



Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Projeto Integrador Engenharia 2

UMISS - Unidade Móvel de Identificação de Saúde e Socorro

Projeto UMISS
Orientadores: Alex Reis, Luiz Laranjeira, Rhander Viana e
Sebastièn Rondineau

Brasília, DF

2017



Afonso Delgado, Cesar Marques, Dylan Guedes, Felipe Assis, Gustavo Cavalcante, Johnson Andrade, Lucas Castro, Lunara Martins, Mariana Andrade, Nivaldo Lopo, Rafael Amado, Tiago Assunção, Wilton Rodrigues

UMISS - Unidade Móvel de Identificação de Saúde e Socorro

Relatório técnico referente à disciplina de Projeto Integrador 2, reunindo os cursos de Engenharias presentes no Campus Gama, da Universidade de Brasília.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Alex Reis, Luiz Laranjeira, Rhander Viana e Sébastien Rondineau

Brasília, DF

2017

UMISS - Unidade Móvel de Identificação de Saúde e Socorro

Relatório técnico referente à disciplina de Projeto Integrador 2, reunindo os cursos de Engenharias presentes no Campus Gama, da Universidade de Brasília.

Brasília, DF

2017

Resumo

Pacientes com capacidade motora reduzida, em certo grau, necessitam de observação contínua a fim de evitar acidentes ou outros problemas. Além disso, em alguns casos, a presença de um cuidador é necessária para ajudar na movimentação da cadeira de rodas e na captura de sinais vitais. Tecnologias nesse campo não evoluem rápido o suficiente, não resolvem estes cenários ao mesmo tempo, e, mais ainda, são custosas. Neste trabalho nós apresentamos a UMISS, uma cadeira elétrica que extrai sinais vitais, notifica eventos críticos, e se move sem intervenção de terceiros. Com a UMISS nós esperamos criar uma solução de baixo custo, que permita ao paciente cuidar de si mesmo de maneira segura.

Palavras-chaves: cadeira de rodas. acessível. monitoramento. sensores.

Abstract

Handicapped people, in a certain degree, needs continuous monitoring in order to prevent accidents or other issues. Besides that, in some cases, the presence of a carer is needed to help with the wheelchair, and to track vital signals. Technologies in this field are not evolving fast enough, does not solve these scenarios at the same time, and, even more, are costly. In this work we present UMISS, an electric wheelchair that tracks vital signals, notifies critical events, and moves without third party intervention. With UMISS we expect to create a low cost solution, that allows the patient to securely take care of himself.

Key-words: wheelchair. accessible. monitoring. sensors

Listas de ilustrações

Figura 1 – Sensor para medição de temperatura corporal.	11
Figura 2 – Eletrodos fabricados para medição de umidade e resistência galvânica da pele.	11
Figura 3 – Circuito para captura de umidade da pele por GSR.	11
Figura 4 – Sensor de presença do usuário na cadeira de rodas.	12
Figura 5 – Sensor ótico para captura de sinais PPG da fabricante <i>SparksFun</i>	13
Figura 6 – Esquemático do circuito condicionador de sinais PPG proposto pela fabricante	13
Figura 7 – <i>Design</i> e Arquitetura do servidor <i>Django</i> (TECHNOLOGY, 2013)	16
Figura 8 – Diagrama da arquitetura parcial do UMISS-frontend.	18
Figura 9 – Layout atual da tela principal do módulo Android.	20
Figura 10 – Diagrama de classes do módulo Android.	21
Figura 11 – <i>Design</i> e Arquitetura do servidor Django	28

Lista de tabelas

Lista de abreviaturas e siglas

PSM	Processamento de Sinais e Monitoramento
CeA	Controle e Alimentação
PE	Projeto Estrutural
PC1	Ponto de controle 1
UMISS	Unidade Móvel de Identificação de Saúde e Socorro

Sumário

1	INTRODUÇÃO	9
2	PROCESSAMENTO DE SINAIS E MONITORAMENTO	10
2.1	Aquisição e Condicionamento de Sinais	10
2.2	Middleware	14
2.2.1	Arquitetura	15
2.3	<i>Backend Django</i>	15
2.3.1	Papel do <i>Backend</i>	15
2.3.2	<i>Design</i> e Arquitetura da Solução	16
2.3.3	Projeto Desenvolvido	17
2.4	Módulo JavaScript EmberJS	18
2.5	Módulo Android	18
2.5.1	Funcionalidades	19
2.5.2	Arquitetura	20
3	CONTROLE E ALIMENTAÇÃO	22
4	ESTRUTURAS	23
5	ATIVIDADES	24
6	CONSIDERAÇÕES FINAIS	25
	REFERÊNCIAS	26
	ANEXOS	27
	ANEXO A – DIAGRAMA DE CLASSES BACKEND	28

1 Introdução

Neste relatório apresentamos o andamento e finalização do projeto WheelShare (antes chamado UMISS) que ocorreu entre os pontos de controle 1, 2 e 3. Focaremos principalmente em questões práticas, e levantaremos os resultados obtidos e os esperados.

O objetivo desta etapa é integrar os subsistemas desenvolvidos, com base no Plano de Integração desenvolvido na segunda parte da disciplina. Desta maneira, este relatório trata sobre os resultados obtidos ao longo das últimas semanas de implementação e integração do Ponto de Controle 3.

Organizamos o relatório da seguinte forma: cada capítulo que segue será relativo aos resultados da integração de um subsistema do projeto. No Capítulo 2 apresentamos o andamento e integração do subsistema de Processamento de Sinais e Monitoramento, que contempla o *middleware*, a aquisição de sinais, o servidor remoto (*backend*), o cliente *web* (*frontend*) e o aplicativo Android. No Capítulo 4 será apresentado o andamento e integração do subsistema de Estruturas. No Capítulo 3 será apresentado o andamento e integração do subsistema de Controle e Alimentação, responsável pela movimentação e alimentação dos sistemas integrantes do projeto. No Capítulo 5 apresentamos as atividades feitas por cada membro do grupo, e por fim, no Capítulo 6 apresentamos nossas considerações finais sobre o projeto e a disciplina.

2 Processamento de Sinais e Monitoramento

2.1 Aquisição e Condicionamento de Sinais

O projeto eletrônico do sistema de processamentos de sinais e monitoramento consiste em quatro módulos de sistemas de captura e condicionamento para os seguintes sinais: temperatura corporal e do ambiente, resistência galvânica da pele (GSR), acontecimento de quedas do usuário da cadeira de rodas e eletrocardiograma.

Para o circuito de monitoramento de sinais de temperatura, foi utilizado um sensor de medição linear de temperatura LM35, fabricado pela *Texas Instruments*. O sensor foi selecionado por sua variação linear de tensão de saída com a variação de temperatura, além do seu baixo consumo e alta precisão de medidas de temperatura com rápida resposta e sua faixa de captura de -55 à 150 graus Celsius, tornando-o viável para monitoramento de temperaturas corporais e ambientais. Tanto o circuito quanto o sistema embarcado foram projetados e dimensionados para tratar de sua variação linear de 10 mV em sua saída a cada 1 grau de variação na temperatura¹.

Foram também realizados testes e algoritmos para medição da temperatura corporal com termistores NTC, porém, os mesmos mostraram-se menos acurados devido ao seu comportamento não linear dado pela equação de *Equação de Steinhart-Hart*² e problemas de arredondamento de valores capturados pelo sistema embarcado. Dessa forma, foi decidido o uso do sensor LM35.

O sensor fabricado para medição da temperatura corporal, apresentado na Figura 1, é aplicado à superfície da pele, no antebraço, possibilitando leitura facilitada e de forma menos invasiva. Uma vez em contato com o corpo do usuário, o sensor apresenta alteração nos valores lineares de temperatura até o momento de estabilidade com a temperatura corporal.

O sistema de monitoramento de sinais de resistência galvânica da pele (GSR), tem como principal função o monitoramento de variações na umidade da pele do usuário, podendo indicar níveis perigosos de estresse ou até mesmo colapsos de hipoglicemias ou por fraqueza por falta de alimentação. O sistema consiste em dois eletrodos fabricados, como visto na Figura 2, para contato com o antebraço do usuário, com uma distância fixa de 5 cm entre eles, dessa forma, mantendo os valores das medições confiáveis para cada usuário com diferentes tipos de pele.

Além disso, o circuito, visto na Figura 3 embarcado conta com um capacitor de

¹ <<http://www.ti.com/lit/ds/symlink/lm35.pdf>>

² <<http://www.sciencedirect.com/science/article/pii/0011747168900570?via%3Dihub>>

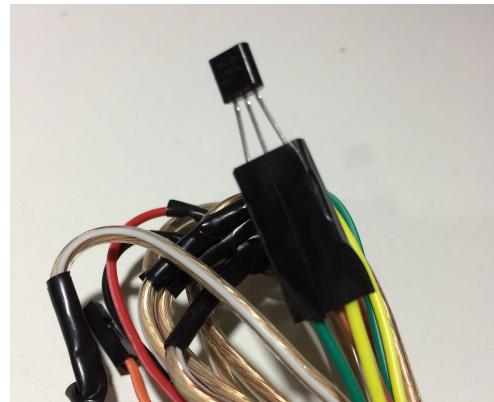


Figura 1 – Sensor para medição de temperatura corporal.

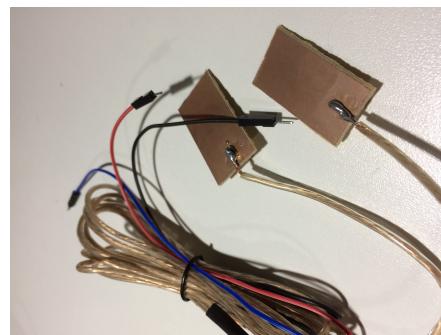


Figura 2 – Eletrodos fabricados para medição de umidade e resistência galvânica da pele.

acoplamento para o contato com a pele humana e um resistor de alta impedância para a medição da variação da resistência da pele por divisor de tensão. O sistema foi simplificado para que pudesse ser implementado junto aos cabos dos eletrodos, de forma a utilizar o mínimo de espaço possível na cadeira de rodas.



Figura 3 – Circuito para captura de umidade da pele por GSR.

O algoritmo para o monitoramento da resistência galvânica e umidade conta com comparadores para diferentes status do usuário, podendo alertar quando o usuário não está realizando a medição, quando apresenta características normais para sua pele ou quando

o usuário encontra-se em situação de risco. O sistema embarcado para o monitoramento e integração com o *middleware* foi programado em linguagem Python.

O sistema de alerta de quedas é responsável por enviar alertas para o servidor caso o usuário encontre-se fora da cadeira, indicando que o mesmo caiu da cadeira e necessita de auxílio urgente. Para esse sistema, apresentado na Figura 4 foi desenvolvido um circuito com um sistema de medição de distância e presença utilizando um módulo com sensor piezoelétrico. O sistema é responsável pelo monitoramento de variações de tensão no sensor, possibilitando o processamento dessa variação, alertando a presença ou não do usuário na cadeira de rodas. O sensor piezoelétrico é inserido no assento da cadeira para que o mesmo esteja pressionado durante todo o momento que o usuário esteja sentado na cadeira, e dessa forma, o algoritmo é capaz de alertar caso a presença do usuário na cadeira não seja identificada.



Figura 4 – Sensor de presença do usuário na cadeira de rodas.

O sistema de monitoramento dos sinais de eletrocardiografia foi desenvolvido para captura dos sinais de pulso cardíaco com um sensor fotoelétrico e um LED verde de alto brilho, a partir de sinais de PPG (photoplethysmografia), para que as medidas e a captura do sinal seja feita da forma menos invasiva possível. Para isso, o circuito deve ser capaz de capturar o sinal de pulso cardíaco por meio ótico com precisão e assim, foi utilizado o sensor SEN-11574, apresentado na Figura 5, baseado em um amplificador operacional MCP6001, que possui produto ganho-banda de 1MHz e possui aplicações específicas para amplificação de sinais provenientes de fotodiodos e fototransistores. Além disso, o sistema consiste em um LED de alto brilho de cor verde e um sensor fotodiode APDS9008 de alta sensibilidade a variações de iluminação, capaz de capturar sinais de PPG com a passagem e retorno de luz pelo tecido humano. Além do amplificador e do sensor selecionado, foram projetados filtros passa-alta para rejeição de frequências abaixo de 0.7Hz que possam aplicar um nível DC no sinal, que acabam tendo papel fundamental na etapa de captura da frequência cardíaca pois elevam o sinal, dificultando a captura da frequência por meio de um algoritmo presente no sistema de processamento digital. O circuito utilizado é dado pelo esquemático apresentado na Figura 6

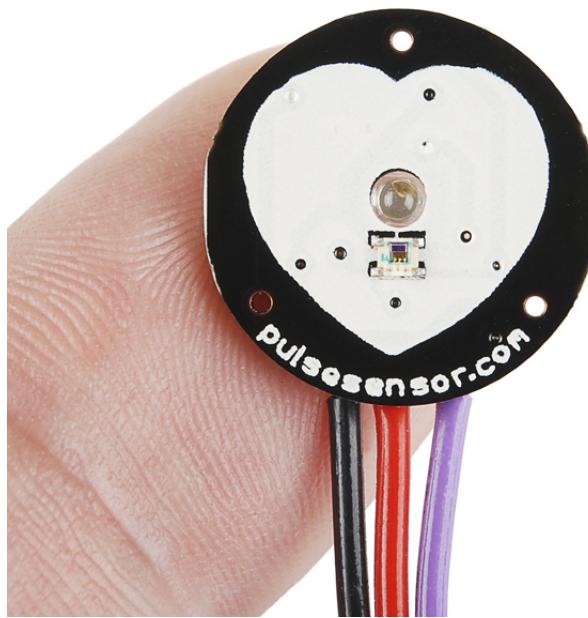


Figura 5 – Sensor ótico para captura de sinais PPG da fabricante *SparksFun*

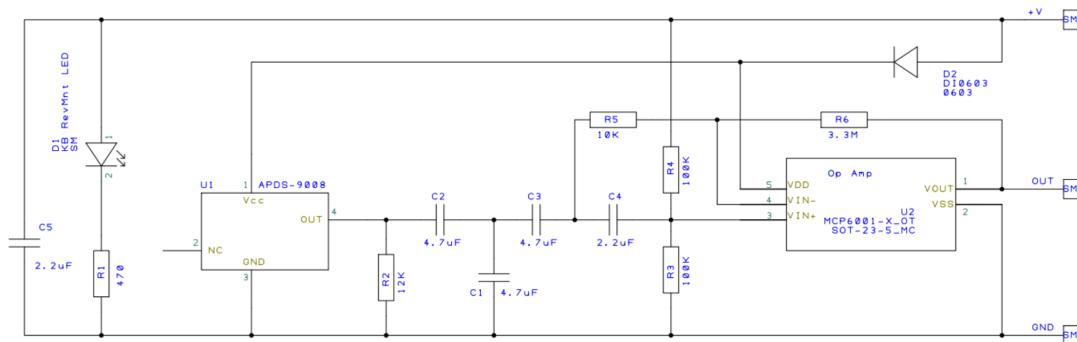


Figura 6 – Esquemático do circuito condicionador de sinais PPG proposto pela fabricante

É importante ressaltar que os sinais de pulso cardíaco e eletrocardiograma variam entre as baixas frequências de 0.5Hz e 40Hz³, e dessa forma os sistemas de condicionamento foram projetados de forma a atenuar frequências maiores que 40Hz enquanto aplicam um alto ganho de tensão AC para a leitura do sinal original de baixa amplitude, entre 0.1 mV e 5 mV. O módulo do ganho de tensão AC do sinal é dado pelo ganho do amplificador MCP6001 em modo não inversor, dado por:

$$Av = 1 + \frac{3.3M}{10k} = 330 \quad (2.1)$$

Com esse ganho, é possível obter os valores de pico dos pulsos cardíacos na faixa

³ Yazicioglu RF, van Hoof C, Puers R. Biopotential Readout Circuits for Portable Acquisition Systems, 2009, Springer Science, ISBN: 978-1-4020-9092-9)

de 1.65 V.

O algoritmo de captura da frequência cardíaca via sinais de PPG foi desenvolvido a fim de calcular o período entre sinais de pulsos sistólicos, eliminando a ocorrência de pulsos diastólicos de menor amplitude.

Após a captura e condicionamento de todos os sinais analógicos, os mesmos são direcionados para um módulo de conversão Analógico-Digital (A/D). O conversor selecionado foi o ADS1115⁴, devido a sua alta resolução de 16 bits, capaz de converter valores analógicos em valores digitais de 0 a 65535, e por sua taxa de amostragem de 800Hz, suficientes para a captura dos sinais a serem monitorados para o projeto. O conversor é conectado com a Raspberry Pi via protocolo I2C, possibilitando o uso de menos portas para o envio de até 4 sinais simultâneos.

Uma vez com os sinais devidamente convertidos para digital, o sistema embarcado e de *middleware* é responsável por monitorar os sinais, realizando verificações de seus valores a partir de calibrações realizadas para cada um dos módulos.

2.2 Middleware

O *middleware* do subsistema, uma Raspberry, tinha como resultados esperados uma aplicação que pudesse, ao rodar no embarcado, receber sinais e enviá-los de maneira correta ao servidor.

Os resultados esperados foram atingidos. Foi desenvolvido a aplicação Shoelace⁵, que serve como abstração para a aquisição dos dados do conversor A/D e que envia os resultados para um servidor remoto.

Definimos que uma regra de negócio deveria ser que toda Raspberry tivesse um *token* e uma senha incluída, e esses dados são então utilizados nas requisições para os servidores. Isso foi feito através da geração de dados aleatórios (*token* e senha), que são obtidos sempre que a Raspberry é ligada, por estarem no *bash_rc*. Assim, todas as adições de sinais feitas por uma Raspberry já serão relacionados com o respectivo paciente, que poderá ter seus dados visualizados por parentes cadastrados.

A comunicação entre a Raspberry e o conversor é feita através do pacote em Python **Adafruit_ADS**, capaz de ler de até quatro canais ao mesmo tempo. Contudo, uma ressalva: os valores enviados pelo sensor de temperatura não são recebidos de uma maneira apresentável, por não estarem normalizados. Utilizamos então a equação de Steinhart-hart:

$$1/t = A + B * \ln(R) + C[\ln(R)]^3$$

⁴ <<https://cdn-shop.adafruit.com/datasheets/ads1115.pdf>>

⁵ <<https://github.com/cadeiracuidadora/shoelace>>

Onde A, B e C são coeficientes, R é a resistência, e T a temperatura que desejamos apresentar. Utilizamos os seguintes valores para os coeficientes:

$$A = 0.001129148; B = 0.000234125; C = 0.0000000876741$$

Para o cálculo do \ln utilizamos a função $\log1p$ do Python, e ressaltamos que tivemos diversos problemas de precisão, pois o Python arredonda os resultados das operações de maneira grosseira em diversas situações. Por fim, ajustamos outros parâmetros (como os utilizados na conversão da resistência) utilizando outros resultados como base, de maneira experimental.

Para diminuir o consumo de banda, criamos um mecanismo que aborta o envio de sinais redundantes.

2.2.1 Arquitetura

2.3 Backend Django

Esta sessão tratará sobre o servidor do sistema, também conhecido como *backend* ou *API Django*. Na primeira subsessão, é tratada uma breve descrição sobre as funções e características deste. A segunda apresenta o *design* e a arquitetura de desenvolvimento. Por fim, trataremos as instâncias do projeto desenvolvido.

2.3.1 Papel do Backend

O *backend* do sistema, responsável por fazer o controle das requisições, atuando como intermediador à cadeira e aos seus respectivos monitores foi desenvolvida utilizando o *Django Rest Framework*, que é uma referência no mundo de desenvolvimento de API's para *Python*.

O *backend* manipula todos os dados enviados pela cadeira, utilizando a *Raspberry* executa o processamento desses dados e envia notificações para os *smartphones* cadastrados no sistema. Além disso, serve dados para uma interface web de controle do monitor.

A manipulação dos dados é feita de forma segura, tratando a autenticação dos usuários que lidam com o sistema e redirecionando os dados de acordo com a sua autenticação. Para a execução dos passos, o servidor conta com um poderoso sistema de autenticação dos seus usuários, tanto para os pacientes que utilizam a cadeira, quanto para os monitores que manipulam as interfaces para cliente, como mobile e interface web. Os pacientes são cadastrados no sistema e enviam os sinais corporais para o servidor. O servidor, além de enviar uma notificação para o monitor respectivo, armazena este sinal em sua base de dados e retorna ao monitor quando solicitado. Os dados enviados ao monitor são apenas os sinais respectivos ao paciente que está sendo monitorado.

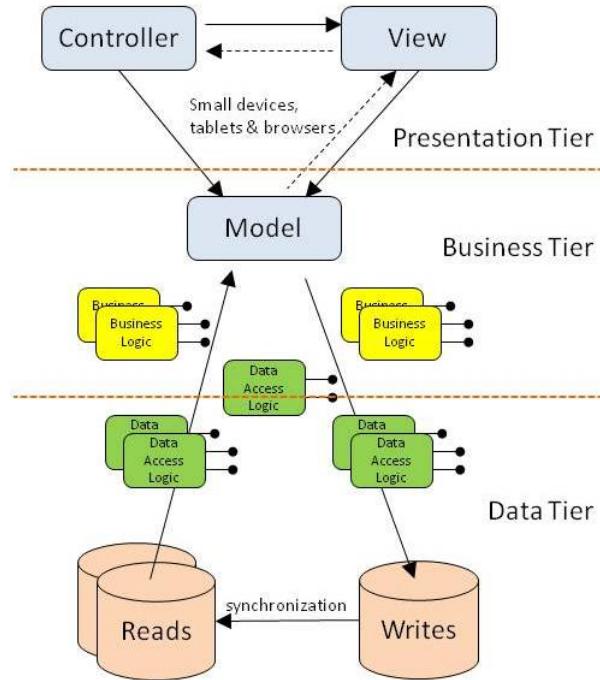


Figura 7 – *Design e Arquitetura do servidor Django* ([TECHNOLOGY, 2013](#))

A ligação entre o paciente e o monitor pode ser feita no momento do cadastro ou em atualizações futuras. O paciente contém um identificador único em sua cadeira, que pode ser utilizado pelo monitor para supervisioná-lo. No momento que o monitor insere os dados no sistema, automaticamente será ligado ao paciente e terá acesso aos dados de sinais enviados pelo paciente.

2.3.2 Design e Arquitetura da Solução

O *django* utiliza de uma arquitetura própria para padronização e aplicação de boas práticas em desenvolvimento de API's. Esta arquitetura está dividida em *Tiers*, que são responsáveis por funções específicas e cada uma tem seu papel definido para a manipulação de dados no sistema. A seguir está uma ilustração do funcionamento da arquitetura do servidor.

Como pode ser observado, existem três *Tiers* principais, que são utilizadas para a manipulação dos dados. Elas são:

- *Presentation Tier*;
- *Business Tier*;
- *Data Tier*.

A primeira, *Presentation Tier* é responsável por ser a interface com as demais aplicações que irão utilizar o servidor. Essa *tier* é responsável por gerir as rotas do sistema, direcionando as urls solicitadas a funções específicas. Após mapeado nas funções, estas irão verificar as permissões de cada requisição, bem como cada usuário que está solicitando uma determinada ação.

Além disso, nesta *tier*, existe uma transformação importante para o consumo de dados por parte dos clientes. Esta é conhecida como serialização dos dados, tanto para quem envia, quanto para quem solicita. Sua responsabilidade é aplicar um modelo de dados formalizado na comunidade, conhecido como *json*. Dessa forma, todo tipo de dado recebido, assim como os enviados são transformados a partir dos objetos *Python*. Esta fase é aplicada logo após a aplicação das rotas para as respectivas ações.

As funções mapeadas na *Tier* passada utiliza de modelos predefinidos na Business *Tier*. Aqui são definidos os domínios da aplicação, bem como quais são as classes e negócio da aplicação. Nesta fase são definidos os tipos de usuários do sistema, as classes de sinais corporais do paciente e toda parte de negócios da aplicação.

A última *tier*, chamada de *Data Tier* é responsável por lidar com a persistência dos dados de todos os usuários e sinais que são manipulados no sistema. Esta *tier* modela o bando de dados assim como foi definido na aplicação de negócios e aplica inserções neste a medida que novas requisições são feitas.

2.3.3 Projeto Desenvolvido

O projeto UMISS-BackEnd, desenvolvido exatamente na arquitetura apresentada na sessão 2.3.2, possui entidades que deram formato à Business *Tier* do projeto, onde são definidas as propriedades do sistema, bem como ele atuará.

No projeto, foi desenvolvido o diagrama de classes, que auxilia na atividade de compreender o domínio do projeto. Este está no Anexo A.

É possível visualizar que existem várias entidades de usuários. Isto foi desenvolvido pois existem dois tipos de usuários diferentes no sistema. Eles são o Paciente e o Monitor. Estes possuem atributos e comportamentos diferentes, gerando assim a necessidade de implementação de dois domínios diferentes. Da mesma maneira, existem três tipos de sinais corporais que são enviados pelo paciente. Esses sinais possuem características diferentes, bem como atributos. Dessa maneira, existe uma entidade generalista chamada *BodySignal* com características comuns de todas. E foram criados as demais especializações com suas características específicas. Elas são: Sinais de temperatura corporal, Sinais de batimento cardíaco e Sinais de resistência galvânica.

Por fim, existem ligações dos usuários com os sinais corporais, sendo que os pacientes são donos destes. Assim como existem pacientes que possuem monitores ligados,

gerando um relacionamento cíclico entre as entidades de usuário.

2.4 Módulo JavaScript EmberJS

O *frontend* do subsistema, módulo responsável pela visualização dos dados, tinha como resultados esperados uma aplicação que pudesse prover a visualização dos sinais referentes a um determinado paciente, baseado nas informações recebidas do servidor de *backend*. Devido à necessidade de uma certa dinamicidade na apresentação desses dados, foi escolhido o *Framework* JavaScript EmberJS⁶. Desta forma, foi então desenvolvida a aplicação UMISS-frontend⁷, que apresenta de forma gráfica e textual o histórico dos sinais monitorados pelo projeto UMISS.

Baseado no *token* fornecido em cada cadeira, o usuário monitor pode fazer o cadastro na aplicação e assim fazer o monitoramento do paciente vinculado à cadeira em questão. O servidor de *backend* é responsável pela filtragem dos dados recebidos, garantindo então que apenas os dados do paciente vinculado ao token fornecido possam ser visualizados pelo usuário monitor.

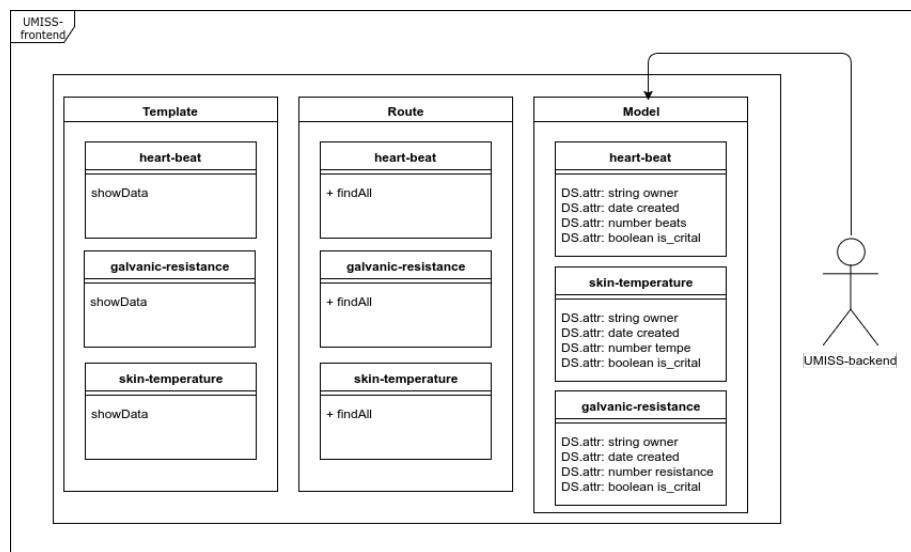


Figura 8 – Diagrama da arquitetura parcial do UMISS-frontend.

2.5 Módulo Android

O módulo Android do sistema é responsável pela visualização dos dados referentes a um determinado paciente e também pela notificação do usuário monitor via *push notification*.

⁶ <<https://emberjs.com/>>

⁷ <<https://github.com/cadeiracuidadora/UMISS-frontend>>

Este módulo atua como cliente, ou seja, os dados referentes a um paciente estão armazenados no *backend* e a aplicação Android somente consome esses dados. O único valor que é armazenado na aplicação Android é um *token* de autenticação enviado do servidor quando um usuário monitor efetua o *login*.

2.5.1 Funcionalidades

- *Login:*

É possível fazer login com um usuário do tipo monitor na aplicação, ao fazê-lo, é recebido um *token* de autenticação do servidor que é armazenado no Android. Essa etapa se difere um pouco do *frontend* pois no Android ocorre um passo adicional de enviar um identificador para o *backend* ser capaz de enviar as notificações para o Android.

- *Cadastro:*

Caso o usuário do tipo monitor ainda não tenha feito um cadastro no *frontend*, é possível fazer o cadastro direto da aplicação Android, basta informar os dados de *login*, senha e o identificador da cadeira.

- *Notificações:*

Caso o *backend* receba um dado crítico do *shoelace*, é enviado ao aplicativo Android uma notificação do tipo *push notification* para que o usuário seja alertado que há algo errado acontecendo com o paciente.

- *Gráficos:*

Os dados referentes à temperatura, batimentos cardíacos e resistência galvânica são exibidos em forma de gráfico, toda vez que o usuário abrir a aplicação uma nova requisição será feita ao servidor e os gráficos serão criados.

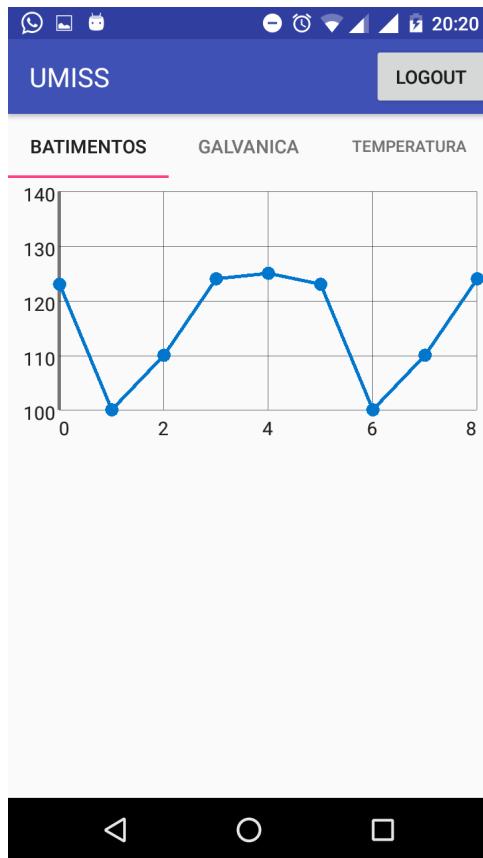


Figura 9 – Layout atual da tela principal do módulo Android.

2.5.2 Arquitetura

A arquitetura do módulo Android foi elaborada da seguinte maneira, foram criados dois pacotes, um chamado *com.umiss* que contém todas as *fragments* e *activities* da aplicação e um pacote chamado *network* que contém as classes responsáveis pelas requisições e notificações.

A *MainActivity* é a classe principal do sistema, ao iniciar, é executado o método *isLogged* que verifica se existe um *token* de autenticação válido, caso exista, será instanciado as *fragments* *HeartBeatsFragment*, *GalvanicFragment* e *TemperatureFragment* que são responsáveis pela criação dos gráficos de batimentos cardíacos, temperatura e resistência galvânica respectivamente. Caso contrário, será instanciado a *activity* de *Login*, a partir da *activity* de *Login* é possível abrir a *activity* de *Cadastro*.

O pacote *network* contém a classe *UMISSRest* que é responsável pelos métodos http *POST*, *GET* e *PUT*. Todas as classes que realizam alguma requisição com o *backend* utilizam os métodos estáticos da classe *UMISSRest*. As notificações são feitas pelo serviço *Firebase Cloud Messaging* através das classes *MyFirebaseMessagingService* que é responsável por criar uma notificação e pela classe *MyFirebaseInstanceIdService* que cria

um identificador que é enviado ao *backend*.

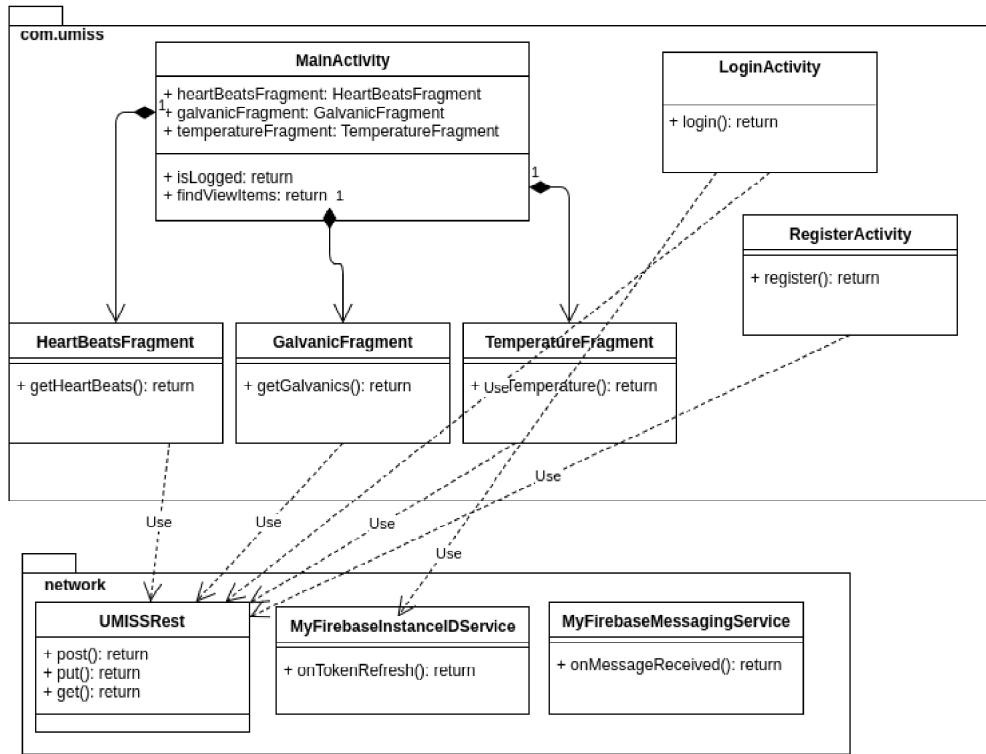


Figura 10 – Diagrama de classes do módulo Android.

3 Controle e Alimentação

4 Estruturas

5 Atividades

- **Afonso Delgado**

- **Dylan Guedes**

Sou aluno de Engenharia de Software, e fiz parte do subsistema de Processamento de Sinais e Monitoramento. Atuei principalmente nas contribuições do servidor *frontend* (feito em Ember.JS), e no código do *middleware*, onde criei uma aplicação chamada Shoelace. O Shoelace apresenta uma arquitetura que permite a adição de novos sensores de uma maneira extensível, e se comunica com o servidor *backend* para o registro de sinais. Junto com os outros membros do subsistema desenvolvemos estratégias para que não fossem mandados dados redundantes, diminuindo o consumo de banda.

- **Gustavo Cavalcante**

- **Tiago Assunção**

- **Wilton Rodrigues**

6 Considerações Finais

Com o UMISS, esperamos criar uma solução de qualidade e acessível para um público alvo que é, constantemente, deixado de lado. O UMISS ainda permite que os responsáveis pelos pacientes não precisem estar observando-o a todo o tempo; dessa forma, o projeto UMISS ainda pode proporcionar maior grau de liberdade aos cuidados dos pacientes, e possivelmente melhorando as relações entre cuidadores e pacientes.

Referências

TECHNOLOGY, C. *Arquitetura do servidor Django*. [S.l.], 2013. Disponível em: <<http://criticaltechnology.blogspot.com.br/2011/09/mvc-in-three-tier-architecture.html>>. Citado 2 vezes nas páginas 5 e 16.

Anexos

ANEXO A – Diagrama de Classes Backend

Diagrama de classes do backend na figura 11, desenvolvido no projeto UMISS.

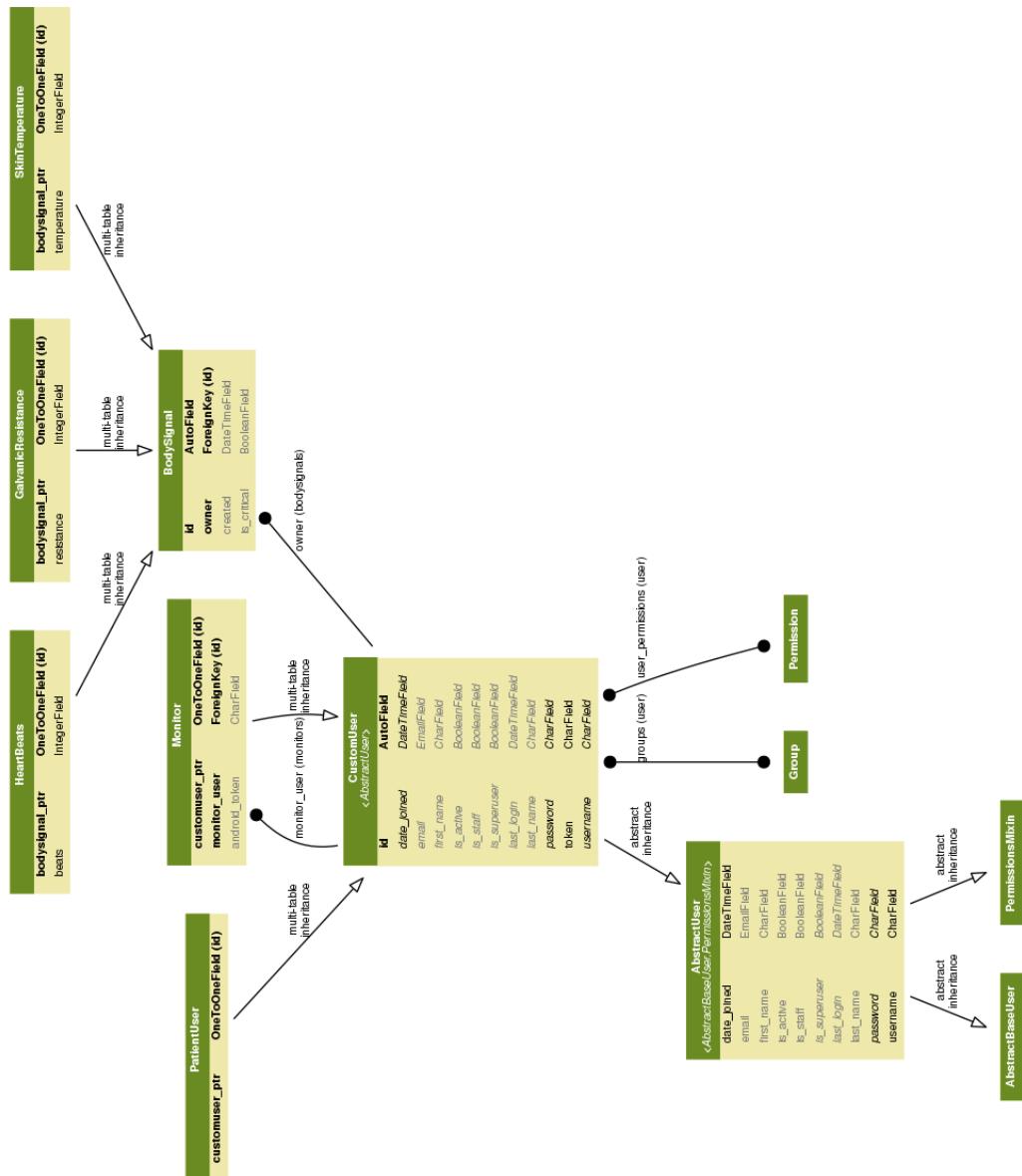


Figura 11 – Design e Arquitetura do servidor Django