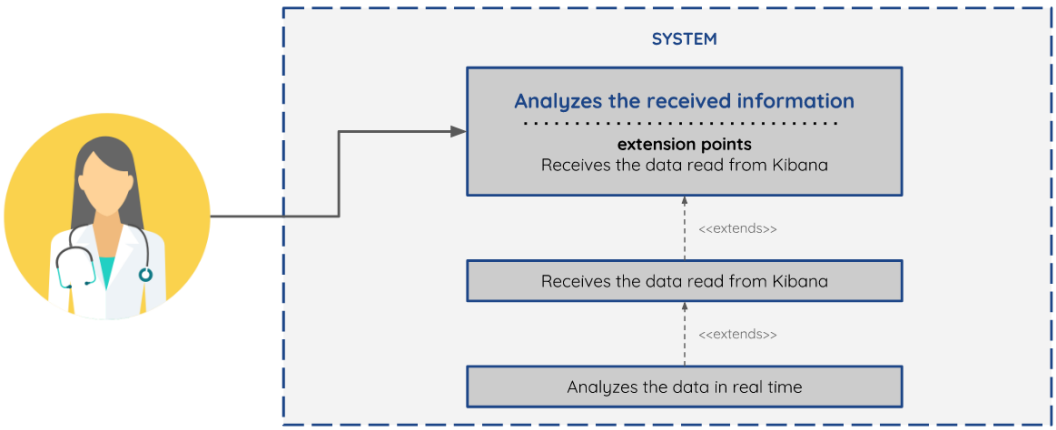

Figura 9 – Caso de uso do observador

Arquitetura do Sistema

O Capítulo 3 foca-se essencialmente na arquitetura detalhada do sistema implementado e respetivos diagramas.

Diagrama de Arquitetura

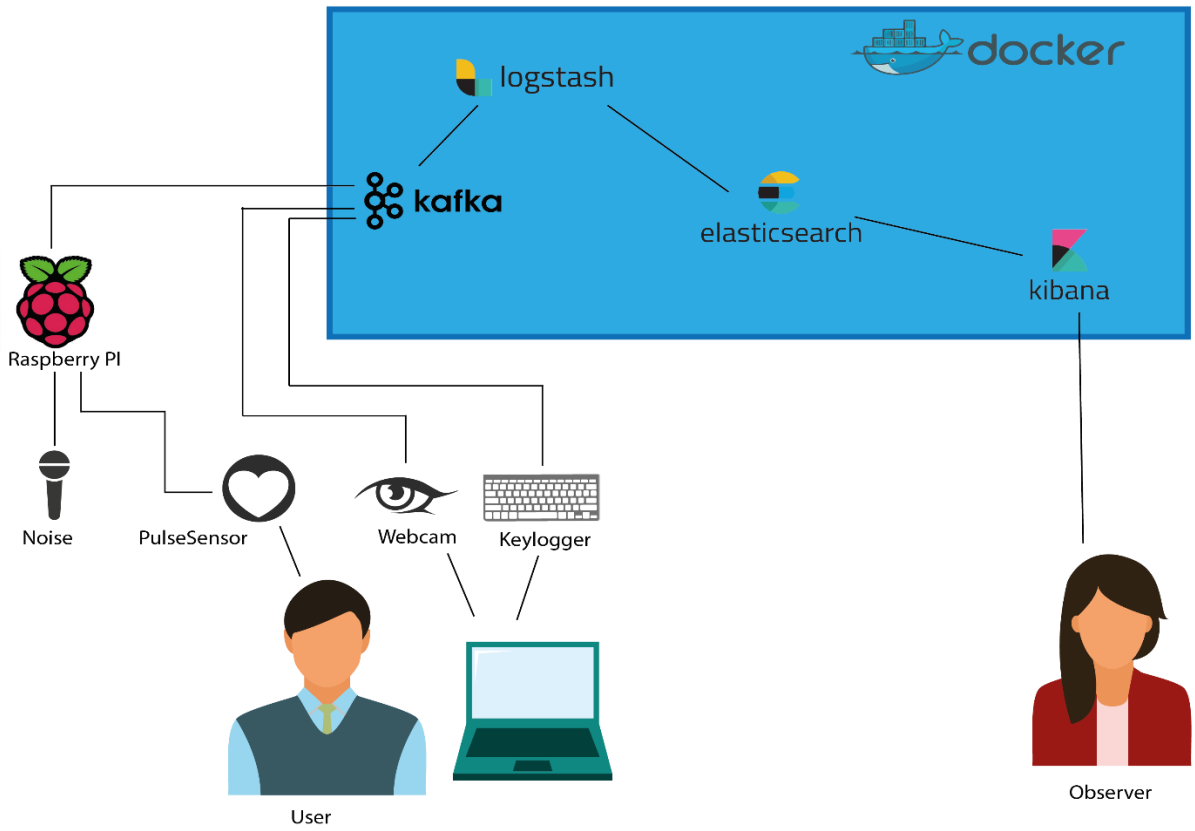

Figura 10 – Arquitetura física do sistema

Diagrama de Implementação

No diagrama seguinte obtém-se uma visão geral das tecnologias usadas dentro do sistema.

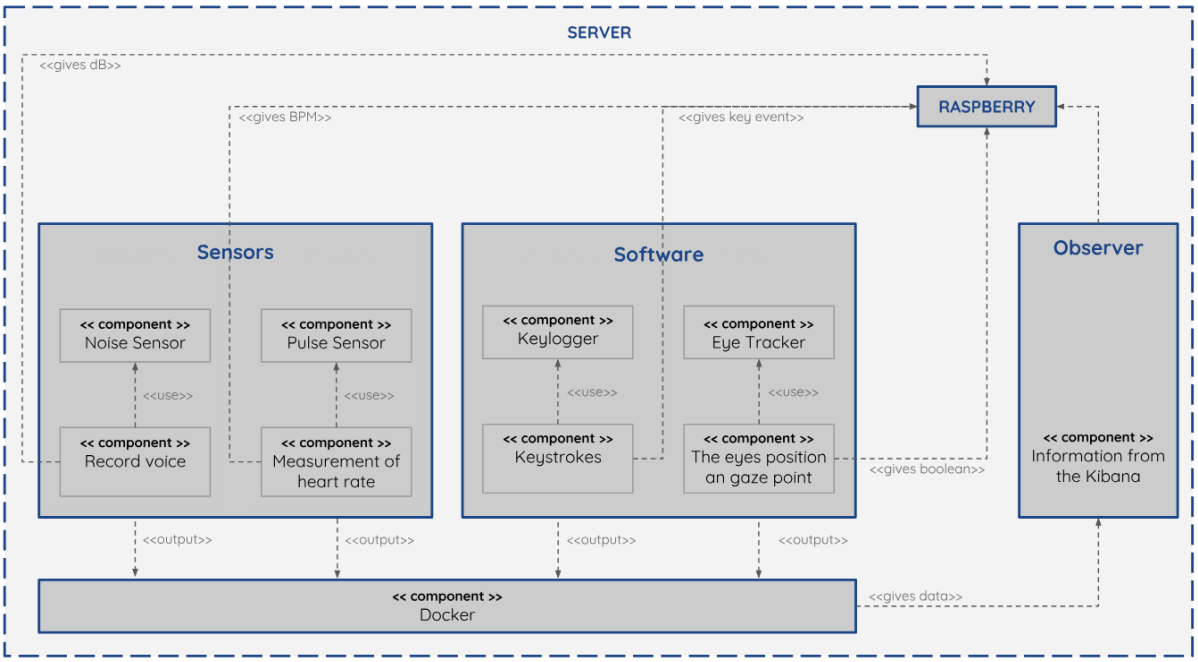

Figura 11 – Diagrama de implementação

Diagrama de Domínio

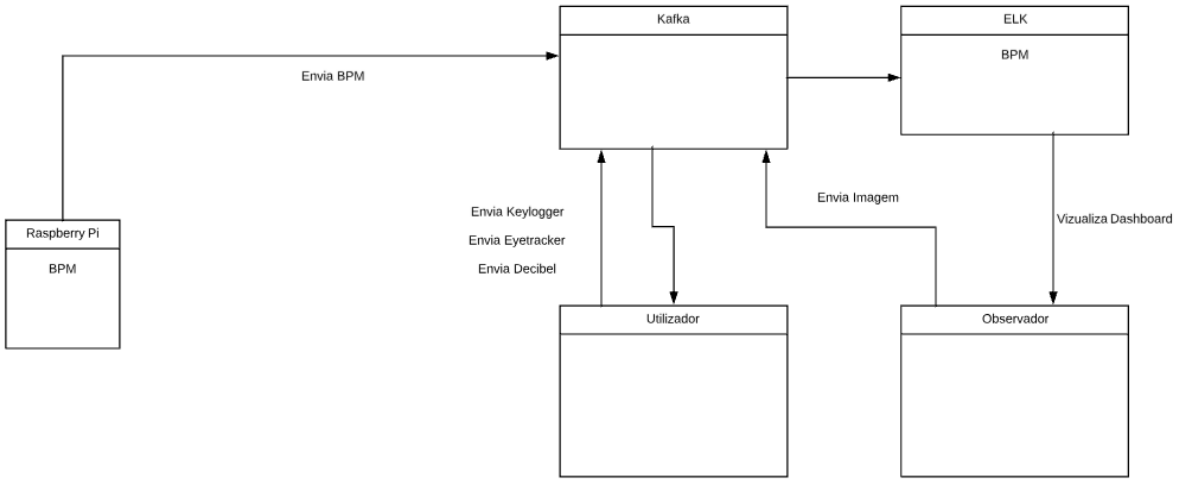

Figura 12 – Diagrama de domínio

Capítulo 4

Infraestrutura de rede

Infraestrutura baseada na utilização de *docker* para um *cloud deployment* futuro. A *subnetwork* de serviços *kibana* e *ELK* são geridas pelo *docker* na gama de *IP's* 172.0.0.0 e emulada internamente.

192.168.2.100 – Servidor

192.168.2.199 – *Raspberry pi + pulse sensor*

192.168.2.127 – Máquina do cliente: k*eylogger* e *eye tracker*

192.168.2.198 – Máquina do Observador

Pulse sensor

A configuração do *Pulse Sensor* é divida em duas categorias:

Software

O *software* é uma adaptação do projeto *Raspberry-Pi-Heartbeat-Pulse-Sensor* [15]. No *driver* do MCP3008 devemos definir para escuta o canal 0. Para a utilização de múltiplos *pulse sensor* (no máximo 8) devemos alterar esta função para escutar os canais utilizados.

```
def read(self, channel = 0):
```

Figura 13 – Configuração MCP3008

Para criar um *producer* de mensagens do tipo *JSON* para o *Kafka* utilizamos a biblioteca *JSON* e *PyKafka* com o seguinte código:

```
import json
from pykafka import KafkaClient

bootstrap_servers = '192.168.2.125:9092'


def pykafka_producer():
    client = KafkaClient(hosts=bootstrap_servers)
    topic = client.topics[b'topic']
    with topic.get_producer() as producer:
```

Figura 14 – Configuração do *Kafka Producer*

Hardware

O *pulse sensor* é um sensor analógico e o *raspberry Pi* não dispõem de portas SPI analógicas, apenas digitais pelo que é necessário incluir um chip adc (*analog digital converter*) no nosso projeto foi escolhido o MCP3008 [16] de 8bits.

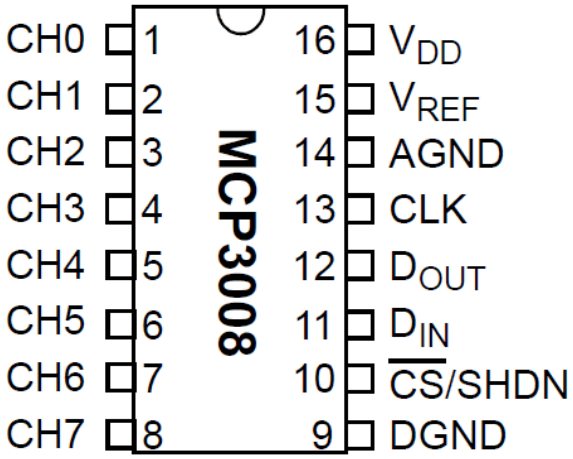

Figura 15 – Chip ADC MCP3008 [17]

Ligação do MCP3008 ao *Raspberry*:

De MCP3008 VDD para *Raspberry* Pi 3.3V

De MCP3008 VREF para *Raspberry* Pi 3.3V

De MCP3008 AGND para *Raspberry* Pi GND

De MCP3008 DGND para *Raspberry* Pi GND

De MCP3008 CLK para *Raspberry* Pi pin 18

De MCP3008 DOUT para *Raspberry* Pi pin 23

De MCP3008 DIN para *Raspberry* Pi pin 24

De MCP3008 CS/SHDN para *Raspberry* Pi pin 25

De MCP3008 CH0 para *Pulsesensor* pin *Signal*

Portainer.io

Para executar o portainer é necessário na consola executar o seguinte comando: docker run -d -p 9000:9000 --name portainer --restart always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer

Para proceder ao *deployment* de *Kafka*, *Zookeeper* e *ELK* de forma mais simples e eficiente optamos por lançar via *docker compose* dentro da mesma rede (com o nome ***elkk***).

Serviços

Elasticsearch

```
elasticsearch:
  build:
    context: elasticsearch/
  container_name: elasticsearch
  volumes:
    - ./elasticsearch/config/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml:ro
  ports:
    - "9200:9200"
    - "9300:9300"
  environment:
    ES_JAVA_OPTS: "-Xmx256m -Xms256m"
  networks:
    - elkk
```

Figura 16 – Configuração do *Elasticsearch*

Logstash

```
logstash:
  build:
    context: logstash/
  container_name: logstash
  volumes:
    - ./logstash/config/logstash.yml:/usr/share/logstash/config/logstash.yml:ro
    - ./logstash/pipeline:/usr/share/logstash/pipeline:ro
  ports:
    - "5000:5000"
  environment:
    LS_JAVA_OPTS: "-Xmx256m -Xms256m"
  depends_on:
    - elasticsearch
    - kafka1
  networks:
    - elkk
```

Figura 17 – Configuração do *Logstash*

Kibana

```
kibana:
  build:
    context: kibana/
  container_name: kibana
  volumes:
    - ./kibana/config/:/usr/share/kibana/config:ro
  ports:
    - "5601:5601"
  depends_on:
    - elasticsearch
  networks:
    - elkk
```

Figura 18 – Configuração do *Kibana*

Zookeeper

```
zoo1:
  image: elevy/zookeeper:latest
  container_name: zookeeper
  environment:
    MYID: 1
    SERVERS: zoo1
  ports:
    - "2181:2181"
  networks:
    - elkk
```

Figura 19 – Configuração do *Zookeeper*

Kafka

```
kafka1:
  image: wurstmeister/kafka
  container_name: kafka
  depends_on:
    - zoo1
  ports:
    - "9092:9092"
  environment:
    KAFKA_BROKER_ID: 1
    KAFKA_ADVERTISED_PORT: 9092
    KAFKA_ZOOKEEPER_CONNECT:  zoo1:2181
    KAFKA_OFFSETS_RETENTION_MINUTES: 1
    KAFKA_AUTO_CREATE_TOPICS_ENABLE: 'true'
    KAFKA_ADVERTISED_HOST_NAME: 192.168.2.125
  networks:
    - elkk
```

Figura 20 – Configuração do *Kafka*

Network

Rede interna do *docker* com ligação tipo bridge para comunicação com a rede externa.

```
networks:
  elkk:
    driver: bridge
```

Figura 21 – Configuração Rede interna Docker

Configuração de Logstash

O *logstash* é configurado no ficheiro *logstash.conf* onde designamos os *inputs* e o *output* das mensagens. Como input configuramos o *kafka* com o IP, o *host* e o formato da mensagem utilizado *JSON* e qual o tópico do *kafka* a consumir.

```
input {
    kafka {
            bootstrap_servers => "192.168.2.125:9092"
            codec => "json"
            topics => ["topic"]
    }
```

Figura 22 – Configuração input Logstash

Como output definimos o *elasticsearch* da rede *docker* na porta 9200 com o formato de *JSON*.

```
output {
    elasticsearch {
        hosts => "elasticsearch:9200"
        codec => "json"
    }
```

Figura 23 – Configuração do *output Logstash*

Para efeitos de *debug* configuramos o *stdout*.

```
stdout {
    codec => rubydebug
}
```

Figura 24 – Configuração do stdout Logstash

Noise Sensor

Interface que permite recolher nível de ruído obtido através de um microfone. Esta interface foi desenvolvida em *Python* e o único parâmetro que é utilizado para este projeto é o valor em Decibéis atual.

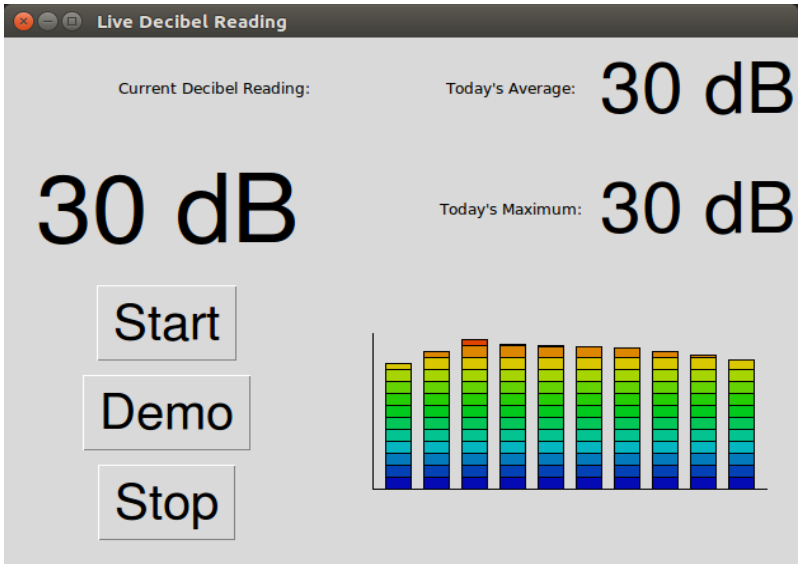

Figura 25 – Interface do noise sensor

Eye Tracker

Interface que permite calibrar a *webcam*, de modo a detetar a pupila do utilizador por parte do utilizador. Esta calibração é um processo simples onde é mostrado um retângulo, sendo apenas necessário clicar na zona do olho do utilizador para que a área de procura da pupila diminua significativamente. Uma vez feita a calibração, o sistema devolve o valor 1 caso o utilizador esteja a olhar e valor 0 em caso contrário.

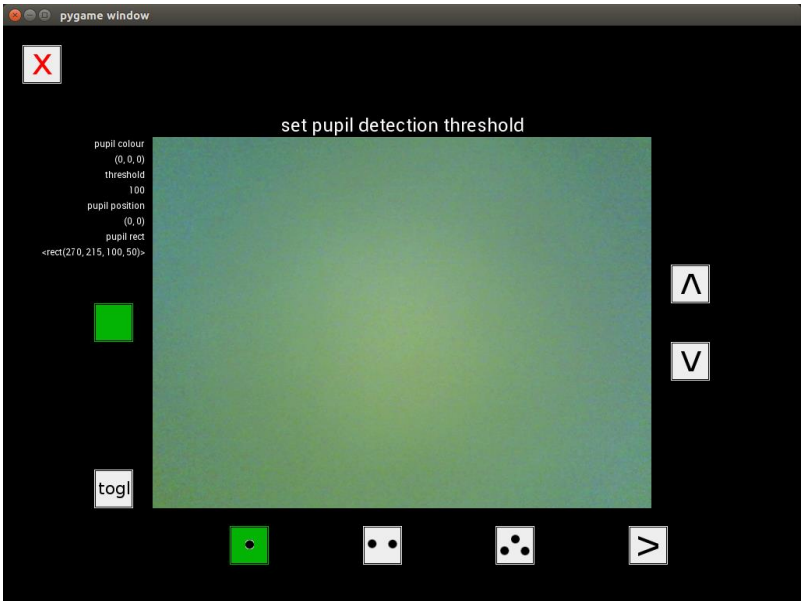

Figura 26 – Interface do *eye tracker*

Keylogger

Interface cujo o objetivo consiste em avaliar a intensidade e frequência das teclas utilizadas pelos utilizadores. A interface do listener recebe os eventos do teclado, designados de *keystrokes*, isto é, o processamento das teclas. Este evento é criado a partir da classe *listener* e é registado através do método *addKeyListener* do componente.

Um evento do teclado é gerado quando uma tecla é *pressed*, *released*, ou *typed*. Assim que o método *addKeyListener* for invocado, o *KeyEvent* [17] é passado por ele.

```
40      @Override
41      public void nativeKeyPressed(NativeKeyEvent e) {
42          System.out.println("Key Pressed: " + NativeKeyEvent.getKeyText(e.getKeyCode()));
43
44          if (e.getKeyCode() == NativeKeyEvent.VC_ESCAPE) {
45              try {
46                  GlobalScreen.unregisterNativeHook();
47              } catch (NativeHookException e1) {
48                  //e1.printStackTrace();
49              }
50          }
51      }
```

Figura 27 – public void nativeKeyPressed(NativeKeyEvent e)

Estes eventos são enviados para o broker do *kafka*, cada vez que o *producer* enviar uma mensagem para o broker. Depois da classe *Properties* proceder à configuração necessária, já é possível criar um *producer*.

```
17  public class SensitiveProducer implements NativeKeyListener {
18
19      public static Properties props = new Properties();
20
21      public static void main(String[] args) {
22
23          props.put("bootstrap.servers", "192.168.2.125:9092");
24          props.put("key.serializer", "org.apache.kafka.common.serialization.IntegerSerializer");
25          props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
26
27          try {
28              GlobalScreen.registerNativeHook();
29          } catch (NativeHookException ex) {
30              System.err.println("There was a problem registering the native hook.");
31              System.err.println(ex.getMessage());
32
33              System.exit(1);
34          }
35
36          GlobalScreen.addNativeKeyListener(new SensitiveProducer());
37      }
```

Figura 28 – SensitiveProducer.java

Sempre que se enviar uma mensagem para o servidor *kafka* é criado um objeto *ProducerRecord*, Este contém dois parâmetros: um que contém o nome do tópico para onde a mensagem irá ser enviada e publicada e outro que está reservado para a mensagem em si.

Por fim, é chamado o método *send*() para concluir o envio da mensagem de forma assíncrona para o tópico.

```
ProducerRecord producerRecord = new ProducerRecord<Integer, JSONObject>("topic", obj);
Producer<Integer, String> producer = new KafkaProducer<Integer, String>(props);
producer.send(producerRecord);
```

Figura 29 – *ProducerRecord*: Criar um tópico kafka

GUI do utilizador

Na máquina do utilizador existe uma interface gráfica no canto superior direito que está sempre visível ao utilizador. Esta interface possui diferentes imagens que aparecem consoante os dados que são recebidos pelos diferentes sensores, dando assim o feedback ao utilizador consoante o seu comportamento.

i) *"Keep It Up"*

Figura 30 - GUI do cliente *"Keep It Up"*

Esta imagem é apresentada quando o utilizador se encontra com um comportamento recomendado para utilizar a máquina. Esta é gerada quando o *pulse sensor* apresenta os batimentos cardíacos entre 60 BPM e 100 BPM. Esta imagem desaparece assim que é detetado algum comportamento irregular por parte do utilizador. A imagem volta a aparecer assim que o utilizar voltar ao seu estado normal.

ii) *"Sleepy"*

Figura 31 - GUI do cliente *"Sleepy"*

Esta imagem é apresentada para alertar o utilizador que está com pouca objetividade, cansado e sonolento. É gerada quando o batimento cardíaco se encontra a um nível abaixo do normal, isto é, quando o *pulse sensor* apresenta valores acima de 0 BPM e abaixo de 60 BPM. É também gerado quando existe uma baixa frequência de teclas pressionadas pelo utilizador, o que representa pouco uso da máquina por parte do utilizador.

iii) *"No There"*

Figura 32 - GUI do cliente *"No There"*

Esta imagem é apresentada quando o utilizador não se encontra a utilizar a máquina. Esta é gerada quando ambas as variáveis, do pulse sensor e do *eye tracker*, se encontram a 0 BPM. Significando que não é registado nenhum batimento cardíaco e que o utilizador não se encontra a olhar para o monitor.

iv) *"Too Much"*

Figura 33 – GUI do cliente *"Too Much"*

Esta imagem é apresentada para alertar o utilizador que está com stress, nervoso ou até mesmo irritado. É gerada quando o batimento cardíaco se encontra a um nível acima do normal, valor do pulse sensor acima de 100 BPM. É também gerado quando existe uma grande frequência de teclas utilizadas como "*Delete*" e o "*Backspace*", caso muito frequente em caso de stress.

GUI do observador

Este GUI foi desenvolvido com o intuito de fornecer ao observador uma interface simples e objetiva. Através de botões, o observador pode ser direcionado para a página do *dashboard* do *kibana*, onde poderá fazer uma análise dos valores apresentados e agir diretamente, consoante os valores, mudando a imagem que é apresentada ao utilizador instantaneamente.

Assim, quando o observador clica numa imagem que pretende que seja apresentada ao utilizador, o sistema envia, via *kafka*, uma mensagem com o estado do utilizador, por forma a alterar a imagem.

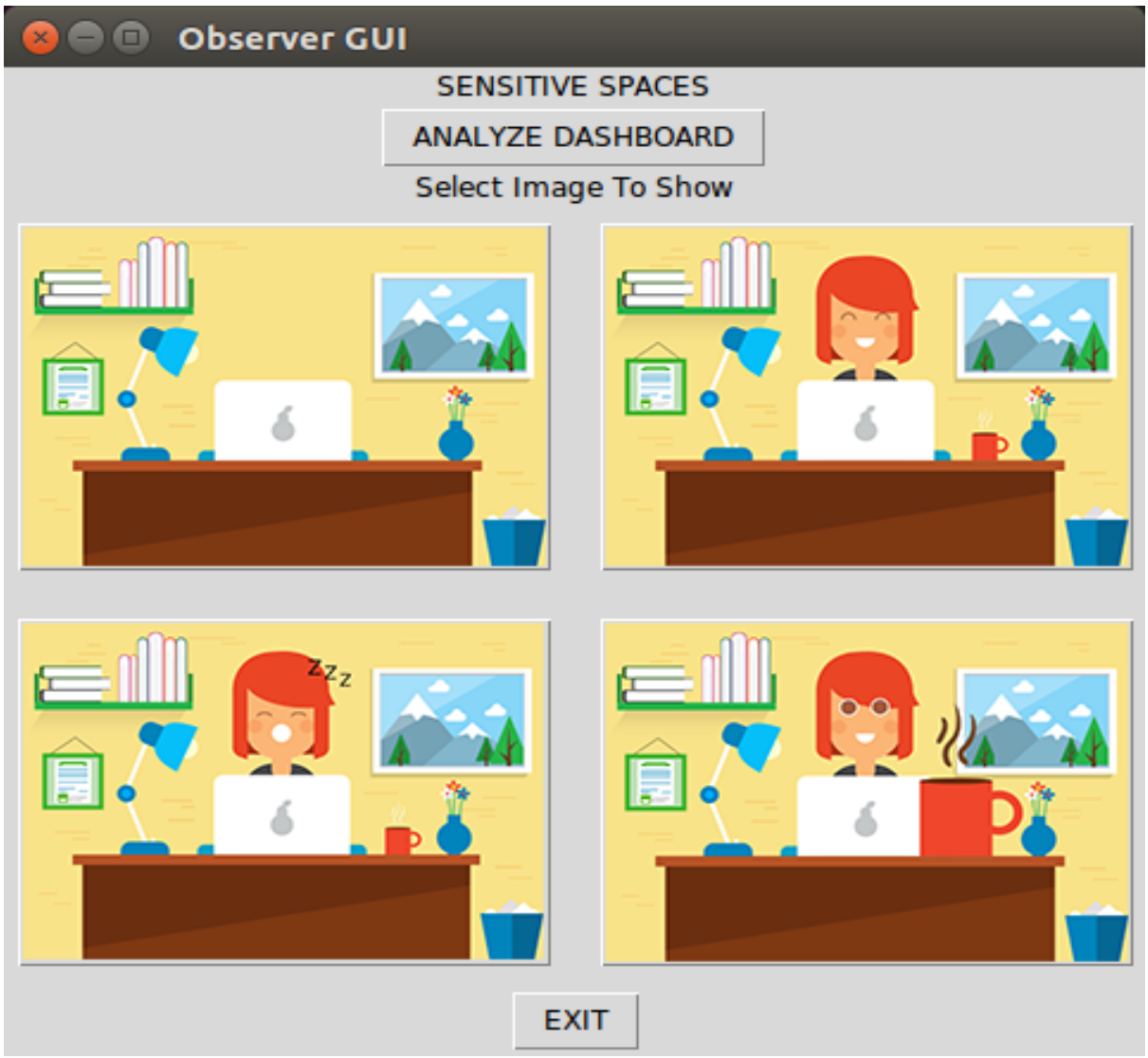

Figura 34 – GUI do Observador

Dashboard Kibana

O *dashboard* do *kibana* permite ao observador visualizar todos os dados recebidos pelos diferentes sensores utilizados. Consoante os dados recebidos, o observador tem a possibilidade de analisar o comportamento atual do utilizador pois, estes dados são recebidos em tempo real, o que permite ao observador uma interação instantânea com o utilizador.

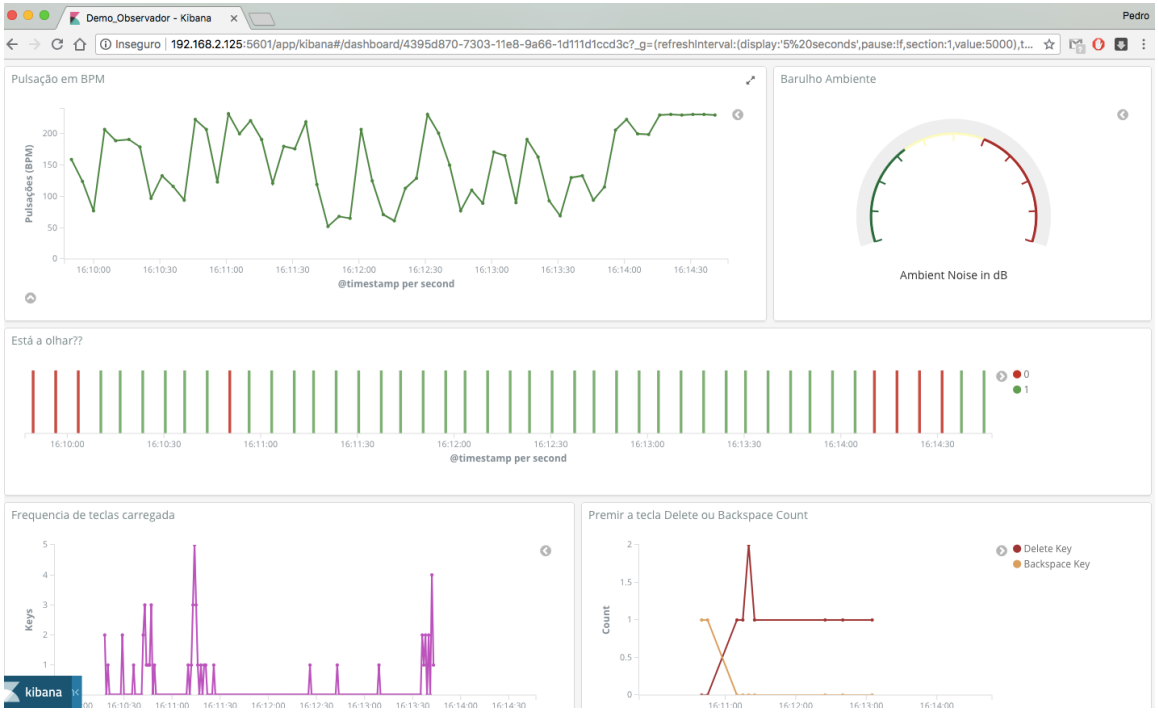

Figura 35 – *Dashboard* do *Kibana*

Gráfico 1 – Apresenta os batimentos cardíacos, em BPM, recebidos pelo *pulse sensor*.

Gráfico 2 – Apresenta a unidade de decibéis que são captados pelo microfone.

Gráfico 3 – Apresenta os valores recebidos pelo *eye tracker*, verde para quando se encontra a olhar e vermelho para o caso oposto.

Gráfico 4 – Apresenta a frequência com que o utilizador pressiona as teclas do teclado, valores recebidos pelo *keylogger*.

Gráfico 5 – Apresenta a frequência com que as teclas específicas, "*delete*" e "*backspace*", são utilizadas pelo utilizador, valores recebidos pelo *keylogger*.

Capítulo 5

Resultados e conclusões

O sistema é capaz de monitorizar o comportamento do utilizador em tempo real e fornecer *feedback* com o intuito de melhorar a sua produtividade. O observador é capaz de interpretar os dados recebidos pela monitorização e alterar a imagem do GUI do cliente durante 10 segundos sobrepondo-se ao feedback automático. O processamento dos dados do GUI é conseguido apenas pelo *eye tracker* e pelo *pulse sensor*, não tendo sido implementado a parte do *keylogger*.

Problemas e Falhas

Foram adquiridos no *ebay* alguns *pulse sensors* de baixo custo e após uma análise exaustiva, foram detetados erros de fabrico dos sensores, concluindo que estes reportam dados errados.

Os chips ADC MCP3008 demonstram um comportamento anormal porque reportam 0 BPM constantemente, devido à frequência do SPI do *raspberry pi*. Para que os valores fiquem corretos optou-se por baixar a frequência no driver MCP3008.py [18].

Students@DETI 2018

No passado dia 12 de Junho de 2018, o grupo *Sensitive Spaces* esteve presente no Students@DETI [19], uma feira de projetos dos alunos do DETI, onde fez a apresentação do seu protótipo funcional desenvolvido na unidade curricular de projeto, inserido no 3º ano de Licenciatura, em Engenharia Informática. Além do protótipo e do poster [A], também se disponibilizaram *flyers* [B] iguais ao da imagem seguinte.

O evento decorreu de forma bastante positiva e o feedback também foi bem recebido por parte dos alunos, professores e algumas empresas. Dada a estrutura da arquitetura, a área que mais despertou interesse foi a do *keylogger*.

Referências

[1] Eye Tracking, http://www.eyetracking.com/About-Us/What-Is-Eye-Tracking

[2] Elétrodos, https://pt.wikipedia.org/wiki/Elétrodo

[3] Keylogger, https://pt.wikipedia.org/wiki/Keylogger

[4] Key Listener, https://docs.oracle.com/javase/8/docs/api/index.html?java/awt/event/KeyListener.html

[5] Docker, https://www.docker.com/what-docker

[6] Docker Engine, https://docs.docker.com/engine/

[7] Imagem de Mike Mackrory, Rancher Labs, https://rancher.com/playing-catch-docker-containers/

[8] Docker Compose, https://docs.docker.com/compose/overview/

[9] Imagem do Unixman, https://www.unixmen.com/container-docker-compose-ubuntu-16-04/

[10] Apache Kafka, https://kafka.apache.org/

[11] Kafka Producer, http://bigdatums.net/2017/04/27/creating-simple-kafka-producer-java/

[12] Kafka Consumer, http://bigdatums.net/2017/05/18/creating-a-simple-kafka-consumer/

[13] Logstash, https://www.elastic.co/guide/en/logstash/current/first-event.html

[14] ELK, https://www.elastic.co/elk-stack

[15] MCP3008, https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008

[16] Raspberry-Pi-Heartbeat-Pulse-Sensor, https://github.com/tutRPi/Raspberry-Pi-Heartbeat-Pulse-Sensor

[17] Key Event, https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html

[18] Comportamento anormal dos chips ADC MCP3008, https://www.raspberrypi.org/forums/viewtopic.php?t=115971

[19] Students@DETI, http://studentsandteachersdeti.web.ua.pt/studentsatdeti/

Anexos

[A] Poster do Students@DETI 2018

[B] Flyer do Students@DETI 2018

Anexo A

students @deti 2018

Sensitive Spaces

Pedro Fernandes, Andreia Castro, João Limas, Alex Santos

Orientador: Prof. José Maria Fernandes

Co-Orientador: Prof. Ilídio Oliveira

Projeto em Engenharia Informática, 3º ano, LEI.

Resumo

O projeto Sensitive Spaces visa avaliar a prestação de um utilizador de computador através de input de teclado, *webcam*, áudio e pulsação e fornecer *feedback* com o intuito de melhorar a sua produtividade. A integração prevê uma utilização futura para equipas e com a capacidade de *cloud computing*. Esta avaliação é feita através de sensores apropriados e pretende quantificar o uso intensivo das teclas do computador, bem como, a sua prestação e quantidade de erros obtidos durante a compilação, informando o utilizador quando deve descansar.

Contexto

Por forma a avaliar o estado psicológico de um utilizador enquanto este desenvolve a sua solução, o sistema Sensitive Spaces reúne dados para que estes sejam analisados e devolvam *feedback* adequado. Deste modo, a proposta de arquitetura apresentada é composta pelas seguintes tecnologias:

KeyLogger – regista em tempo real todas as teclas que o utilizador digita, de modo a avaliar a intensidade com que são pressionadas.

PulseSensor – recolhe de forma contínua e autónoma a variação da frequência cardíaca do utilizador para uma análise posterior.

EyeTracker – avalia, em tempo real, se o sujeito em análise se encontra a olhar para o ecrã, através de uma *webcam*.

NoiseSensor – regista, em tempo real, o ruído existente no ambiente envolvente, através de um microfone.

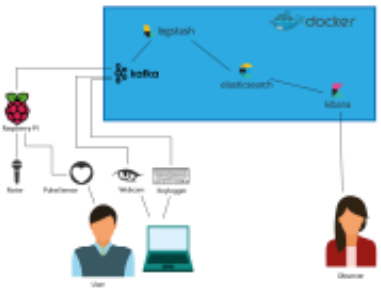

Fig 1- Arquitetura do Sistema

Tratamento de Dados

Uma vez recolhidos os dados dos respetivos sensores, estes irão ser tratados para o sistema agir adequadamente e para serem apresentados ao observador. Este observador será um estudante ou um profissional de psicologia, capaz de analisar toda a informação recolhida em tempo real.

SENSITIVESPACES

Fig 2- Proposta de logótipo

Este tratamento de dados é feito usando as seguintes tecnologias:

Docker – plataforma que permite a execução e armazenamento com recurso à tecnologia de *containers* para *deployment* na internet *cloud computing* [1].

Kafka – plataforma de transmissão de dados em tempo real. O funcionamento é semelhante a uma fila de mensagens [2].

Logstash – sistema que processa os dados provenientes de diversas fontes, preparando-os para serem enviados [3].

Elasticsearch – mecanismo de armazenamento, pesquisa e análise de dados [3].

Kibana – permite a visualização dos dados, através de mapas e gráficos [3].

Resultados

O sistema possui uma interface gráfica que se encontra sempre visível no canto do ecrã, com o intuito de alertar o utilizador para o seu estado psicológico. Após uma análise dos resultados obtidos pelos diferentes sensores, o sistema é capaz de alterar a imagem mostrada para que esta seja adequada ao estado do utilizador.

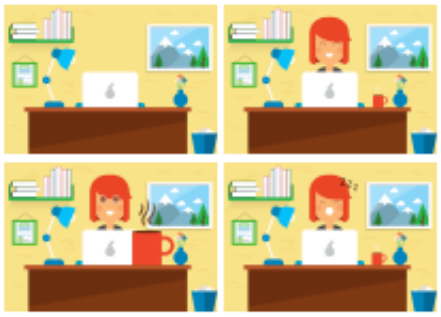

Fig 3- Imagens apresentadas pelo GUI para os diferentes estados.

Referências

[1] Docker. Disponível https://www.docker.com/
[2] Kafka. Disponível https://kafka.apache.org/
[3] ELK. Disponível https://www.elastic.co/elk-stack

universidade de aveiro theoria poiesis praxis

deti departamento de electrónica, telecomunicações e informática

Figura 36 – Poster Students@DETI

Anexo B

Figura 37 – Flyer Students@DETI