UNIVERSIDADE FEEVALE

FELIPE FERNANDES LORENZONI

CAMADA DE COMUNICAÇÃO ENTRE FRONT-END E BACK-END DO PROJETO HEALTH SIMULATOR

Novo Hamburgo

2017

FELIPE FERNANDES LORENZONI

CAMADA DE COMUNICAÇÃO ENTRE FRONT-END E BACK-END DO PROJETO HEALTH SIMULATOR

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel em
Sistemas de Informação pela
Universidade Feevale

Orientador: Paulo Ricardo Muniz Barros

Coorientadora: Marta Rosecler Bez

Novo Hamburgo

2017

AGRADECIMENTOS

Dedico a todos que, de alguma maneira contribuíram para realização deste trabalho, em especial:

À minha mãe Kathia, que me proporciono estar realizando este objetivo.

À minha namorada Tanara, por estar ao meu lado, pela sua paciência e compreensão ao longo deste trabalho.

Ao grupo de pesquisa Computação Aplicada que sempre apoio e ajudou de alguma forma.

Aos meus orientadores, Marta Rosecler Bez e Paulo Ricardo Munis Barros, os quais me ajudaram a escolher o melhor caminho sempre.

E em memória de meu pai Álvaro.

RESUMO

O ensino na área da saúde vem se beneficiando de forma expressiva de aplicações que simulam casos clínicos. O Health Simulator é um simulador do tipo Paciente Virtual que pode ajudar a diminuir a lacuna que existe entre o aprendizado teórico e a grande quantidade de informações que o aluno deve absorver. O simulador é separado em duas partes: o *front-end*, que contempla o aplicativo do jogo e o *back-end*, que abrange o servidor que armazena as informações trocadas entre o jogo e o servidor. O objetivo principal deste trabalho é desenvolver a integração entre estas duas partes. O meio de comunicação que está sendo desenvolvido se utiliza de um Serviço Web que é comumente utilizado para soluções complexas de comunicações entre diferentes aplicações, que dispõem de uma solução que visa obter maior escalabilidade e que diferentes plataformas possam se beneficiar desta solução. Desse modo, optou-se por criar uma arquitetura baseada em REST (*Representational State Transfer*) entre as aplicações cliente (jogo/cliente) e um fornecedor do serviço (*back-end*). Uma vez estipulado que o sistema será baseado numa arquitetura do tipo REST, este deve obedecer algumas restrições: implementar o tipo de comunicação cliente-servidor, ser *stateless*, entre outras. Desta forma, o trabalho proposto visa implementar um modo de realizar a integração e comunicação entre as partes do simulador, buscando como um dos objetivos a escalabilidade do projeto, ou seja, estar disponível para o maior número possível de alunos.

Palavras-chave: *Health Simulator. REST, Web Service,* Escalabilidade, Simulação.

ABSTRACT

Health education has benefited significantly from applications that simulate clinical cases.The Health Simulator is a Virtual Patient type of simulator that can help reduce the gap between theoretical learning and the vast amount of information that the student must absorb. The simulator is separated into two parts: the front-end, which includes the game application and the back-end, which covers the server that stores the information exchanged between the game and the server. The main objective of this work is to develop the integration between these two parts. The communication medium that is being developed uses a Web Service that is commonly used for complex communications solutions between different applications, which have a solution that aims at greater scalability and that different platforms can benefit from this solution. Thus, it was decided to create an architecture based on REST (Representational State Transfer) between the client applications (game / client) and a service provider (back-end). Once established that the system will be based on a REST type architecture, it must obey some restrictions: implement the type of client-server communication, be stateless, among others. In this way, the proposed work aims to implement a way to perform the integration and communication between the parts of the simulator, seeking as one of the objectives the scalability of the project, that is, be available to as many students as possible.

Key words: *Health Simulator, REST, Web Service*, Scalability, Simulation.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de Dados
CA	Computação Aplicada
E-R	Entidade-Relacionamento
HTTP	*Hypertext Transfer Protocol*
URI	*Uniform Resource Identifiers*
MIT	*Massachusetts Institute of Technology*
PV	Paciente Virtual
RB	Redes Bayesianas
REST	*Representational State Transfer*
STP	Sistema Toyota de Produção
SUS	Sistema Único de Saúde

SUMÁRIO

INTRODUÇÃO

O modo como o ensino na área da saúde é transmitido para os alunos nem sempre consegue atingir as “Diretrizes Curriculares Nacionais do Curso de Medicina”. Segundo Bez (2013), o currículo e o método pedagógico que se espera devem permitir aos alunos o desenvolvimento da capacidade de observar e de escutar. Com isso, é importante tornar o estudante preparado para pensar e a aprender, ser, fazer e conviver com a sua aprendizagem. Tsuji e Silva (2010) evidenciam que os estudantes necessitam praticar desde o início do curso, efetuando atividades e tarefas de nível de dificuldade crescente com o passar do tempo nele.

De acordo com Maroni (2013), a utilização de procedimentos para simulação de cenários da prática clínica, com o objetivo de educar, vem a ser uma ferramenta adicional ao ensino. Tratando-se destas simulações, técnicas que reproduzem o cenário real por meio de equivalências (OED, 2012), provêm a reprodução de ambientes e situações clínicas.

Ainda segundo Maroni (2013), realizar simulações de casos clínicos com um nível de complexidade alta, apresentam-se difíceis de serem obtidos no ambiente real para a finalidade de estudos. Desta maneira, simuladores demonstram vantagem significativa, oportunizando aos alunos vivenciar situações de crise antes que elas ocorram na prática clínica (STANFORD, 2010).

Simulação vem sendo definida como o método de imitar o comportamento de uma situação ou procedimento (BRADLEY, 2006). Isto é, a representação simplificada de um quadro real, acontecimento ou método (BEZ, 2013).

A simulação computacional compreende um programa com um modelo de um sistema, podendo ser natural ou artificial, ou um processo (JONG E JOOLINGEN, 1998). Pode incluir revisão de teoria, criação de perguntas, elaboração de tese ou acervo de dados (BEZ, 2013) com a finalidade de estudo ou formação de pessoal. (BRADLEY 2006).

Segundo Akpan (2001), a simulação por meio do computador tem uma grande capacidade no aperfeiçoamento do ensino e na melhora da aprendizagem. O conhecimento é adquirido através da interação com a simulação, por meio de uma atividade. Entre as razões para o seu uso estão as considerações éticas, como em testes com animais, redução de custos e tempo, sondagem de hipóteses e a

possibilidade de refazer os experimentos. Também apresentam gráficos, diagramas, animações e vídeos para melhorar o entendimento do assunto (BLAKE E SCANLON, 2007). Assim, o aluno tem a oportunidade de executar sozinho uma diversidade de situações que representam a “vida real” e os dilemas que possa enfrentar no futuro (AKPAN, 2001).

As simulações aplicadas na área da saúde, do tipo Paciente Virtual (PV) possibilitam aos educadores domínio absoluto do quadro clínico escolhido previamente, e permitem ao aluno a obtenção da história clínica e exames para a formação de diagnóstico e decisão de tratamento (Orton e Mulhausen, 2008). Servem como uma opção ao paciente real. Tem como objetivos ajudar nas decisões, gerir doentes, reduzir erros durante o tratamento, melhorar os cuidados médicos e a segurança do paciente (ZIV et., al. 2005) e aumentar a compreensão das doenças e tratamentos (BASS, 2006).

Utilizando como interface de comunicação o PV, os professores podem criar situações clínicas diversas, reais ou não, que contemplem todos os temas necessários para o aprendizado. É possível, com isso, desenvolver aplicativos que simulem um ambiente e contexto plausível para o aluno estudar (BEZ, 2013).

Conforme Ziv et al. (2005), um simulador voltado ao ensino, na área da saúde, pode ser lido como um conjunto de ferramentas que proporciona aos professores total domínio em cenários clínicos pré-estabelecidos. Destaca-se a importância do educador possuir um modo de elaborar suas próprias simulações dentro de ambientes monitorados. Pois, diferentemente de situações cotidianas, é possível saber quais variáveis estão sendo alteradas e o porquê desta alteração. Além disso, provê total controle ao professor com *feedbacks* em tempo de execução, verificação de pontos que poderiam passar sem serem notados, visando alcançar um resultado satisfatório.

Tendo em vista prover aos educadores da área da saúde um simulador fiel ao ambiente real vivenciado no dia-a-dia de hospitais e clínicas, na Universidade Feevale, está em desenvolvimento o simulador *Health Simulator*. Este é divido em duas partes devido a sua complexidade, o *front-end*, ambiente ao qual o aluno terá acesso aos casos elaborados pelo seu professor, e o *back-end*, site administrativo onde o professor terá acesso para criação dos casos de estudos.

Devido a necessidade do Health Simulator estar disponível em cenários e plataformas diversas, como por exemplo: dispositivos móveis, navegadores diversos,

em salas de aulas e em instituições de ensino diferentes. Existe a necessidade de elaborar uma camada de comunicação robusta que possibilite atender estas variações.

Além dos diversos cenários em que o Health Simulator visa alcançar, outro ponto importante que dever ser considerado é a singularidade entre as linguagens de programação escolhidas para o desenvolvimento do front-end e do back-end. O front-end faz uso da tecnologia denomina *Unity*, a qual é ferramenta de desenvolvimento que possibilita a criação de jogos para navegadores assim como a criação de jogos para consoles de videos *games (UNITY,2016)*. Já o back-end é desenvolvido em C# ASP.Net, além disto é utilizado um framework denominado MVC (*model view controler*).

Tendo em mente as necessidades/restrições do projeto Health Simulator, fica evidente a carência por uma camda de comunicação robusta, a qual será responsável pela troca de informaçoes de dados, entre os dois ambientes do projeto. Portanto, a plataforma de comunicação que está sendo desenvolvida faz uso de um Serviço Web (SOMMERVILLE, 2011) que possibilita uma solução com maior escalabilidade. Desse modo, optou-se por aplicar princípios da arquitetura REST (*Representational State Transfer*) entre a aplicação cliente (*front-end*) e um fornecedor do serviço (*back-end*) (SOMMERVILLE, 2011).

Deste maneira, este trabalho tem como objetivo geral, a construção da camada de comunicação entre as duas partes do projeto. E como objetivos específicos foram definidos como: Integrar o front-end com o back-end do projeto Health Simulator; Explorar e corrigir falhas de segurança que possam existir entre a comunicação do front-end com o back-end; Validar a comunicação entre o back-end e o front-end.

A organização deste trabalho é apresentada inicialmente com esta introdução e mais quatro capítulos. No primeiro será abordado o tema *Health Simulator*, onde será apresentada a criação de personagens e cenários para um simulador crível e fidedigno aos alunos, além do *back-end* do projeto, que contempla o ambiente administrativo, onde o professor possui acesso para cadastro de casos de estudos. No segundo capítulo são abordadas as características de uma arquitetura de sistema do tipo REST, bem como, dois pontos referentes a segurança e a escalabilidade. No terceiro capítulo é apresentada a camada de comunicação desenvolvida no presente trabalho. No quarto capítulo será demonstrada a validação

da camada de comunicação fazendo uso de testes unitários, juntamente com os resultados obtidos.

1 HEALTH SIMULATOR

O modo como o ensino na área da saúde é transmitido para os alunos nem sempre consegue atingir as “Diretrizes Curriculares Nacionais do Curso de Medicina”. Segundo Bez (2013), o currículo e o método pedagógico que se espera devem permitir aos alunos o desenvolvimento da capacidade de observar e de escutar. Com isso, tornar o estudante preparado para pensar e a aprender, ser, fazer e conviver com a sua aprendizagem.

Aspirando diminuir a lacuna que existe entre o aprendizado teórico e o grande volume de informações que este deve absorver, foi criado, na universidade Feevale, um grupo de estudos e pesquisa intitulado de Computação Aplicada (CA). Nesta equipe está sendo desenvolvido um projeto denominado Health Simulator.

Este projeto visa a implementação de um simulador do tipo paciente virtual. Segundo Orton e Mulhausen (2008, p. 75), um PV é um *software* interativo que cria uma simulação de um cenário clínico real, que proporciona o conhecimento de práticas profissionais da saúde, obtendo prontuários clínicos, exames e realizando análises e resoluções terapêuticas.

O Health Simulator, devido a sua complexidade e diferentes arquiteturas, é divido em duas partes: o *front-end*, que contempla o aplicativo do jogo e o *back-end*, que abrange o servidor que armazena as informações trocadas entre o jogo e o servidor. Desta forma, é necessária uma estrutura de comunicação entre o *front-end* e o *back-end*, que será responsável pela troca de informações entre os dois ambientes.

No projeto, o *front-end* diz respeito a parte do simulador que será designado aos alunos da área da saúde. Este será apresentado na forma de um jogo sério, com cenários associados a espaços de atendimento à saúde e personagens que possibilitam a representatividade de profissionais da área da saúde e de pacientes.

Além disso, foi prevista uma construção ampla de elementos de arte no projeto, desse modo, foi implementada uma metodologia simples, visto que esta foi associada aos princípios e filosofia ágil de desenvolvimento de *software*. Estes princípios e metodologias de desenvolvimento serão abordados a seguir.

Atualmente, o projeto *Health Simulator* possui quatro categorias de personagens (médicos, pacientes, dentistas e enfermeiros), e estas categorias possuem os gêneros, masculino e feminino. Além disso, os personagens são

desenvolvidos com diversidade etária (bebê e criança (somente para pacientes), jovem, adulto e idoso) e étnica (caucasiano, oriental, afrodescendente e indiano). No momento atual foram finalizados 131 personagens do gênero masculino e 81 do feminino.

O método escolhido para a construção dos personagens utilizados no projeto *Health Simulator* é baseado em técnicas de modelagens intitulada de *Face-by-Face* para as personagens. A criação dos polígonos dos modelos foi desenvolvida um a um (Gomes et al., 2016). No desenvolvimento dos modelos tridimensionais foi utilizado o método de Ward (2008), adequado para as práticas de desenvolvimento ágil. Juntamente com esta prática, foram apreciados os princípios e a filosofia do *Lean Thinking* (Pensamento Enxuto), privilegiando o profissional responsável pelo desenvolvimento dos modelos e o esforço empregado no projeto como um todo. E como motor de jogo foi adotado a Unity3D *Engine*, devido a sua interface simplificada, a qual ajuda no aprendizado e na manipulação, assim como, proporciona a geração de arquivos executáveis em múltiplas plataformas.

1.1 MÉTODOS AGEIS

Em referência a métodos ágeis de desenvolvimento, Teles (2006, p. 31) define que “o termo ‘desenvolvimento ágil’ faz referência ao desenvolvimento iterativo, em espiral”, que é realizado de maneira incremental e que possibilita uma maior liberdade na transição entre as fases do desenvolvimento.

Para o desenvolvimento de modelos tridimensionais, pode ser adotado o método de Ward (2008), adaptado para o formato de desenvolvimento ágil. Neste formato, é levado em consideração a filosofia e princípios do Pensamento Enxuto (*Lean Thinking*), priorizando-se o cliente (entendido aqui, como sendo o profissional que desenvolve os modelos) e o trabalho focado no projeto como um todo, não suas partes isoladas apenas.

Durante o processo são realizados testes dentro da Unity3D, no intuito de validar o modelo desenvolvido e prever futuros erros e inconsistências no ambiente do simulador. Caso for obtido um resultado positivo dos testes realizados, este modelo é encaminhado para a próxima etapa do processo, que mantém a dinâmica de trabalho. Para todo início, meio e fim de uma etapa, adota uma sequência de

desenvolvimento e estas sequências são incrementais, isto é, o modelo recebe mais detalhes.

1.2 ESTUDO DE PERSONAGENS

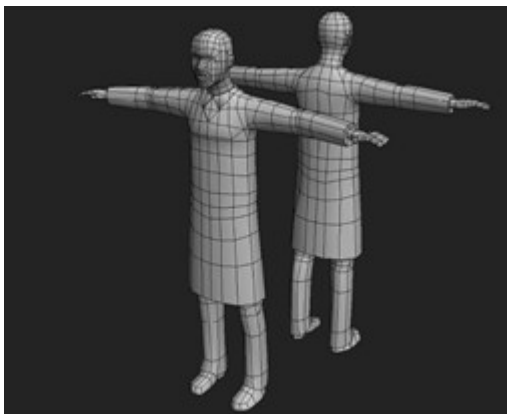
Na criação de modelos realistas para jogos é preciso prezar pelas características fundamentais. Conforme Fox (2004), para o desenvolvimento de um modelo 3D complexo, como uma parte do corpo ou a cabeça do personagem, é necessário realizar um trabalho de grade de malha ou uma topologia, para então ser possível aplicar a arte desejada.

Sendo assim, são realizadas três etapas, divididas entre a criação do *moodboard*, a modelagem tridimensional e a texturização dos personagens. *Moodboard* são pranchetas onde é apresentado um estudo de variações possíveis para cada personagem a ser desenvolvido. Pode-se citar que a etapa de criação do *moodboard* é onde se realiza um estudo de características dos personagens, para nas etapas de modelagem e texturização realizar a criação virtual. Uma prancha de *moodboard* é exemplificada na Figura 1.

Figura 1 - Moodboard Médico

Fonte: Heckel (2015)

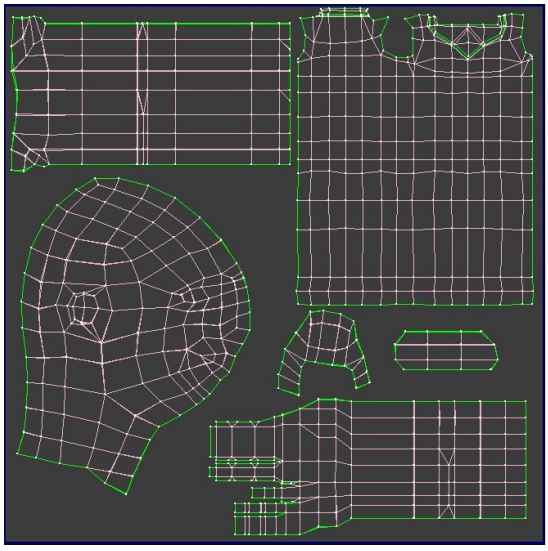
Figura 2 - Exemplo de modelo 3D



Fonte: Heckel (2015)

A Figura 2 exemplifica a modelagem de um modelo da classe médica, do gênero masculino. Para a criação dos mapeamentos para a concepção das texturas dos modelos, é estabelecido um layout UV de mapeamento dos modelos (disposição do mapeamento), para futuro revestimento em texturas. A Figura 3 exemplifica uma disposição retilínea do layout UV do modelo exibido na Figura 2.

Figura 3 – *Layout* de modelo 3D



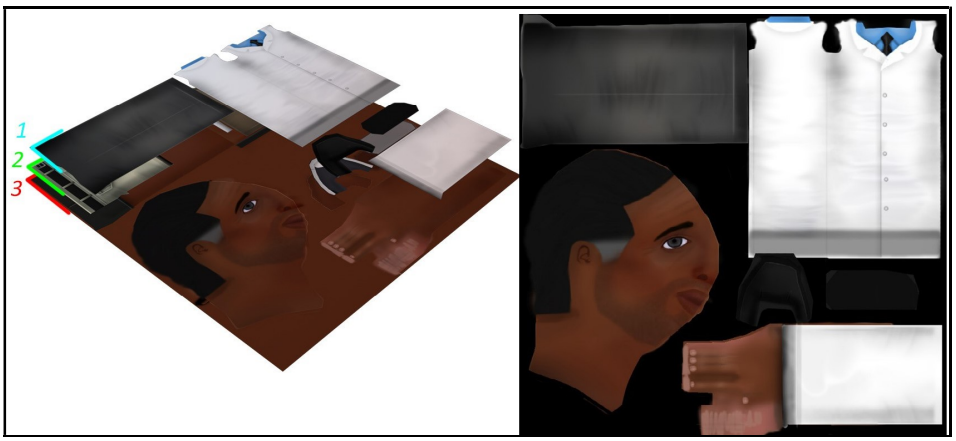
Fonte: Heckel (2015)

Segundo Heckel et al. (2015), a escolha para a produção das texturas foi realizada de maneira a obter uma estética mais limpa, devido a análise durante o projeto. Com esta escolha obtém-se uma textura com característica mais suaves.

Ainda conforme o mesmo autor, a criação das texturas para os personagens foi apoiada na metodologia demostrada por Appleby e Greveson (2011) do estúdio *Splash Damage*, que se compõe da sobreposição de camadas uma sobre as outras,

sendo vinculadas somente quando existir a necessidade. Sendo assim, possui diversas camadas acima com diferentes vestes e uma camada inferior com a pele do modelo. Isso é exemplificado na Figura 4.

Figura 4 - Arquivo de textura do modelo 3D



Fonte: Heckel (2015)

Conforme Heckel et al. (2015), foi estabelecida uma organização UV e normatizada para todos os modelos de personagens. Diante disso, foi discernido em camadas diferentes cada peça ou conjunto de roupas, para futuramente serem constituídas na *engine* do jogo. A Figura 5 ilustra o modelo do personagem montado, já com sua modelagem e seu *layout* UV estabelecido e textura difusa.

Figura 5 - Modelo 3D com Textura Difusa

Fonte: Heckel (2015)

1.3 ESTUDO DE CENÁRIOS

A formação de um cenário é um agrupamento de vários elementos que, em consonância, proporcionam um ambiente que segue a função à qual lhe foi proposta. Dessa maneira, no desenvolvimento de um ambiente específico, é necessário notar o espaço que lhe é apresentado e, a partir dele, elaborar um *design* que seja viável com a ideia sugerida. Conforme Gurgel (2012, p. 28), o *designer* deve criar formas que possam suprir as necessidades exigidas por determinadas funções ou tarefas. Consequentemente, é primordial que a atribuição do ambiente seja clara e objetiva. Nesse processo de produção, conforme Gurgel (2012, p. 21), é ponderado também um elemento crucial, o *design*, que como compreendido hoje, é um processo sagaz e premeditado que busca dispor materiais (com suas linhas, texturas e cores) e diversas formas com o intuito de atingir um determinado objetivo, funcional ou estético.

Como característica primordial, o *design* proporciona a organização e a funcionalidade total do meio que deve ser desenvolvido. Isto é, um projeto adequado é aquele que contém um bom *design*, o qual alcança um resultado funcional e harmônico, além de apresentar uma decoração de um ambiente que contemple características visuais que são apresentadas. A decoração abrange elementos como a iluminação e as cores do ambiente, que possibilita diversas impressões, e a climatização do local, além da distribuição e da seleção dos melhores elementos decorativos ou mais adaptáveis com a proposta do ambiente.

As cores, vitais na decoração, exercem uma forte influência no conforto e nas impressões nos ambientes. Conforme Gurgel (2012), a atribuição das cores é utilizada para influenciar o estado de espírito do usuário, criando atmosferas distintas, ajustando as proporções do ambiente, corrigindo imperfeições arquitetônicas, modificando a temperatura e valorizando centros de interesse. No processo de decoração de ambiente, é fundamental uma iluminação que trabalhe bem com o ambiente que está sendo proposto.

A iluminação de um ambiente possui grande significado durante o desenvolvimento do projeto, sendo um elemento cativante quando articulado com outros componentes do cenário (GURGEL, 2011). Ela pode atuar na emoção, na psique, no humor, no estado de espírito. Em complemento, o mesmo autor relata ser fundamental a iluminação para efeitos particulares e distintos. A iluminação, além

disso, possui outras funções além de iluminar, pois ela pode realçar elementos e criar pontos de interesse ou diferentes atmosferas. É útil, ainda, para dar maior sensação de aconchego, alegrar, entristecer, estimular ou acalmar os sentidos. Portanto, é importante a combinação da coloração e da iluminação. O conjunto é que permite esta gama de sensações ao jogador.

Por meio dos conceitos apresentados, é necessário dedicar uma quantidade grande de atenção no momento de projetar e desenvolver um ambiente tridimensional, com a intensão de que o jogador se sinta aconchegado e que o ambiente difunda uma impressão de realidade almejada. Considerando esses conceitos, adquire-se um norte para a produção de cenários hábeis de envolver o jogador.

Assim como no estudo de personagens foram idealizadas e criadas as pranchas de *moodboard* para os cenários. Por tanto, antes da criação das pranchas, os cenários foram separados por classes sociais e ambientes, onde é mais habitual o atendimento de médicos, assim sendo, consultórios, hospitais e residências, conforme ilustrado na Figura 6.

Figura 6 - Moodboard

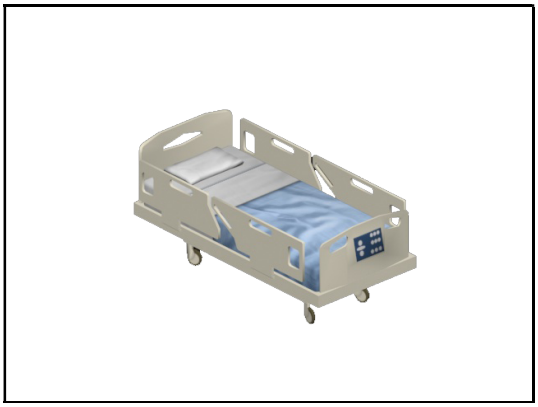
Fonte: Morche (2015)

Com as pranchas de *moodboards* definidas, é possível desenvolver os a*ssets. Asset* é uma representação de qualquer objeto que pode ser utilizado em um projeto ou jogo. Este objeto pode ser gerado a partir da *Unity* (ferramenta de

desenvolvimento de jogos) como um modelo 3D, uma imagem, ou qualquer outro arquivo que é suportado pela *Unity (UNITY,2016)*. No contexto do projeto Health Simulator, *assets* são tratados como objetos do cenário, como, por exemplo: macas, cadeiras, equipamentos médicos, entre outros.

No decorrer do processo de modelagem, as malhas tridimensionais dos *asstes* são desenvolvidas. Segundo Fox (2004), são objetos em 3D constituídos por porções geométricas estabelecidas como: vértices, que são pontos no espaço tridimensional, bordas, que são as conexões entre os vértices, e planos, que são constituídos de bordas e pontos, os quais são intitulados de polígonos. A Figura 7 exemplifica um *asset* após realizado o processo.

Figura 7 – Maca de leito hospitalar



Fonte: Asset criado no projeto Health Simulator

A etapa de texturização é dada pelo revestimento dos modelos pela textura, isto é, o procedimento que definirá as tonalidades do modelo, assim como outras características, por exemplo, o relevo, a transparência ou o reflexo. Conforme Novack (2010), a aplicação da textura contempla a concepção de texturas superficiais 2D (por exemplo, roupas e pele), denominadas como mapas de texturas, que os modeladores aplicam às malhas de fios 3D.

1.4 BACK END

O *back-end* do projeto Health Simulator é dividido em três estágios, que são: modelagem do conhecimento, interface de administração e serviço *web* de comunicação. O primeiro estágio, modelagem do conhecimento, compreende o conhecimento a ser evidenciado pelo especialista. Para tal, fez-se uso de uma diretriz clínica, um modo sistêmico de orientação e definição do conteúdo a ser

produzido a partir de evidências, particularidades fundamentais para um *software* educacional com foco na área da saúde (LIMA et al., 2015). Sendo assim, foi preciso criar um modelo estatístico de representação do conhecimento, utilizando-se de uma rede bayesiana (RB). Esta é uma das possíveis abordagens que são reconhecidas e que auxiliam na tomada de decisões na área da saúde (LIMA et al., 2015). A base de conhecimento é armazenada nesta rede, que futuramente servirá como um guia para a estruturação dos casos clínicos. Sendo assim, são definidas as variáveis para a criação da rede, como diagnósticos e sintomas e, por fim, são geradas as relações das probabilidades entre cada variável.

O segundo estágio, é denominado de Interface administrativa, que é uma interface web em processo de desenvolvimento. Neste estágio, o professor tem acesso as variáveis que foram armazenadas na rede, como exames físicos e complementares, possíveis históricos anteriores do paciente, entre outros. A partir da escolha das variáveis, o sistema apresentará o possível resultado para o caso clínico, com os diagnósticos e condutas, gerados de acordo com as probabilidades difundidas na rede. Deste modo, os métodos e a criação de cada caso clínico é facilitada. Cada caso clínico gerado é armazenado em um Banco de Dados (BD) com as informações relevantes a modelagem do caso.

O último estágio, Serviço Web de Comunicação, recebe os casos clínicos armazenados no BD e expõe para os alunos no formato de jogo. O desenvolvimento dos casos clínicos é baseado em uma das soluções mais adotadas para realizar a comunicação entre aplicações distintas e integração de sistemas, provendo uma maior consistência entre diferentes plataformas de desenvolvimento de software (SOMMERVILLE, 2011). A escolha por utilizar Serviços Web possibilita a criação de um padrão para trocar de mensagens entre aplicações clientes (jogo) e um provedor de serviço (site administrativo).

1.4.1 Análise do sistema

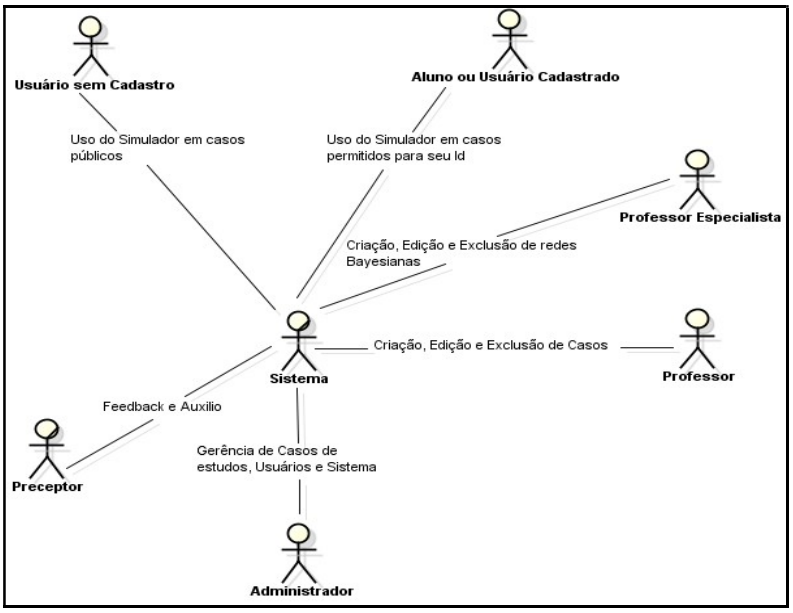
A catalogação dos requisitos para o desenvolvimento da interface administrativa do *back-end* foi realizada de forma presencial, com encontros semanais. Estes requisitos são a parte fundamental do projeto, pois neles estão as características que devem ser apresentadas para o professor em formato digital.

1.4.1.1 Diagrama de caso de uso Health Simulator

Um diagrama de Casos de Uso possui o propósito de facilitar a comunicação entre os analistas e os clientes de um projeto. O diagrama representa um cenário que expõe as funcionalidades de um sistema, tendo como o ponto de vista o usuário. O cliente deve ser capaz de visualizar no diagrama de Casos de Uso as funcionalidades essências de seu sistema (FOWLER, 2004).

Neste subcapítulo será exposto o caso de uso geral do Health Simulator. Na Figura 8 é demostrado como os principais atores envolvidos estão associados a suas atividades. A atividade de liberar permissões a usuários e a casos de uso é de responsabilidade do administrador. A criação, edição e exclusão de casos de estudos, bem como as suas informações, é de responsabilidade dos professores. Assim como a criação, edição e exclusão de redes bayesianas, é de responsabilidade dos professores especialistas. Os usuários cadastrados ou alunos, por sua vez, fazem uso do simulador como jogo, para os casos de estudos autorizados para o seu usuário e casos públicos cadastrados. Usuários não cadastros, podem apenas jogar casos públicos. Todos os atores aqui citados, podem fazer uso do simulador como jogadores. O *feed*back do sistema e auxílio aos jogadores é realizado pelo preceptor, ou um agente que interage com o aluno de acordo com sua conduta no jogo.

Figura 8 - Diagrama de Caso de Uso



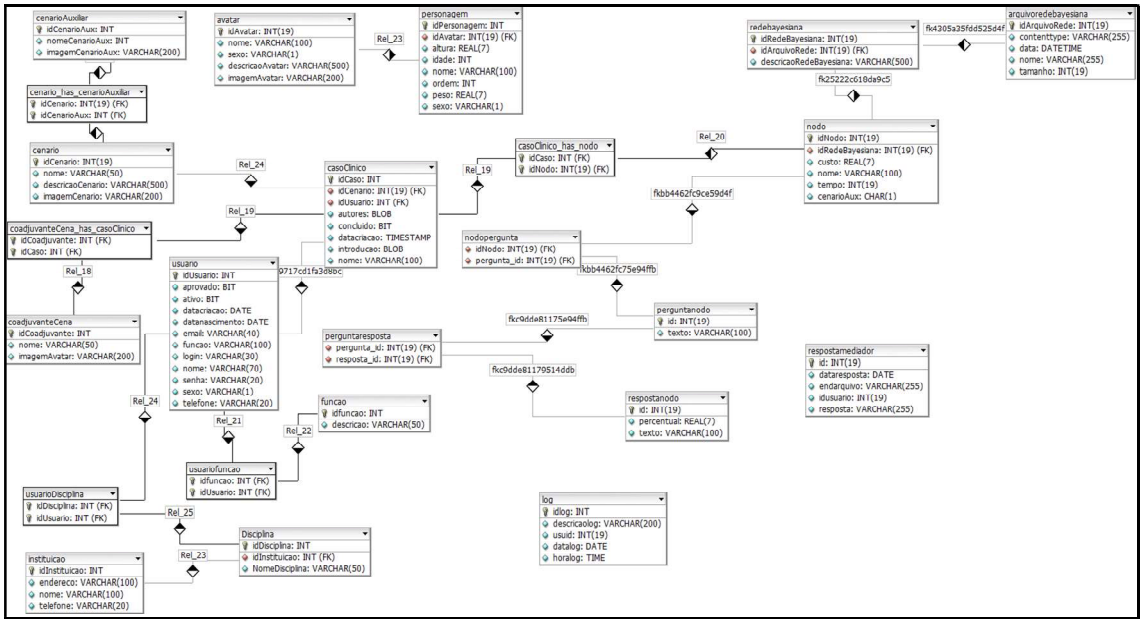
Fonte: Diagrama de Caso de Uso do Health Simulator

1.4.1.2 Modelo ER do Health Simulator

O Modelo Entidade-Relacionamento é a metodologia para a construção de modelos conceituais, os quais tem como base a compreensão do âmbito do cenário como um agrupamento de objetos básicos, denominados entidades e o seu relacionamento entre outras entidades (SILBERSCHATZ et al., 1999). O diagrama de Entidade-Relacionamento (E-R) é gerado a partir da modelagem conceitual do modelo E-R, ou seja, o modo gráfico para realizar a representação deste modelo.

O E-R da Figura 9 demostrada as entidades do simulador *Health Simulator*, porém, de forma reduzida do modelo completo.

Figura 9 - Modelo E-R



Fonte: Modelo E-R *Health Simulator*

1.4.1.3 Decupagem

Segundo Aumont e Marie (2006), decupagem pode ser definida uma estrutura de um filme, a qual é separada com planos e de sequências. No âmbito do Health Simulator, a decupagem do simulador é complementada com informações sobre o que irá ocorrer em determinado plano e sequência, conforme consta no Apêndice A. Por tanto, com base neste documento, foi realizado um levantamento de requisitos de métodos a serem desenvolvidos, com a finalidade de realizar a comunicação entre ambas as partes deste projeto.

1.4.2 Interface administrativa

A interface administrativa é o local onde o professor possui acesso a variáveis como sintomas e sinais (salvos na rede), provável história clínica do paciente, assim como, exames adicionais e físicos. A partir da escolha destas variáveis, o simulador apresentará o resultado do caso clínico com os diagnósticos e procedimentos, baseado nas probabilidades semeadas na rede. Desta maneira, é simplificado o processo e a criação de cada caso clínico.

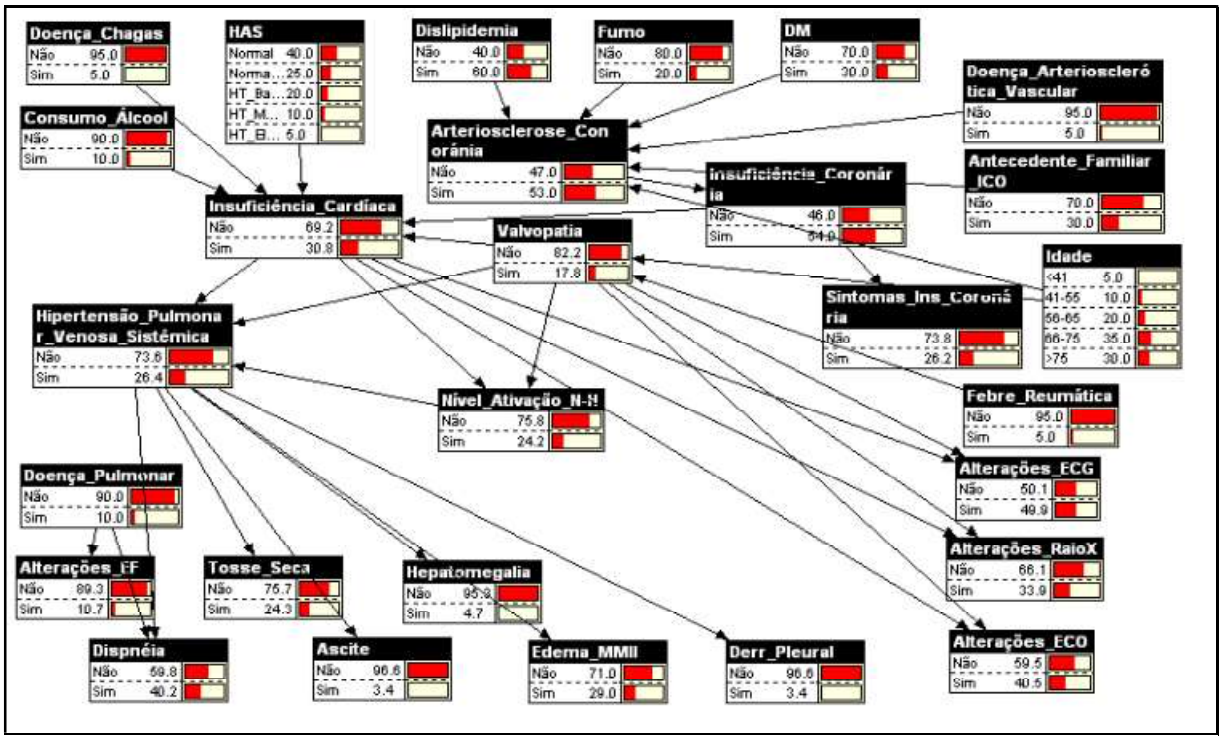
Os casos criados pelo professor serão salvos no Banco de Dados (BD) em conjunto com as informações relacionadas a modelagem do caso. No próximo subcapítulo será tratado sobre rede Bayesiana e qual a usa finalidade neste simulador.

1.4.3 Rede bayesiana

Redes Bayesianas podem ser classificadas como modelos representativos do conhecimento incerto, fundamentados no Teorema de Bayes (PINHEIRO, 2015). Esta rede é um modelo estatístico de representação do conhecimento, sendo umas das abordagens admitida na escolha de decisões na área da saúde (HIGGS et al., 2008).

Modelos estes que possuem uma característica qualitativa, demostrado por um grafo acíclico que designa as relações causais entre as variáveis do domínio, e outra característica quantitativa, representada por valores de probabilidades que sistematizam a incerteza quanto a essas relações causais (PINHEIRO,2015). Ao utilizar redes bayesianas, o conhecimento do especialista é agregado ao modelo de dados. Desta forma, é possível realizar inferências com os dados desconhecidos, os quais, são fornecidos pelas interações realizadas no simulador. Na Figura 10 é exemplificada uma rede bayesiana.

Figura 10 - Rede Bayesiana

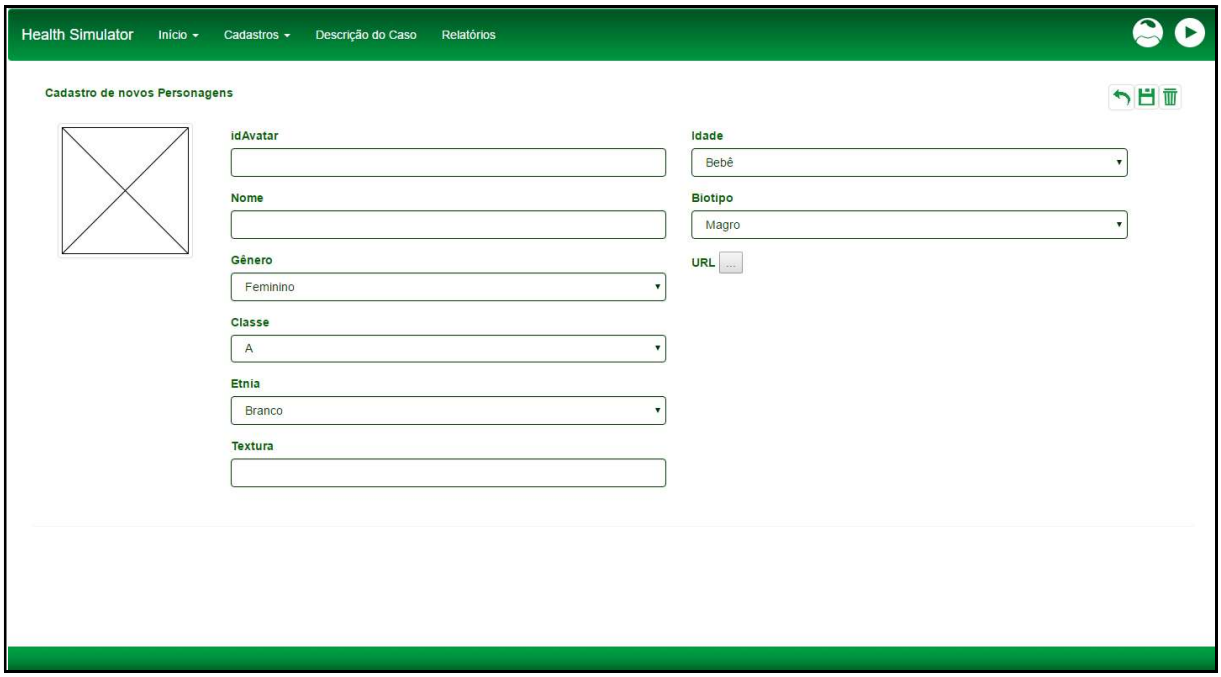


Fonte: Saheki (2005)

1.4.4 Casos de estudos

A construção dos casos de estudo é feita através da interface administrativa. Nesta, são realizados o cadastro de dados clínicos, o cadastro de usuários, a escolha do(s) cenário(s) (consultórios, hospital, hospital do Sistema Único de Saúde - SUS), da mesma maneira que o castrado do Personagem Paciente. Além disto, é crucial a escolha da rede bayesiana que será associada ao caso. O responsável por essas e demais ações realizadas na interface administrativa é do professor. Na Figura 11 é demostrando o menu de cadastro de personagens do *Health Simulator*.

Figura 11 - Cadastro de Novos Personagens do *Health Simulator*



Fonte: Elaborado Pelo autor

Para realizar o armazenamento de casos de estudos gerados no *Health Simulator*, é utilizado um banco de dados (Microsoft SQL Server - MSSQL). Assim sendo, estes casos estarão disponíveis futuramente para a reutilização por alunos no formato de jogo digital. No próximo serão aprofundados os conceitos, as características, restrições e necessidades para se implementar uma comunicação entre sistemas distintos utilizando REST.

2 REST

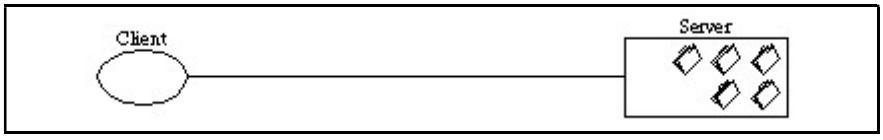
Representational State Transfer (REST) ou transferência de Estado Representacional, é um modelo de arquitetura de *software* para soluções de serviços *web (*DEITEL, DEITEL, 2011). Projetos que optam por utilizar uma arquitetura REST são denominados RESTful (RICHARDSON; RUBY,2007). A construção de um sistema que utiliza o REST, como modelo de arquitetura, deve atender as restrições estipuladas dentro da arquitetura, que serão abordadas nos subcapítulos seguintes.

Um serviço web RESTful, se equiparado com protocolos de serviço web (*Simple Object Acess Protocol* e *Web Services Definition Language*), possui a vantagem de aproveitar um número maior de recursos que contemplam o próprio protocolo HTTP (*Hypertext Transfer Protocolo*), como códigos de resposta, cabeçalhos e verbos (WEBBER; PARASTATIDIS; ROBINSON, 2010). Dessa forma, é simplificada a solução e existe a possibilidade de diminuir a quantidade de dados trafegados, uma vez que existirá um nível a menos de abstração na implementação da comunicação.

2.1 CLIENTE-SERVIDOR

Determinado que o sistema seguirá o modelo de arquitetura REST, este deve atender a algumas restrições para alcançar uma alta escalabilidade no serviço (FIELDIN, 2000). Na arquitetura *REST,* o modelo de troca de informações a ser utilizado é do tipo cliente-servidor (FIELDING, 2000). Neste modelo, o cliente iniciará uma ação de requisição de uma tarefa disponibilizada pelo servidor. O servidor, por sua vez, receberá a tarefa e devolverá para o cliente o seu resultado (TANENBAUM; STEEN, 2006). Na Figura 12 é exemplificada esta troca de informação entre o cliente e o servidor.

Figura 12 - Comunicação entre Cliente e Servidor



Fonte: Fielding (2000)

Neste modelo há uma distribuição clara da responsabilidade de cada lado. O servidor é responsável por prover e manter as informações aos clientes. Os clientes

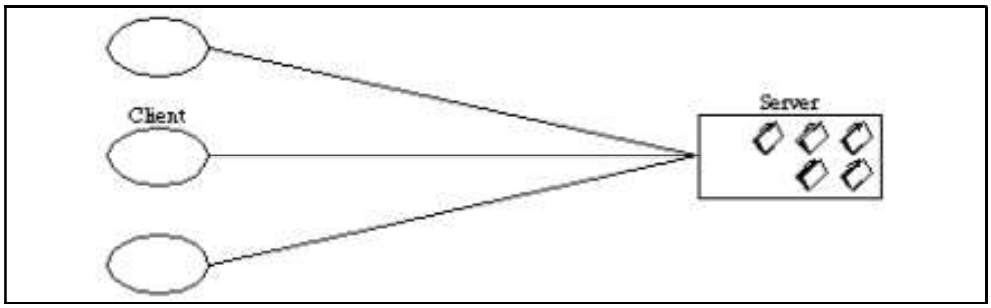
devem, de algum modo, realizar a conexão com o servidor para que possam utilizar a informação armazenada.

Como não se mantém uma relação entre os clientes, a tramitação entre soluções ocorre de maneira simplificada. Uma vez que não existe dependência entre os clientes, a sua portabilidade para diversas plataformas é facilitada. A incomplexidade do servidor agrega para a sua escalabilidade. Além disso, neste modelo é possível que os clientes e os servidores se desenvolvam de maneira distinta (FIELDING, 2000).

2.2 STATELESS

Devido à característica citada no subcapitulo 2.1, a comunicação realizada entre cliente e servidor deve ser sem estado (*stateless*), isto é, nenhum estado da sessão, comunicação entre cliente e servidor, pode ser armazenado no servidor. Toda e qualquer solicitação do cliente para o servidor deve conter todas as informações necessárias para que o servidor consiga compreender e atender à requisição feita. Estados de sessão devem ser mantidos inteiramente no cliente. (FIELDING, 2000). Na Figura 13 é demostrada a comunicação realizada.

Figura 13 - Comunicação entre Cliente e Servidor - *Statless*



Fonte: Fielding (2000)

Estas restrições provêm um aumento de pontos importantes do sistema, como visibilidade, confiabilidade e escalabilidade. Para o servidor não é necessário controlar/verificar nenhuma informação enviada pelo cliente, uma vez que as solicitações realizadas devem conter todos os dados necessários para serem resolvidas, é garantida uma melhora no quesito de visibilidade do sistema.

A confiabilidade é aperfeiçoada, pois facilita a tarefa de recuperação de falhas parciais. A escalabilidade é melhorada, tendo em vista que não é necessário guardar os

estados das sessões, possibilitando ao servidor liberar rapidamente os recursos que foram utilizados, e também simplificando a implementação do sistema.

Entretanto, a desvantagem de implementar uma comunicação sem estado (*stateless*), é que possivelmente o desempenho da rede, entre cliente e servidor, seja prejudicado, devido ao aumento do trafego de dados repetitivos emitidos em uma série de solicitações (*per-interaction overhead*). Isto ocorre, pois não é permitido que os dados enviados para o servidor sejam armazenados e compartilhados em interações futuras.

2.3 CACHE

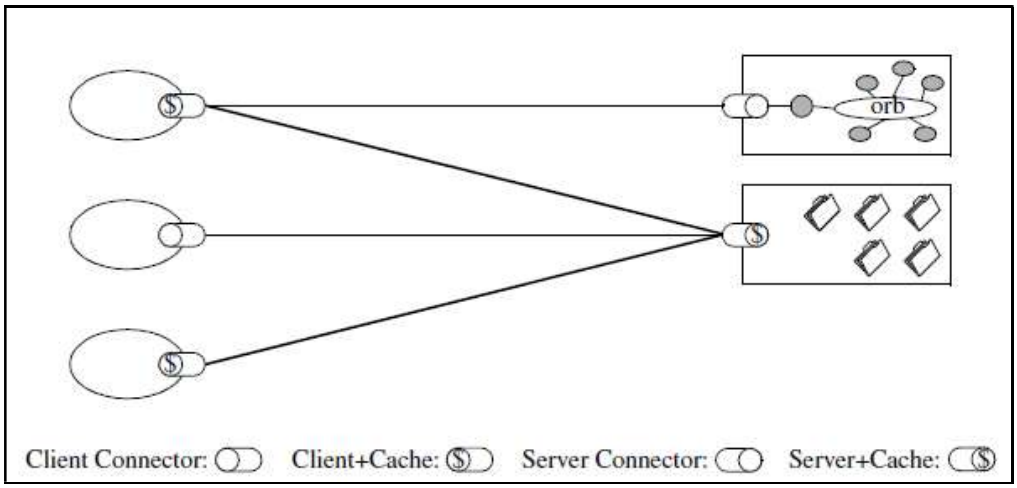
A restrição *cache* é fundamentada sob as restrições cliente-servidor e *Stateless* (FIELDING, 2000). *Cache* é um componente que pode ser implementado, para que seja possível armazenar temporariamente informações, visando um ganho de performance da rede (VELLOSO, 2014).

As respostas devolvidas pelo servidor devem ser sinalizadas de forma direta ou indiretamente como *cacheable*[1] ou *non-cacheable*[2]. Um componente de *cache* atua como um interlocutor entre o cliente e o servidor, proporcionando respostas de solicitações já requisitadas. Estas respostas poderão ser reutilizadas, uma vez que forem do tipo *cacheable* e que gerem o mesmo resultado de uma solicitação que foi realizada anteriormente (FIELDING, 2000).

Na Figura 14 pode ser observado que as solicitações são enviadas para um componente de *cache*, onde é verificada a possibilidade de resolver a solicitação, com respostas já recebidas. Com isso, é possível eliminar algumas interações realizadas entre o cliente e o servidor.

1 Identifica a resposta ao servidor como passível de cache.
2 Identifica a resposta ao servidor como não passível de cache.

Figura 14 – Componente *Cache*



Fonte: Fielding (2000)

O motivo da construção de componentes de *cache* em projetos considerados *RESTful*, é a criação de contrapesos para mitigar possíveis desvantagens apresentadas pela restrição *Stateless.* Deste modo, é possível eliminar interações, melhorando a eficácia, a escalabilidade e o desempenho notado pelo usuário.

2.4 INTERFACE UNIFORME

A Interface uniforme é um acordo de comunicação entre o cliente e o servidor, isto é, uma série de pequenas restrições que visam generalizar a comunicação. Os dados devem ser enviados, entre o cliente e o servidor de forma padronizada. Esta padronização pode afetar a performance percebida na rede, devido a conversão/transformação dos dados para o modelo acordado (FIEDLDING,2000).

Para tornar mais simples e melhorar a visibilidade das interações, um sistema *RESTful* necessita que todos os seus componentes possuam uma interface uniforme (FIELDING, 2000). Além disso, a implementação não fica dependente de outros serviços disponibilizados, o que possibilita que ambas as partes consigam ser desenvolvidas de forma distinta. Entretanto, para a implementação de uma interface uniforme num sistema, este deve obedecer quatro restrições: identificação de recursos, manipulação de recursos por meio de representações, mensagens auto-descritivas e hipermídia como o motor do estado da aplicação (FIELDING, 2000).

O protocolo HTTP, teoricamente, é um entre tantos protocolos que possibilita a implementação de uma interface uniforme, sendo assim, a maior parte das aplicações *RESTful* são construídas utilizando-o (WEBBER; PARASTATIDIS; ROBINSON, 2010). O reconhecimento de recursos, no protocolo HTPP é realizado mediante URIs (*Uniform Resource Identifiers*) (STEPHEN, 2001). Um exemplo de URI que representa recurso de pessoas seria: http://healthsimulator/pessoas.

Um recurso detectado por uma URI pode dispor de variados tipos de representação. O tipo de representação de um recurso não exige ser o mesmo no qual ele é armazenado no servidor. Este encapsulamento garante um baixo acoplamento (WEBBER; PARASTATIDIS; ROBINSON, 2010), uma vez que o tipo de armazenamento do recurso pode ser convertido para outro mais eficaz sem causar alterações no modo como os clientes manipulam o serviço.

Na web, comumente são utilizados alguns formatos de representação, como: PNG e JPEG para imagens, MPEG para vídeos, HTML para documentos, XML e JSON para dados, entre outros. Ainda que alguns serviços fazem uso de extensões na URI para sinalizar o formato da representação, por exemplo http://healthsimulator/pessoas.json, existem formas mais estruturadas para solucionar este problema.

A abordagem mais usual para tal problema é a conversação do conteúdo, isto é, o cliente estipula no cabeçalho *Accept* do protocolo HTTP quais sãos os formatos aceitos por ele, e o servidor opta por um formato, destes estipulados, como o mais adequado (WEBBER; PARASTATIDIS; ROBINSON, 2010). A URI não identifica de qual maneira o recurso deve ser armazenado, mas sim, realiza a identificação no servidor.

Mensagens auto-descritivas, devem possuir todas as informações essenciais que caracterizam o modo de como serão interpretadas. O cabeçalho de resposta *Content-Type*, no protocolo HTTP, detém o tipo da mídia do corpo da resposta (RICHARDSON; RUBY, 2007). Consequentemente, o cliente pode distinguir uma forma de compreender a mensagem.

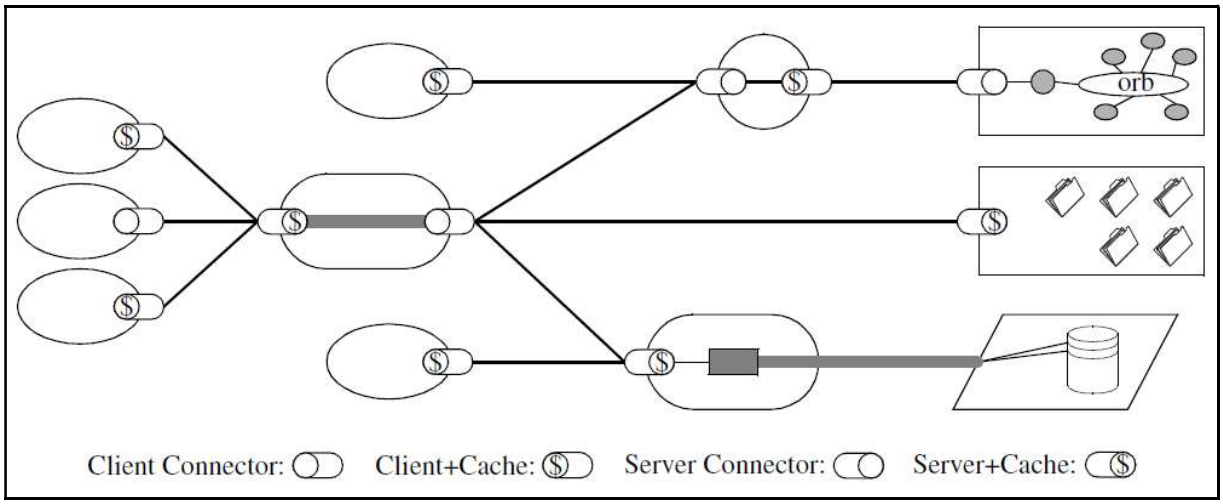
Hipermídia como o motor do estado da aplicação possui o significado de que o estado de uma sessão HTTP não é guardado no servidor, no entanto, é definido pelo caminho que o cliente escolhe navegando pela web (RICHARDSON; RUBY, 2007). O cliente possui o conhecimento de caminho de entrada para a aplicação, baseando neste caminho, o servidor transmite, em cada resposta, *links* que levam o cliente para os

próximos recursos que poderão ser acessados, conduzindo o trajeto que o cliente pode optar na aplicação.

2.5 SISTEMA DE CAMADAS

Um sistema de camadas equivale a uma “torre de camadas”, onde cada uma é sobreposta à outra. A camada superior dispõe dos serviços da inferior a si, contudo, a inferior não tem percepção da superior. Uma camada só é exibida para as que consomem os serviços disponíveis por ela, isto é, em um grupo de camadas no qual a camada 3 usa serviços da 2, e a 2 usa serviços da 1, a 3 desconhece a existência da 1 (FOWLER et al, 2012). Deste modo, numa arquitetura *REST*, uma condição que deve ser atendida é que ela deve ser uma arquitetura de camadas e que cada componente não consegue acessar além da camada com a qual interage diretamente (FIELDING, 2000). A Figura 15 exemplifica a visualização descrita sobre um sistema de camadas.

Figura 15 - Sistema de Camadas



Fonte: Fielding (2000)

2.6 CÓDIGO SOB DEMANDA

A última restrição numa arquitetura *REST* é a oportunidade de o cliente baixar um código (na forma de *applets* ou *scripts*) do servidor (FIELDING, 2000). Isto tem como objetivo simplificar a codificação do cliente, reduzindo o número de funções a serem desenvolvidas, além de melhorar a extensibilidade. Entretanto, é uma restrição opcional,

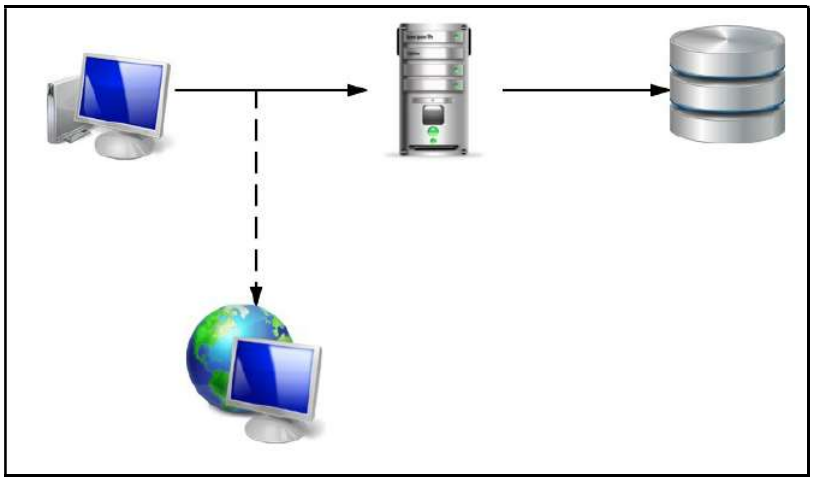
haja visto diminuir a visibilidade do sistema. No subcapitulo seguinte será abortado o assunto de segurança.

2.7 SEGURANÇA

Segundo Saudate (2014), ao realizar requisições HTTP, estas são recebidas e transmitidas através de aparelhos ligados na rede, como, por exemplo, *switches*, *firewalls*, *proxies*, entre outros. Este é o modo de funcionamento normal da internet, isto é, sem estes aparelhos não teríamos Internet. Ainda segundo o mesmo autor, tendo em vista os aparelhos citados anteriormente, não é possível certificar que todos são confiáveis, pois as suas configurações são passíveis de modificação, possibilitando que as informações das requisições possam ser roubadas. Portanto, as modificações das informações proporcionam dois tipos de ataques principais: *man-inthe-midle* (homem no meio) e *eavesdropping* (ouvir escondido).

Eavesdrop possui o significado de espiar, escutar por acaso, portanto neste contexto isto ocorre quando o atacante consegue capturar a requisição, mas não a altera. Este modo de ataque pode ser utilizado para simplesmente “espiar” os dados de uma requisição realizada, com isso o atacante pode reproduzir a requisição com os dados que foram obtidos. A Figura 16, exemplifica como ocorre este ataque.

Figura 16 - Ataque Eavesdropping

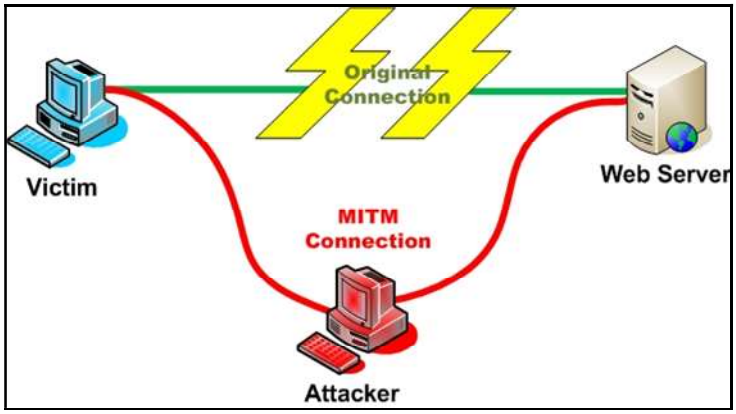


Fonte: Saudate (2014)

Por outro lado, o *man-in-the-middle* é mais grave, pois o atacante altera o curso original da requisição, com a intenção de modificá-la e submetê-la para o servidor, se

passando pelo cliente que originou a requisição. Desta forma, no caso de uma troca de mensagens entre cliente e servidor, a mensagem será enviada para outro cliente com outras informações. Na Figura 17 é exemplificado o ataque *man-in-the-middle*.

Figura 17 - Ataque *Man-in-the-Middle*



Fonte: Graça (2013)

A prevenção para os casos aqui citados encontra-se na criptografia. Utilizando o protocolo HTTP para realizar a comunicação, é possível utilizar dois métodos de autenticação disponibilizados pelo protocolo: HTTP *Basic* e HTTP *Digest*.

O HTTP *Basic* constitui um “desafio e uma resposta”. Se o cliente tentar acessar um recurso que é protegido por esta autenticação e não forem fornecidas as credencias corretas, o servidor encaminhará novamente um “desafio”, para então o cliente enviar novamente a requisição. Este “desafio” é gerado com base no algoritmo Base64 [3] (SAUDATE, 2014).

O HTTP *Digest* é outra forma de prevenir que as credencias de autorização possam ser capturadas por algum dos ataques citados anteriormente. Entretanto, é mais complexa do que a autenticação básica, mas é segura mesmo em HTTP não criptografado. O *Digest* segue o mesmo padrão básico do Basic: o cliente emite um pedido e recebe um desafio.

Quando o cliente realiza uma requisição, sem estar autenticado, recebe como resposta do servidor um desafio. Este desafio possui o formato demonstrado na Figura 18.

3 Trata-se de um algoritmo utilizado principalmente para converter dados binários em código alfanumérico.

Figura 18 - Desafio *Digest*

```
401 Unauthorized
WWW-Authenticate: Digest realm="Default",
   qop="auth",
   nonce="0cc194d2c0f4b6b765d448a578773543",
   opaque="92ea7eedd41225f43d534bc544c2223"
```

Fonte: Saudate (2014)

Os campos *realm*, *qop*, *nonce* e *opaque*, consistem, respectivamente, o contexto da autenticação, o nível de proteção dado, uma *string*[4] aleatória, que será usada futuramente no cálculo de autenticação, e uma *string* que deve ser enviada para o servidor, como maneira de certificar que o cabeçalho não foi modificado. O Cliente em poder destas informações necessita calcular a resposta do desafio, além da senha, número de mensagens já enviadas, o usuário, a URL, o método HTTP a ser utilizado e uma *string* aleatória produzida por ele mesmo. Em posse destes dados, o servidor calculará a resposta do cliente utilizando o algoritmo MD5[5]. Caso o cálculo da resposta seja igual a recebida, o cliente será autorizado, caso contrário, o processo será repetido (SAUDATE, 2014).

2.8 ESCALABILIDADE

A escalabilidade compreende a necessidade de uma arquitetura de sistemas de sustentar um grande número de interações simultâneas ou número de instâncias (ERL et al., 2012). Portanto, são identificados quatro pontos para atender as demandas de escalabilidade:

- Escalonamento - Aumentar a capacidade de serviços, consumidores e dispositivos de rede.
- Dimensionamento - Distribuição de carga entre os serviços e programas.

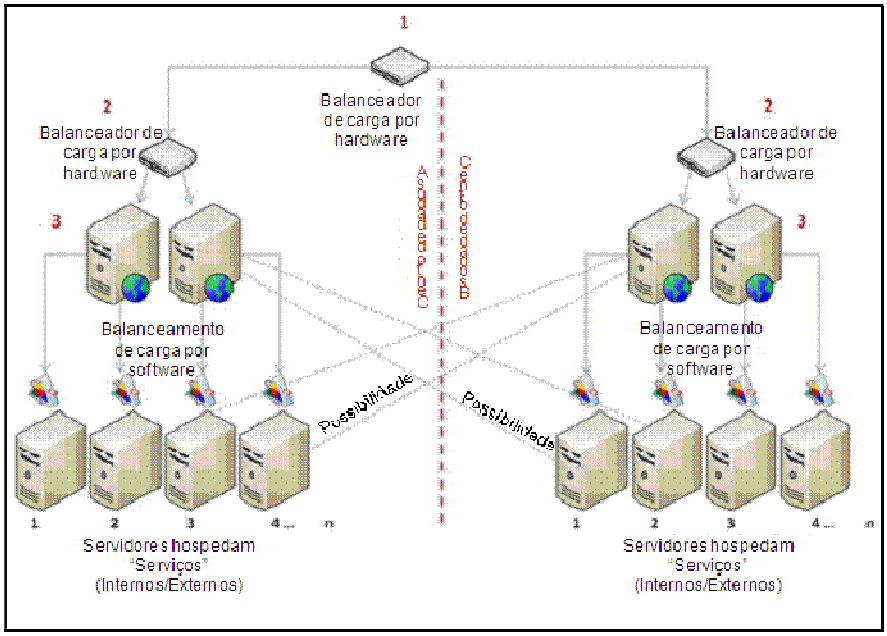
4 Um tipo de dado cadeia de caractere é uma modelagem de uma cadeia formal de caracteres.
5 Não é um algoritmo de criptografia, e sim, de hash (sequência alfanumérica). Por ter essa natureza, uma vez calculado, não pode ser revertido.

- Suavizar – Realizar análise das interações em períodos de congestionamento e de não congestionamento, com isso otimizar a infraestrutura visando reduzir o tempo ocioso da mesma.
- Desacoplando o consumo de recursos finitos - Como a memória de consumidores concorrentes.

A maior parte das restrições da arquitetura REST, apresentada nos capítulos anteriores, pouco servem de auxílio para o escalonamento ou a suavização da quantidade de interações. Entretanto, o foco está em ajudar a escalabilidade de instâncias e na distribuição do consumo de recursos finitos disponível no servidor.

Arquiteturas de sistemas do tipo REST possibilitam o uso de balanceamento de carga entre instâncias de serviços através das restrições de Sistemas de camadas (subcapítulo 2.5) e Interfaces uniformes (subcapítulo 2.4). A Figura 19, demostra o balanceamento de carga.

Figura 19 - Balanceamento de Carga



Fonte: Microsfot (2008)

Deste modo, é possível simplificar a divisão de requisições para servidores distintos, pois uma que é utilizada a restrição *Stateless*, apresentada no subcapítulo 2.2, que estrutura as interações em pares de requisição e resposta, gerando a possibilidade de serem tratadas por serviços distintos e de maneira independente de outras

requisições. Sendo assim, não é necessário manter o direcionamento de um cliente para o mesmo servidor ou sincronizar o estado da sessão explicitamente entre os servidores, pois os dados da sessão estão contidos em cada solicitação. A camada de comunicação para o Health Simulator.

3 CAMADA DE COMUNICAÇÃO

REST é uma das soluções mais comuns e utilizadas na integração de sistemas e comunicação entre aplicações distintas (RICHARDSON; RUBY, 2007). Desta maneira a camada de comunicação desenvolvida neste trabalho é baseada na arquitetura REST.

A camada de comunicação que será apresentada neste capítulo não pode ser considera RESTful, pois não implementa todas as restrições apresentadas no capítulo anterior. Durante o processo de desenvolvimento da camada foi identificado que não existia a necessidade de aplicar a restrição de *cache*. Devido que pode gerar uma baixa confiabilidade em situações onde as inforamções sejam modificadas em um espaço de tempo onde o *cache* é compreendido como válido. (FIELDING, 2000).

Entretanto, é possível implementar esta restrição caso seja identificada a necessidade de se utitilizar. Uma vez que *cache* traz beneficios como diminuir o número de interações, refletindo numa melhor escalabilidade e redução de tempo de respota.

A escolha de uma arquitetura do baseada em REST, possibilita que algumas características sejam desenvolvidas na camada de comunicação, sendo elas: maior escalabilidade, maior compatibilidade entre aplicações distintas, sem bloqueios por *firewalls ou proxies* de rede, entre outras. Nos próximos subcapítulos serão abordados assuntos sobre o desenvolvimento da camada de comunicação, desenvolvimento dos métodos e suas definições.

3.1 APLICAÇÃO WEB API

Ao total foram desenvolvidos 13 métodos, até a apresentação deste trabalho, para a camada de comunicação. Estes métodos são os pilares da troca de informação entre o simulador (front-end) com a parte administrativa do Health Simulator (back-end).

O desenvolvimento dos métodos visa atender o levantamento de requisitos que foi realizado com base do arquivo de decupagem, conforme consta no Apêndice A. Portanto, foi definido que para acessar as ações dos métodos que foram implementados, deve ser usada uma requisição HTTP com as informações serializadas no formato JSON (JSON,2014). No próximo subcapitulo serão abordados os métodos desenvolvidos.

3.2 DESENVOLVIMENTO DA CAMADA DE COMUNICAÇÃO

O desenvolvimento dos métodos da camada de comunicação foi realizado baseado em princípios de um arquitetura do tipo REST, apresentada no capítulo anterior. Além da utilização do *framework*[6], que possibilita uma base forte e escalável, facilitando a integração entre as duas partes do simulador.

Visando atender a restrição de uma interface uniforme da arquitetura REST foi definido o seguinte padrão de acesso aos métodos, http://cas.feevale.br/api/v1/{NomeDoMetodo}/{Parametros}, assim como, devem ser respeitados os tipos de requisição, GET e POST, os quais foram estabelecidos como padrão para o projeto.

Segundo Fielding (2000), POST é um método desenvolvido no protocolo HTTP, o qual possui finalidade de requisitar ao servidor que aceite os dados que são enviados no corpo de sua requisição. Ainda segundo o mesmo autor, GET é projetado para recuperar qualquer informação identificada na requisição de uma URL. Como, por exemplo, ao realizar uma requisição do tipo GET para um método da camada de comunicação que retorna uma lista de informações de casos de estudo. Além disto, visando a segurança da troca de informações entre cliente e servidor, foi criado um controle de autenticidade de usuários, que será abordado no subcapítulo 3.3.

Portanto, nos próximos subcapítulos serão abordados os 13 métodos que foram desenvolvidos. Métodos estes que servem como meio de troca de dados para o simulador. No próximo subcapitulo será apresentado o método de autenticação de usuário

3.2.1 Solicita Autenticação Usuário

O método **solicitaAutenticacaoUsuario** é utilizado pelo simulador, quando um aluno solicitar a validação de seu usuário. Desta forma, será necessário que o simulador realizar uma requisição do tipo *GET* para o método **solicitaAutenticacaoUsuario**. Neste contexto, a chamada do método será:

6 Framework: Um conjunto de blocos de construção de software pré-fabricados que programadores podem usar, ampliar ou personalizar para soluções especificas de computação (TALIGNET,1997)

http://cas.feevale.br/api/v1/solicitaAutenticacaoUsuario?loginUser=admin&senhaUser=admin. A Figura 20 apresenta o código fonte do método.

Figura 20 - Método Solicita Autenticação Usuário

```
[HttpGet]
[Route("solicitaAutenticacaoUsuario")]
public SolicitaAutenticacaoUsuarioResposta SolicitaAutenticacaoUsuario(
string loginUser, string senhaUser)
{//cria variavel com dados do usuario
UsuarioDados _user = new UsuarioDados();
//verifica no banco dados as inforamçoes de login e password informadas
var q = from u in context.User where
u.loginUser == loginUser && u.passwordUSer == senhaUser
select u;
//carrega informações para dentro da variavel user
var user = q.FirstOrDefault();
//verificar se user é igual a nulo, caso for nullo retorno mensagem de erro.
if (user == null){
return new SolicitaAutenticacaoUsuarioResposta("Usuário ou senha inválidos");}
else{//Cria chave de segurança, para validar autencidade do usuário.
var criaResultado = new Result()
{idUser = user.idUser,idSession = Guid.NewGuid(),
idCaseStudy = 0,idNetWork = 0};
//Salvo chave criada no banco.
context.Result.Add(criaResultado);
context.SaveChanges();
//Carrega a variavel _user com as informações que retornaram do banco de dados.
_user.birthDayUser = user.birthDayUser;
_user.emailUser = user.emailUser;
_user.functionUser = user.functionUser;
_user.genderUser = user.genderUser;
_user.idInstitution = user.idUser;
_user.idUser = user.idUser;
_user.loginUser = user.loginUser;
_user.nameUser = user.nameUser;
_user.phoneUser = user.phoneUser;
_user.photoUser = user.pictureUser;
_user.idSession = criaResultado.idSession;
//Retorno JSON com dados do usuário e mensagem de sucesso.
return new SolicitaAutenticacaoUsuarioResposta( user);}}
```

Fonte: Elaborado Pelo autor

Neste cenário é utilizado o recurso /api/v1/solicitaAutenticacaoUsuario, que verificará a existência das credenciais informadas no banco de dados. O acesso para iniciar a simulação é disponibilizado ao aluno, mediante a confirmação das credencias enviadas. Além disto, são retornadas as informações do usuário que foi autenticado, conforme é exemplificado na Figura 21.

Figura 21 - Retorno de Sucesso Solicita Autenticação Usuário


```
{
    "dados": {
        "idUser": 1,
        "phoneUser": null,
        "emailUser": "admin@email.com",
        "birthDayUser": "1992-05-15T00:00:00",
        "photoUser": null,
        "functionUser": "adm",
        "loginUser": "admin",
        "nameUser": "admin",
        "genderUser": "M",
        "idInstitution": 1
    },
    "status": "Sucesso",
    "mensagem": "Usuário autenticado"
}
```


Fonte: Elaborado Pelo autor

Além do retorno de dados, é gerada uma chave única para cada acesso realizado. A mesma é salva no banco dados, respeitando a característica *stateless*, apresentada no Capítulo 2.

A finalidade de criação de uma chave única é para realizar o controle de autenticidade dos usuários que estão realizando a simulação. Desta maneira, é possível garantir a segurança na transação de informação entre *back-end* e *front-end*. Caso seja identificado que as credências informadas pelo aluno não existam, é retornada uma mensagem de erro, exemplificado pela Figura 22.

Figura 22 - Retorno de Erro Solicita Autenticação Usuário


```
{
    "dados": null,
    "status": "Erro",
    "mensagem": "Usuário ou senha inválidos"
}
```


Fonte: Elaborado Pelo autor

3.2.2 Cadastra Usuário

O recurso **cadastraUsuario** desenvolvido para a camada de comunicação, possibilita ao jogador, através do simulador, informar um novo usuário para que seja cadastrado no banco de dados. A requisição esperada por este método é do tipo POST, devido a necessidade de receber informação de um novo usuário.

O método **cadastraUsuario** foi disponibilizado na URL http://cas.feevale.br/api/v1/cadastraUsuario. A Figura 23, expõe o código fonte criado para realizar a implementação do método apresentado.

Figura 23 - Método Cadastra Usuário

```
[HttpPost]
[Route("cadastraUsuario")]
public Resposta CadastraUsuario(CadastraUsuarioDados dados)
{// Carrego informações recebidas para a variável novoUsario
var novoUsuario = new User()
{phoneUser = dados.phoneUser,emailUser = dados.emailUser,
birthDayUser = dados.birthDayUser,pictureUser = dados.photoUser,
functionUser = dados.functionUser,loginUser = dados.loginUser,
nameUser = dados.nameUser,passwordUSer = dados.passwordUser,
genderUser = dados.genderUser,creationDateUser = DateTime.Now,
activeUser = false,approvedUser = false};
//Valido as informações passadas
var resposta = ValidaDadosUsuario(novoUsuario);
/*Caso alguma das informaçoes passadas for invalida,
é retorna uma mensagem contendo o que está errado
Exemplo: Telefone não informado*/
if (resposta != null){
return resposta;}
/*Salvo as informações que foram
atribuidas para a variável novoUsario*/
context.User.Add(novoUsuario);
context.SaveChanges();
// Retorno mensagem de sucesso.
return new Resposta();}
```

Fonte: Elaborado Pelo autor

Antes de salvar novo usuário no banco de dados, é verificada a integridade das informações deste usuário. Portanto, se houver elemento incorreto ou não informado, o método retorna ao simulador o que foi identificado. Entretanto, se for comprovada a integridade dos dados informados, é realizado o cadastro do novo usuário e o método retorna uma mensagem de sucesso. A Figura 24, demonstra as duas possibilidades aqui apresentadas.

Figura 24 – Retornos Método Erro e Sucesso

```
{
    "status": "Erro",
    "mensagem": "Telefone não informado"
}
```

```
{
    "status": "Sucesso",
    "mensagem": ""
}
```

Fonte: Elaborado Pelo autor

3.2.3 Altera Cadastra Usuário

A construção do método **alteraCadastraUsuario** tem como objetivo possibilitar ao aluno que atualize ou retifique algum dado de seu cadastro que esteja incorreto. Devido a necessidade de informar dados para realizar esta alteração, o método espera uma requisição do tipo POST, sendo necessário informar os dados no corpo da requisição.

Na Figura 25 é exposto o código fonte do método apresentado. Assim como no método de Cadastro de Usuário é necessário verificar a integridade dos dados informados. Além disso, é fundamental a busca pelo usuário que sofrerá a alteração de seu cadastro.

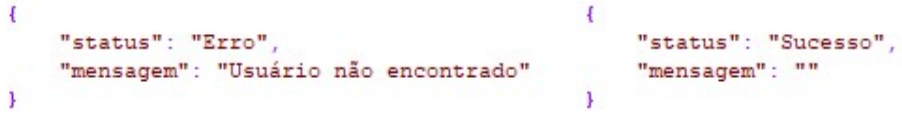
Figura 25 - Método Altera Cadastra Usuário

```
[HttpPost]
[Route("alteraCadastraUsuario")]
public Resposta AlteraUsuario(AlteraUsuarioDados dados)
{/*Verifico a existencia do usuário que está sendo
informado na chamda do método*/
var usuario = context.User.Find(dados.idUser);
//Caso o usuário não existir
if (usuario == null){
//Retorno resposta de que não consegui achar o usuário informado.
return new Resposta("Usuário não encontrado");}
// Carrego as novas informações para a variável usuário
usuario.phoneUser = dados.phoneUser;
usuario.emailUser = dados.emailUser;
usuario.birthDayUser = dados.birthDayUser;
usuario.pictureUser = dados.photoUser;
usuario.functionUser = dados.functionUser;
usuario.loginUser = dados.loginUser;
usuario.nameUser = dados.nameUser;
usuario.genderUser = dados.genderUser;
//Valido as informaçoes que foram carregas
var resposta = ValidaDadosUsuario(usuario);
/*Caso alguma informação for invalida,
retorno mensagem de acordo com o campo validado.*/
if (resposta != null){return resposta;}
//Salvo as alterações no banco de dados.
context.SaveChanges();
//Retorno de mensagem de sucesso.
return new Resposta();}
```

Fonte: Elaborado Pelo autor

Portanto, foram projetados dois possíveis tipos de retorno para requisições realizadas neste método, sendo um status que apresente erro e uma mensagem de auxílio, assim como um *status* de sucesso. Na Figura 26 são demonstrados os dois possíveis cenários.

Figura 26 – Retornos Método Altera Cadastra Usuário



Fonte: Elaborado Pelo autor

3.2.4 Listar Avatar

O método **ListaAvatar** foi desenvolvido para retornar uma lista que contém informações sobre avatares e personagens. Por retornar dois tipos de informações distintas, é possível filtrar os dados retornados, com um parâmetro opcional nameCategoryAvatar. Portanto, a requisição esperada neste método será do tipo GET. Além, do parâmetro opcional, é requerida a informação dos parâmetros idUser e IdSession. A Figura 27Figura 27 exibe o código gerado pela implementação do método.

Figura 27 – Método Lista Avatar

```
[HttpGet]
[Route("listaAvatar")]
public ListaAvatarResposta ListaAvatar(int idUser,
Guid idSession, string nameCategoryAvatar){
//Valido a sessão com a chave criada no método de autenticação de usuário
var validasession = context.Result.Where(x => x.idUser == idUser &&
x.idSession == idSession && x.sessionValid == true).FirstOrDefault();
//Caso encontre a combinação de chave e idUser
if (validasession != null){//busco no banco de dados, o idAvatar e urlAvatar
var listaAvatar = from a in context.Avatar
where a.nameCategoryAvatar == (nameCategoryAvatar == "todos" ?
a.nameAvatar : nameCategoryAvatar)select new ListaAvatarDetalhe()
{idAvatar = a.idAvatar,nameEthnicityAvatar = a.nameEthnicityAvatar,
nameTextureAvatar = a.nameEthnicityAvatar,
nameCategoryAvatar = a.nameCategoryAvatar,
nameAgeAvatar = a.nameAgeAvatar,nameBiotypeAvatar = a.nameBiotypeAvatar,
nameAvatar = a.nameAvatar,genderAvatar = a.genderAvatar,
thumbnailAvatar = a.thumbnailAvatar,urlAvatar = a.urlAvatar};
/*retorno uma listagem com todos as informações que foram encontadas,
juntamente com resposta de sucesso.*/
return new ListaAvatarResposta(listaAvatar);}
/*Caso sessão seja invalida retorno nulo.*/
else{return new ListaAvatarResposta(null);}}
```

Fonte: Elaborado Pelo autor

Antes de iniciar qualquer busca com os parâmetros informados, é verificado se o usuário e sessão são autenticas. Uma vez reconhecido o usuário, é efetuada a pesquisa no banco de dados. Assim como nos demais métodos existem dois possíveis retornos, um de sucesso, contendo uma lista com as informações requisitadas e um de erro, com uma mensagem para auxílio. Na Figura 28 é exemplificado um retorno contendo os dados solicitados e o *status* de sucesso.

Figura 28 – Retorno de Sucesso Lista Avatar


```
{
    "listaAvatar": [
        {
            "idAvatar": 37,
            "nameEthnicityAvatar": "Asiatico",
            "nameTextureAvatar": "Asiatico",
            "nameCategoryAvatar": "Medico",
            "nameAgeAvatar": "Jovem",
            "nameBiotypeAvatar": "Normal",
            "nameAvatar": "Medicos",
            "genderAvatar": "M",
            "thumbnailAvatar": "caminhothumbnail",
            "urlAvatar": "Medicos/Masculino/Asiatico/Normal"
        },
        {
            "idAvatar": 38,
            "nameEthnicityAvatar": "Caucasiano",
            "nameTextureAvatar": "Caucasiano",
            "nameCategoryAvatar": "Medico",
            "nameAgeAvatar": "Adulto",
            "nameBiotypeAvatar": "Gordo",
            "nameAvatar": "Medicos",
            "genderAvatar": "F",
            "thumbnailAvatar": "caminhothumbnail",
            "urlAvatar": "Medicos/Feminino/Caucasiano/Gordo"
        }
    ],
    "status": "Sucesso",
    "mensagem": ""
}
```


Fonte: Elaborado Pelo autor

3.2.5 Consulta Avatar

O método de consulta avatar, tem como objetivo retornar um avatar ou personagem específico. Assim como o método de **listaavatar**, o tipo da requisição enviada ao método é GET, sendo necessário informar o iduser, idession e o idavatar que deseja encontrar. É exposto pela Figura 29, a implementação criada para o método apresentando.

Figura 29 – Método Consulta Avatar

```
[HttpGet]
[Route("consultaAvatar")]
public ConsultaAvatarResposta ConsultaAvatar(int idUser,
Guid idSession, int idAvatar){/*Valido a sessão com a chave criada no
método de autenticação de usuário*/var validasession = context.Result
.Where(x => x.idUser == idUser && x.idSession == idSession &&
x.sessionValid == true).FirstOrDefault();
//Caso encontre a combinação de chave e idUser
if (validasession != null){/*Busco no banco de dados as informações
sobre o idAvatar requisitado.*/var listaAvatar = from a in context.Avatar
where a.idAvatar == idAvatar select new ConsultaAvatarRespostaDetalhes()
{idAvatar = a.idAvatar,nameEthnicityAvatar = a.nameEthnicityAvatar,
nameTextureAvatar = a.nameEthnicityAvatar,
nameCategoryAvatar = a.nameCategoryAvatar,
nameAgeAvatar = a.nameAgeAvatar,nameBiotypeAvatar = a.nameBiotypeAvatar,
nameAvatar = a.nameAvatar,genderAvatar = a.genderAvatar,
thumbnailAvatar = a.thumbnailAvatar,urlAvatar = a.urlAvatar};
//Retorno o avatar requisitado.
return new ConsultaAvatarResposta(listaAvatar);}
else{//caso a sessão seja invalida, retorno nulo
return new ConsultaAvatarResposta(null);}}
```

Fonte: Elaborado Pelo autor

O retorno que é informado em cenários de sucesso é semelhante ao método **listaavatar**. Entretanto, será retornado uma única informação. A Figura 30 demonstra um retorno de sucesso para uma requisição realizada.

Figura 30 – Retorno de Sucesso Consulta Avatar


```
{
    "consultaAvatar": [
        {
            "idAvatar": 34,
            "nameEthnicityAvatar": "Asiatico",
            "nameTextureAvatar": "Asiatico",
            "nameCategoryAvatar": "Medico",
            "nameAgeAvatar": "Idoso",
            "nameBiotypeAvatar": "Normal",
            "nameAvatar": "Medicos",
            "genderAvatar": "M",
            "thumbnailAvatar": "caminhothumbnail",
            "imgAvatar": null,
            "pathAvatar": null,
            "urlAvatar": "Medicos/Masculino/Asiatico/Normal"
        }
    ],
    "status": "Sucesso",
    "mensagem": ""
}
```


Fonte: Elaborado Pelo autor

3.2.6 Consulta Lista Caso

O recurso api/v1/consultListaCasos prove a lista de todos os casos de estudos que foram cadastrados na base de dados. Este método, disponibiliza dados fundamentais para o aluno optar pelo caso de estudo que deseja simular.

Neste método espera-se uma requisição do tipo GET, além dos parâmetros de iduser, idsession. Estes parâmetros são utilizados para verificar o usuário autenticado. NaFigura 31 Figura 31 é demonstrada a codificação do método.

Figura 31 – Método Consulta Lista Caso

```
[HttpGet]
[Route("consultaListaCaso")]
public ConsultaCasoResposta ConsultaListaCaso(int idUser, Guid IdSession)
{
    //Valido a sessão com a chave criada no método de autenticação de usuário
    var validasession = context.Result
        .Where(x => x.idUser == idUser && x.idSession == IdSession)
        .FirstOrDefault();
    //Caso a Sessão não for nula
    if (validasession != null)
    {
        //Consulto banco de dados para criar uma lista de casos de estudos cadastrados.
        var listaCasos = from c in context.CaseStudy
                         select new ConsultaCasoRespostaDetalhes()
                         {
                             idCaseStudy = c.idCaseStudy,
                             authorsCaseStudy = c.authorsCaseStudy,
                             introductionCaseStudy = c.introductionCaseStudy,
                             nameCaseStudy = c.nameCaseStudy,
                             publicTypeCaseStudy = c.publicTypeCaseStudy,
                         };
        //retorno em formato de lista as informações encontradas, com mensagem de sucesso.
        return new ConsultaCasoResposta(listaCasos);
    }
    else
    {
        return new ConsultaCasoResposta(null);
    }
}
```

Fonte: Elaborado Pelo autor

A requisição realizada para o método em questão obterá como resultado uma listagem de casos de estudos. Estas informações são utilizadas no simulador com a finalidade de expor as opções ao aluno. Do mesmo modo como os demais métodos já apresentados, existem dois possíveis retornos para o método, sendo um deles de sucesso e outro de erro. A Figura 36 demonstra um resultado de sucesso.

Figura 32 – Retorno de Sucesso Consulta Lista Caso


```
{
"listaCaso":[
  {
     "idCaseStudy":5,
     "authorsCaseStudy":"Professor de Enfermagem",
     "introductionCaseStudy":"Paciente Sofreu queda de bicicleta",
     "nameCaseStudy":"AcidenteBicicleta",
     "publicTypeCaseStudy":"P",
     "enable":false
  },
  {
     "idCaseStudy":6,
     "authorsCaseStudy":"Professor de Enfermagem",
     "introductionCaseStudy":"Mussum Ipsum, cacilds vidis litro abertis. "
     "nameCaseStudy":"Acidente de Carro",
     "publicTypeCaseStudy":"P",
     "enable":false
  }
],
"status":"Sucesso",
"mensagem":""
}
```


Fonte: Elaborado Pelo autor

3.2.7 Solicita Caso

O método **solicitaCaso** tem como finalidade consultar informações de um caso específico, desta maneira o recurso disponibilizado pode ser requisitado pela URL http://cas.feevale.br/api/v1/solicitaCaso. Conforme a Figura 37, os parâmetros necessários são: idUser, idSession e idCaseStudy. Além disto, é possível visualizar que este método somente aceita requisições do tipo GET.

Figura 33 – Solicita Caso

```
[HttpGet]
[Route("solicitaCaso")]
public SolicitaCasoResposta SolicitaCaso(int idUser, Guid IdSession, int idCaseStudy)
{
    //Valido a sessão com a chave criada no método de autenticação de usuário
    var validasession = context.Result
        .Where(x => x.idUser == idUser && x.idSession == IdSession)
        .FirstOrDefault();
    //Caso a Sessão não for nula
    if (validasession != null)
    {
        //Busco informações especificas do caso de estudo solicitado.
        var caso = from c in context.CaseStudy
                   where c.idCaseStudy == idCaseStudy
                   select new SolicitaCasoRespostaDetalhes()
                   {
                       idCaseStudy = c.idCaseStudy,
                       authorsCaseStudy = c.authorsCaseStudy,
                       introductionCaseStudy = c.introductionCaseStudy,
                       nameCaseStudy = c.nameCaseStudy,
                       publicTypeCaseStudy = c.publicTypeCaseStudy
                   };
        //Retorno as informações encontradas, juntamente com mensagem de sucesso.
        return new SolicitaCasoResposta(caso.FirstOrDefault());
    }
    else
    {
        return new SolicitaCasoResposta(null);
    }
}
```

Fonte: Elaborado Pelo autor

Deste modo, o método viabiliza as informações do caso solicitado, em situações onde todos os pontos de validação obtiveram sucesso. Entretanto, em um cenário contrário, é disponibilizado um retorno com a informação do possível erro ocorrido, conforme é exemplificado na Figura 34.

Figura 34 – Retorno de Erro Solicita Caso

```
{
    "caso":null,
    "status":"Erro",
    "mensagem":"Caso não encontrado"
}
```

Fonte: Elaborado Pelo autor

3.2.8 Pacote Caso

O método **pacoteCaso** desempenha um papel importante na camada de comunicação, uma vez que os principais dados para dar início a simulação são entregues por ele. A requisição para este método ocorre no momento em que o aluno seleciona o caso de estudo para simular.

O método está disponível na URL http://cas.feevale.br/api/v1/pacoteCaso, sendo necessário realizar uma requisição do tipo GET, além de informar os parâmetros requisitados. O código fonte do método é apresentando na Figura 35.

Figura 35 – Método Pacote Caso

```
[HttpGet][Route("pacoteCaso")]
public PacoteCasoResposta Pacotecaso(int idUser, Guid IdSession, int idCaseStudy)
{//Valido a sessão com a chave criada no método de autenticação de usuário
var validasession = context.Result.Where(x => x.idUser == idUser &&
x.idSession == IdSession && x.sessionValid == true).FirstOrDefault();
/*Caso a Sessão não for nula*/if (validasession != null){
/*Realiza a busca conforme filtro informado*/
var pacote = from c in context.CaseStudy
join cn in context.CaseStudy_has_tbNode on c.idCaseStudy equals cn.idCaseStudy
join n in context.Node on cn.idNode equals n.idNode
join cs in context.CaseStudy_has_tbScene on c.idCaseStudy equals cs.idCaseStudy
join s in context.Scene on cs.idScene equals s.idScene
join st in context.Scene_has_tbTypeScene on s.idScene equals st.idScene
join t in context.TypeScene on st.idTypeScene equals t.idTypeScene
where c.idCaseStudy == idCaseStudy
select new PacoteCasoDetalhes(){idCaseStudy = c.idCaseStudy,
nameCaseStudy = c.nameCaseStudy,authorsCaseStudy = c.authorsCaseStudy,
introductionCaseStudy = c.introductionCaseStudy,completedCaseStudy = c.completedCaseStudy,
statusCaseStudy = c.statusCaseStudy,creationDateCaseStudy = c.creationDateCaseStudy,
publicTypeCaseStudy = c.publicTypeCaseStudy,nameScene = s.nameScene,
descriptionScene = s.descriptionScene,classScene = s.categoryScene,
categoryScene = s.categoryScene,amountSupportingCharacter = s.amontSupportingCharacter,
imageScenario = s.imageScenario,descriptionTypeScene = t.descriptionTypeScene,
nameTypeScene = t.nameTypeScene};/*Retorno as informações encontradas juntamente
com mensagem de sucesso.*/return new PacoteCasoResposta(pacote);}
else{return new PacoteCasoResposta(null);}}
```

Fonte: Elaborado Pelo autor

O retorno do método PacoteCaso é exposta pela Figura 36, que exemplifica um retorno onde todos os dados foram encontrados e validados com sucesso. Entre estes dados podem se destacar informações sobre qual o tipo de cena irá ocorrer, além da quantidade de personagens suportado nesta cena.

Figura 36 – Retorno de Sucesso Pacote Caso


```
{
   "pacote":{
      {
         "idCaseStudy":6,
         "authorsCaseStudy":"Professor de Enfermagem",
         "completedCaseStudy":"S",
         "creationDateCaseStudy":"2017-05-20T00:00:00",
         "introductionCaseStudy":"Mussum Ipsum, cacilds vidis litro abertgna.",
         "nameCaseStudy":"Acidente de Carro",
         "publicTypeCaseStudy":"P",
         "statusCaseStudy":"A",
         "nameScene":"Consultorio",
         "descriptionScene":"Consultório com atendente e médico",
         "imageScenario":"caminho img do cenário",
         "classScene":"Privado",
         "categoryScene":"Privado",
         "amountSupportingCharacter":3,
         "nameTypeScene":"Consultorio",
         "descriptionTypeScene":"Consultório Particular"
      }
   },
   "status":"Sucesso",
   "mensagem":""
}
```


Fonte: Elaborado Pelo autor

3.2.9 Infere Nodo

O método **infereNodo**, assim como o método **pacoteCaso**, é de grande importância para o projeto, uma vez que as inferências na rede bayesiana são realizadas através dele. Portanto, o recurso foi definido como api/v1/infereNodo. Podendo ser acessado no link http://cas.feevale.br/api/v1/infereNodo?idUser=aluno1&idSession=6D0228BBED944C758E7CF4C9E35B7C8E&idNode=23. A Figura 37 demostra o código que foi desenvolvido para realizar a inferência de um nodo específico.

Figura 37 – Método Infere Nodo

```
[HttpGet][Route("infereNodo")]
public Retornoinferencia json(int idUser, Guid idSession, int idNode, string rede)
{/*Valido a sessão com a chave criada no método de autenticação de usuário*/
var validasession = context.Result.Where(x => x.idUser == idUser && x.idSession
 == idSession).FirstOrDefault();//Caso a Sessão não for nula
if (validasession != null){string result;/*Configuração para realizar enferência
do nodo informado*/ var httpWebRequest = (HttpWebRequest)WebRequest
.Create("https://bayesjs-ws-mugnnggzhv.now.sh/"); httpWebRequest
.ContentType = "application/json";httpWebRequest.Method = "POST";
//leitura do arquivo que contem a rede bayesiana e Captura do retorno da enferência
using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{JObject o1 = JObject.Parse(File.ReadAllText(rede)); string jsson = o1.ToString();
streamWriter.Write(jsson); streamWriter.Flush(); streamWriter.Close();}
var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{result = streamReader.ReadToEnd();}var percent = double.Parse(result);
//Busca na base de dados qual a resposta de deve retornar para o jogador.
var msg = from n in context.Node join nq in context.Node_has_tbQuestionNode
on n.idNode equals nq.idNode join qn in context.QuestionNode on nq.idQuestionNode
equals qn.idQuestionNode join qa in context.QuestionNode_has_tbAnswerNode on
qn.idQuestionNode equals qa.idQuestionNode join an in context.AnswerNode on
qa.idAnswerNode equals an.idAnswerNode where an.percentageAnswerNode == percent
select new RetornoinferenciaDetalhe(){msg = an.textAnswerNode};
//retorno da mensagem encontrada na base de dados.
return new Retornoinferencia(msg);}else{return new Retornoinferencia(null);}}
```

Fonte: Elaborado Pelo autor

Realizada a inferência para o nodo desejado é obtida a resposta textual cadastrada no banco de dados. Como exemplo de retorno pode ser citado, dados de um exame médico, resposta de um paciente, entre outras informações. Na Figura 38 é apresentado um retorno de uma inferência realizada.

Figura 38 – Retorno de Sucesso Infere Nodo


```
{
    "msg":"Resposta da Inferência nodo X",
    "status":"Sucesso",
    "mensagem":""
}
```


Fonte: Elaborado Pelo autor

3.2.10 Salva Nodos Rede

O recurso api/v1/salvaNodosRede tem como funcionalidade ler uma rede bayesiana e salvar seus nodos, juntamente com os valores de seus estados. A requisição aceita pelo recurso é do tipo GET. O recurso pode ser acesso através da chamada:

http://cas.feevale.br/api/v1/salvaNodosRede?rede=CaminhoDaRede. Na Figura 39 é apresentado o código implementado no desenvolvimento do método.

Figura 39 – Método Salva Nodos Rede

```
[HttpGet][Route("salvaNodosRede")]
public Resposta salvanodos(string rede){
JObject o1 = JObject.Parse(File.ReadAllText(rede));
var data = JsonConvert.DeserializeObject<formatjson>(o1.ToString());
if(data != null) {foreach (var id in data.network){
var nodo = new Core.Database.Entity.Node(){nameNode = id.Key,
networkPath = rede};context.Node.Add(nodo);context.SaveChanges();
foreach (var item2 in data.network[id.Key].states){
var nodov = new NodeValues(){idNode = nodo.idNode,
value = item2.ToString()};context.NodeValues.Add(nodov);
}context.SaveChanges();}//Retorno respota de sucesso.
return new Resposta();}else{
return new Resposta("Rede Bayesiana apresentou problemas!");}}
```

Fonte: Elaborado Pelo autor

Este método não será utilizado pelo simulador, uma vez que o simulador não cria as redes bayesianas, portanto deverá ser utilizado pelo *back-end* deste projeto, durante a criação dos casos de estudos. Devido a utilização no ambiente administrativo do projeto, não é necessário informar os parâmetros de idUser e idSession. Na Figura 40 é exemplificado a mensagem de retorno quando obtiver sucesso para salvar todos os nodos da rede bayesiana.

Figura 40 – Retorno de Sucesso Salva Nodo Rede

```
{
    "status":"Sucesso",
    "mensagem":""
}
```

Fonte: Elaborado Pelo autor

Em caso de ocorrer algum erro durante a leitura da rede bayesiana, ou durante qualquer parte do processo de salvar os dados no banco de dados, o retorno para a requisição do método será com uma mensagem informando que ocorreu algum erro, conforme é demonstrado na Figura 41.

Figura 41 – Retorno de Erro Salva Nodo Rede

```
{
    "status":"Erro",
    "mensagem":"Rede Bayesiana apresentou problemas!"
}
```

Fonte: Elaborado Pelo autor

3.2.11 Resultado

O método **resultado** tem como objetivo exibir o resultado da simulação realizada pelo aluno. O recurso é definido como api/v1/resultado, sendo obrigatória a informação dos parâmetros idUser e idSession. O tipo da requisição aceita neste método é GET. A chamada do método pode ser realizada da seguinte forma: http://cas.feevale.br/api/v1/resultado?idUser=aluno1&idSession=6D0228BBED944C758E7CF4C9E35B7C8E. A Figura 42 apresenta o código do método aqui abordado.

Figura 42 – Método Resultado

```
[HttpGet][Route("resultado")]
public RetornaResposta RetornaResultado(int idUser, Guid idSession)
{//Valido a sessão com a chave criada no método de autenticação de usuário
var validasession = context.Result.Where(x => x.idUser == idUser
&& x.idSession == idSession).FirstOrDefault();
//Caso a Sessão não for nula
if (validasession != null){var retornaresultado = from r in
context.Result where r.idUser == idUser && r.idSession == idSession
select new ResultadoRespostaDetalhe(){idResult = r.idResult,
report = r.report,observation = r.observation};
/*finalizo a sessão*/ validasession.sessionValid = false;
context.SaveChanges();//retorna json com id do resultado e string com report
return new RetornaResposta(retornaresultado.FirstOrDefault());
}else{return new RetornaResposta(null);}}
```

Fonte: Elaborado Pelo autor

Como resposta do método serão retornados em formato JSON três campos *idResult*, *report* e *observation*. O campo *report* contém os dados da simulação realizada pelo aluno. Estas informações podem variar conforme o resultado de cada aluno. O campo *observation* poderá conter indicações de materiais didáticos com objetivo de auxiliar e tirar dúvidas que o aluno possa ter apresentado durante a simulação.

Além de retornar as informações que o aluno obteve durante a simulação, é neste método que a sessão que foi criada no início da simulação pelo método api/v1/solicitacaautenticacaousuario, será finalizada. Na Figura 43 é exposto o retorno em caso de sucesso da requisição para o método.

Figura 43 – Retorno de Sucesso Resultado


```
{
   "retornaresultado":{
      "idResult":1,
      "report":"RELATÓRIO SOBRE O ALUNO",
      "observation":"OBSERVAÇÕES SOBRE A SIMULAÇÃO",
      "sessionValid":false
   },
   "status":"Sucesso",
   "mensagem":""
}
```


Fonte: Elaborado Pelo autor

Na ocorrência de erro durante as validações do método será retornada uma mensagem de erro, informando do ocorrido, conforme é exemplificado na Figura 44

Figura 44 – Retorno de Erro Resultado


```
{
   "retornaresultado":null,
   "status":"Erro",
   "mensagem":"Resultado não encontrado"
}
```


Fonte: Elaborado Pelo autor

3.2.12 Resultado Relatório

O recurso api/v1/resultadoRelatorio possui a funcionalidade de armazenar os dados que o aluno obteve durante a simulação realizada. Este recurso aceita requisições do tipo POST, estas devem conter todo o conteúdo dos resultados no seu corpo de requisição. A Figura 45 apresenta o código criado para este método de comunicação.

Figura 45 – Método Resultado Relatório

```
[HttpPost][Route("resultadoRelatorio")]
public Resposta SalvaRelatorio(DadosResultado dados)
{var resultado = context.Result.Where(x => x.idUser ==
dados.idUser && x.idSession == dados.Idsession && x.sessionValid
== true ).FirstOrDefault();if (resultado == null){return
new Resposta("Relatório de usuário não encontrado");}
resultado.idCaseStudy = dados.idCaseStudy;
resultado.idNetWork = dados.idNetwork;
resultado.report = dados.report;
resultado.observation = dados.observation;
var resposta = ValidaResultado(resultado);
if (resposta != null){return resposta;}
context.SaveChanges();return new Resposta();}
```

Fonte: Elaborado Pelo autor

A requisição para este método tem como retorno uma mensagem de sucesso quando todas as etapas ocorrem de forma esperada e sem apresentação de erros. Esse caso é demonstrado na Figura 46.

Figura 46 – Retorno de Sucesso Resultado Relatório


```
{
   "status":"Sucesso",
   "mensagem":""
}
```


Fonte: Elaborado Pelo autor

3.2.13 Consulta Instituição

O recurso api/v1/consultaInstuicao está disponível para realizar requisições do tipo GET. O recurso retorna todas as instituições que estão cadastradas no banco de dados. A requisição do recurso pode ser chamada conforme o exemplo: http://cas.feevale.br/api/v1/consultaInstuicao?idUser=aluno1&idSession=6D0228BBED944C758E7CF4C9E35B7C8E. Na Figura 47 é exemplificado o código desenvolvido para o método da camada de comunicação.

Figura 47 – Método Consulta Instituição

```
[HttpGet][Route("consultaInstituicao")]
public ConsultaInstituicaoResposta ConsultaInstituicao(
int idUser, Guid idSession, int? id){/*Valido a sessão com
a chave criada no método de autenticação de usuário*/
var validasession = context.Result.Where(x => x.idUser ==
idUser && x.idSession == idSession).FirstOrDefault();
/*Caso a Sessão não for nula*/if (validasession != null)
{//busco no banco de dados as informações sobre instituições.
var listaInstituicoes = from i in context.Institution where
i.idInstitution == (id == null ? i.idInstitution : id)select
new ConsultaInstituicaoRespostaDetalhes(){idInstitution =
i.idInstitution,addressInstitution = i.adressInstitution,
nameInstitution = i.nameInstitution,phoneInstitution =
i.phoneInstitution,addressLogoInstitution = i.adressLogoInstitution,
siteInstitution = i.siteInstitution};
/*retorno em formato de lista as informações que foram
encontradas sobre as instituições cadastradas.*/
return new ConsultaInstituicaoResposta(listaInstituicoes);}else{
return new ConsultaInstituicaoResposta(null);}}
```

Fonte: Elaborado Pelo autor

Como resultado positivo, o método resultado retorna uma listagem de todas as instituições cadastradas no banco de dados. Na Figura 48 é demostrando um retorno para a requisição do recurso.

Figura 48 – Retorno de Sucesso Consulta Instituição


```
{
   "listaInstituicao":[
      {
         "idInstitution":1,
         "addressInstitution":"rua ABC",
         "nameInstitution":"Univerisidade F",
         "phoneInstitution":"51999999999",
         "addressLogoInstitution":"caminho_logo",
         "siteInstitution":"www.feevale.br"
      }
   ],
   "status":"Sucesso",
   "mensagem":""
}
```


Fonte: Elaborado Pelo autor

Como os demais, apresentados neste trabalho, contém um retorno em casos de erros de verificações no banco ou de validação de usuário e sua sessão, conforme é exposto na Figura 49.

Figura 49 – Retorno de Erro Consulta Instituição


```
{
   "listaInstituicao":null,
   "status":"Erro",
   "mensagem":"Instituição não encontrada!"
}
```


Fonte: Elaborado Pelo autor

O próximo subcapitulo, será abordado a integridade das informações que são disponibilizadas pela camada de comunicação.

3.3 CONTROLE DE AUTENTICIDADE DE USUÁRIO

A criação do controle de autenticidade de usuários foi baseado numa arquitetura de tokens de autenticação. Tokens pode ser definido como objeto que um usuário dispõe para realizar a sua autenticação (STALLINGS; BROWN, 2013).

Portanto, pensando em garantir a veracidade das informações e dados gerados durante as simulações do projeto Health Simulador, sendo gerenciado através uma chave única. Esta, por sua vez é criada no momento em que o jogador informa suas credências

para validação do seu *login*, através do método Solicita Autenticação Usuário, apresentado no subcapítulo 3.2.1.

Fazendo uso do controle de autenticidade de usuário é possível certificar que cada simulação será única e exclusiva do aluno que realizou a validação de suas credências. Além disto, para garantir que a chave não possa ser reutilizada pelo mesmo aluno ou outro, esta é inativada no final da simulação do caso de estudo.

3.4 ESCALABILIDADE DOS MÉTODOS

A arquitetura REST, por ser um agrupamento coordenado de restrições, tem como objetivo diminuir a latência e o tráfego gerado na rede. Além de, simultaneamente, prover uma independência e escalabilidade aos seus componentes (FIEDLING, 2000).

Devido a escolha da implementação da camada de comunicação, utilizando-se dos princípios REST, conforme apresentado no segundo capítulo, viabilizou a construção de uma camada que atende as necessidades de um sistema distribuído dentro de uma escala de internet.

4 VALIDAÇÃO E RESULTADOS

Aspirando efetuar validação factual e objetiva, foi definido a metodologia de teste de software. Segundo Neto (2007), teste de software é o meio de estabelecer que um objeto ou item produzido atinge critérios e funcionalidade de modo satisfatório para o ambiente no qual foi planejado. Ainda, segundo o mesmo autor, o propósito da aplicação e construção de teste de *software*, é evidenciar possíveis pontos falhos durante a etapa de desenvolvimento. Deste modo, é viável a correção destas antes da liberação do produto final, no caso deste trabalho, os métodos que compõem a camada de comunicação.

Para compreender os conceitos de testes de *software* é necessário ter como base os conceitos padrões descritos pelo *Institute of Eletrical and Eletronics Engineers*, como:

- Defeito é a atitude inconsciente praticada por um indivíduo ao averiguar uma informação especifica, solucionar uma adversidade ou fazer uso de um *framework* ou método.
- Erro é uma revelação factual de um defeito num trecho de código. Como por exemplo: a divergência entre um parâmetro esperado e o seu valor informado.
- Falha é um procedimento anormal do *software,* diferente do previsto pelo usuário. Um ou mais erros podem ser a razão de uma falha, entretanto, alguns erros jamais poderão gerar uma falha.

Em busca de manter um alto nível e qualidade do código desenvolvido neste trabalho, optou-se por adotar a implementação de testes unitários. Estes têm como objetivo averiguar a menor unidade de um projeto, ou seja, vasculhar frações de código em busca de imperfeições, tanto de lógica, quanto desenvolvimento de código (NETO,2007).

Tomou-se como base inicial da implementação dos testes unitários, um guia criado por Rowan Miller no ano de 2012, intitulado *Testing with a Fake DbContext.* Na Figura 50, é exemplificado como realizar a implementação de um teste unitário, verificando os dados criados.

Figura 50 - Teste Unitário Exemplo

```
[TestMethod]
public void IndexOrdersByName()
{
    var context = new FakeEmployeeContext
    {
        Departments =
        {
            new Department { Name = "BBB"},
            new Department { Name = "AAA"},
            new Department { Name = "ZZZ"},
        }
    };

    var controller = new DepartmentController(context);
    var result = controller.Index();

    Assert.IsInstanceOfType(result.ViewData.Model, typeof(IEnumerable<Department>));
    var departments = (IEnumerable<Department>)result.ViewData.Model;
    Assert.AreEqual("AAA", departments.ElementAt(0).Name);
    Assert.AreEqual("BBB", departments.ElementAt(1).Name);
    Assert.AreEqual("ZZZ", departments.ElementAt(2).Name);
}
```

Fonte: Roawn Miller (2012)

A adoção por realizar testes de software como validação da camada de comunicação resultou em 18 métodos de testes unitários. Para cada método da camada de comunicação, foi elaborado um ou mais testes unitários. O código fonte desenvolvido pode ser consultado no Anexo I.

A elaboração de um número maior de testes possibilita que distintos cenários possam ser apurados e validados. Desta maneira, é possível garantir que a integração dos métodos da camada de comunicação com dados de um cenário real, não apresente nenhum erro ou falha. Em caso de falhas em cenários reais, cria-se uma oportunidade para a elaboração de um novo teste unitário. Oportunidade esta que permite gerar uma camada de comunicação mais concreta e menos suscetível a novas falhas.

A aplicação de testes unitários possibilitou um aproveitamento de cem porcento das validações criadas. A existência de uma taxa de aproveitamento elevada se dá ao fato da metodologia escolhida, pois esta visa a escrita de código buscando a correção de possíveis erros antes da liberação final.

Portanto, aqui serão expostos os resultados alcançados utilizando-se desta técnica. O código fonte elaborado para os testes pode ser consultado no Apêndice B. A disposição dos resultados será dívida da seguinte forma: método e seus respectivos testes

A criação do teste unitário para os métodos da camada de comunicação busca verificar as funcionalidades as quais estes foram propostos. O Quadro 1, apresenta os testes unitários elaborados para verificar a camada de comunicação criada.

Quadro 1 - Testes Unitário Camada de Comunicação Health Simulator

Método de Teste	Obteve Sucesso
AutenticaUsuario_E_Cria_Sessao	Sim
Cadastra_Novo_Usuario	Sim
Altera_usuario_1	Sim
Lista_Avatar_Teste_Deve_Retornar_Maior_Que_Zero	Sim
ConsultaAvatar_Testa_Retorno_Erro	Sim
ConsultaAvatar_Testa_Retorno_Sucesso	Sim
ConsultaAvatar_Testa_Busca	Sim
SolicitaCaso_Retorna_Caso_5	Sim
Solicita_Caso_Anaminesia	Sim
PacoteCaso_Retorno_Quantidade_de_personagens	Sim
PacoteCaso_CasoEstudo_1	Sim
SalvaNodos_rede_1	Sim
Faz_enferencia_nodo_1_Respota_VOCETEMCERTEZADISTO	Sim
VerificaExistencia_de_Resultado_1	Sim
Verifica_conteudo_do_resultado	Sim
Salva_Resultado	Sim
ConsultaInstituicao_1	Sim
ConsultaInstituicao_busca_feevale	Sim

Fonte: Elaborado Pelo autor

O teste unitário **AutenticaUsuario_E_Cria_Sessao** busca simular a autenticação de um usuário, ou seja, é realizada uma requisição para solicitaAutenticacaoUsuario, a fim de verificar se a lógica implementa está funcional, permitindo assim, este método ser liberado para a integração com o *front-end* do projeto.

A construção de um teste unitário pode ser realizada de maneira simples, mas é gerado um grande valor, pois torna-se fácil a identificação de erro caso a estrutura do projeto seja alterada. A fim de verificar a estrutura dos métodos, CadastraUsario e AlteraCadastraUsuario, foram implementados os testes unitários **Cadastra_Novo_Usuario** e **Altera_usuario_1**.

Estes, respectivamente, possuem os objetivos de cadastrar um novo usuário na base de dados e realizar alteração dos dados que foram cadastrados para um usuário

existente. Portanto, para realizar estes testes foi necessário criar dados fictícios, pois a realização de teste unitário possui o objetivo de validar as funcionalidades implementadas.

A fim de simplificar o entendimento da real utilização de um teste elaborado, é boa prática optar por nome sugestível ao seu objetivo. Pode ser citada a escolha do nome **Lista_Avatar_Teste_Deve_Retornar_Maior_Que_Zero**, onde é facilmente compreendido o que deve ocorrer cenário. Neste caso, é expressado que o retorno da requisição do método ListaAvatar deve conter pelo menos um registro.

Desta maneira, foram elaborados três métodos para a validação do método ConsultaAvatar, sendo eles: **ConsultaAvatar_Testa_Retorno_Erro**, **ConsultaAvatar_Testa_Retorno_Sucesso** e **ConsultaAvatar_Testa_Busca**. Portanto são enviadas requisições para o método ConsultaAvatar, afim de validar se existe um retorno de sucesso, um retorno de erro e integridade da busca.

O desenvolvimento dos testes **SolicitaCaso_Retorna_Caso_5** e **Solicita_Caso_Anaminesia**, têm como funcionalidade a verificação do retorno do método SolicitaCaso. No teste SolicitaCaso_Retorna_Caso_5 é esperado que o caso de teste retornado é igual ao id solicitado, já Solicita_Caso_Anaminesia é aguardado que o nome do caso de estudo que retornou seja igual à Anamnésia.

Da mesma forma, os testes unitários **PacoteCaso_Retorno_Quantidade_de_personagens_5** e **PacoteCaso_CasoEstudo_1,** esperam obter, respectivamente, os valores de 5 personagens e o id do caso de estudo igual a 1. Para realizar a implementação do teste unitário SalvaNodos_rede_1, foi criada uma rede bayesiana com nodos e valores fictícios, com o propósito de averiguar que as funcionalidades implementadas no método estão agindo conforme o esperado.

Com o intuito de validar o método infereNodo, foi desenvolvido o teste unitário **Faz_enferencia_nodo_1_Respota_VOCETEMCERTEZADISTO**. Neste teste é efetuada uma inferência em um nodo de uma RB fictícia. Como retorno para obter sucesso na validação, espera-se que resposta contida no nodo seja igual à “você tem certeza disto? ”.

Pensando em garantir a funcionalidade do método Resultado, foram implementados 2 testes unitários, sendo eles: VerificaExistencia_de_Resultado_1, Verifica_conteudo_do_resultado. Nestes testes, é averiguado o retorno da requisição,

onde no primeiro teste é verificada a existência de um resultado de simulação que contém o id resultado igual a 1. No segundo teste é analisado o conteúdo do resultado alcançado por um aluno fictício.

De modo semelhante a validação de um novo usuário, foi construído o teste unitário **Salva_Resultado**. Este teste tem como objetivo realizar uma requisição do tipo POST para o método, ResultadoRelatorio, buscando inserir, em um cenário fictício, um novo registro de relatório para um aluno que finalizou sua simulação.

Por fim, foram elaborados dois testes unitários para a validação do método de consultaInstituicao, no qual foram nomeados da seguinte forma, **ConsultaInstituicao_busca_feevale** e **ConsultaInstituicao_1**. Estes testes visam realizar a validação da busca por uma instituição através seu id. Na primeira validação é esperado que o nome da instituição seja igual a “Universidade Feevale”, já para a segunda validação é esperado que exista um retorno contendo um id igual a 1.

Além dos testes unitários aqui apresentados, foram testados e intregrados ao simulador os métodos de autenticação de usuário e de listagem de casos de estudos, ou seja, dois dos treze métodos desenvolvidos estão integrados. Este número baixo é reflexo direto do desenvolvimento do simulador, uma vez que este não encontra-se finalizado. Ao passo que novas cenas e etapas do simulador forem concluídas é possível integrar mais métodos aqui apresentados e validados. A integração destes dois métodos pode ser consultada no Anexo II, onde existe a imagem do simulador realizado a chamada do método para a camada de comunicação.

Visando validar e comprovar que a camada de comunicação desenvolvida é escalável foram criados testes de cargas. Estes testes tem como principal objetivo criar uma cenário similar ao real, ou seja, é simulado o acesso de diversos usuários simulando requisições para a camada de comunicação com a finalidade de verificar a sua estabilidade quando exposta a um grande número de requisições. Por tanto, foi utilizado como base na elaboração dos testes de carga, a documentação disponibilizada pela Microsoft no ano de 2010 (MICROSOFT,2010).

Desta maneira, foram definidos quatro cenários, os quais possuem a seguinte parametrização:

- Tempo de Execuação: 5 minutos
- Testes Escolhidos: Autenticação de ususário e Cadastro de Usuário

- Balanceamento entre testes: 50%

Além disto, foi difinido que cada um dos testes teria uma carga diferente, sendo elas respectivamente, 25, 50, 100 e 150. Os resultados dos testes de carga podem ser consultados no Anexo III deste trabalho.

No primeiro cenário com carga igual à 25 obteve-se um tempo médio de execução de teste igual a 0,021 segundos, onde o tempo médio para a execução da autenticação de usuário é igual 0,016 segundos, e para o cadastro de novos alunos o tempo médio é igual á 0,026 segundos. Totalizando um somatório de 101.364 mil requisições aos métodos no decorer dos cinco minutos estipulados como parâmetro.

O segundo cenário com carga igual a 50, foi obtido um tempo médio de execução de teste igual 0,066 segundos, cerca de 31% maior que o primeiro cenário. Neste cenário foram totalizadas 76.219 mil interações do dois métodos verificados. Onde o tempo médio de execução dos métodos de cadastro e de autenticação foi respectivamente, 0,082 e 0,051 segundos.

Para o terceiro cenário onde a carga foi definida com o valor de 100, obteve-se um total de interações igual à 69.864 mil, este valor é cerca de 9% menor que o valor obtido no cenário 2 .Um tempo médio de execução de testes igual á 0,089 segundos, onde o método de autenticação obteve o valor médio de execução de teste igual à 0,072 segundos e o método para cadastro de usuários obteve um tempo médio de 0,11 segundos.

No ultimo cenário a carga foi estipulada com o valor igual à 150. A escolha desta carga gerou um tempo médio de execução de teste igual à 0,077 segundos. Onde o método de autenticação de usuário obteve o valor de 0,088 segundos para cada teste excutado, e o método de cadastro de usuário gerou um valor de 0,067 segundos. Neste cenário foi gerado um total de 60.626 mil interações.

Outro dado importante que foi verificado no teste de carga é que nenhum dos métodos apresentou falha. Este valor é reflexo direto da aplicação de testes unitários para validar os métodos desenvolvidos para a camada de comnicação.

CONCLUSÃO

A simulação no meio educacional, em conjunto com o avanço tecnológico, cada vez mais possibilita a criação de simulações que são fidedignas a realidade que presenciamos hoje na área da saúde. Deste modo, a formação de profissionais com experiência na resolução de casos clínicos comuns, também como casos mais complexos e raros é bem vinda.

O *front-end* do Health Simulator se apresentou desafiador para o desenvolvimento de personagens e cenários, devido as variações apresentadas dos modelos. Por esta razão, fazer uso de uma filosofia e princípios foi essencial para este projeto. Além disso, o desenvolvimento individual do profissional é vidente, pois demanda do desenvolvedor um pensamento aguçado para que as práticas ágeis sejam desenvolvidas conforme o esperado.

O *back-end*, assim como o *front-end*, se demonstra como um desafio devido a divisão em três estágios (modelagem do conhecimento, interface de administração e serviço web de comunicação), formando uma estrutura robusta para gerenciar e prover conteúdo para a simulação a ser entregue aos alunos.

O fato de serem utilizadas redes bayesianas evidencia outro ponto importante, pois a catalogação de variáveis incertas da área de saúde, possibilitam tanto para o professor quanto para o aluno, um novo horizonte de possibilidades. É possível “brincar” com as informações das variáveis inseridas, gerando novos diagnósticos.

Como apresentando no capítulo 2 deste trabalho, um dos principais objetivos de um serviço *web* é prover uma alta escalabilidade e estar disponível para um maior número de clientes. Deste modo, ao desenvolver uma arquietura baseada em REST se mostra apropriada para o simulador Health Simulator. Os princípios de REST aplicados neste trabalho se demostram ideal para projeto de grande porte como é o caso do Health Simulator. No capítulo 3, foram evidenciados os métodos desenvolvidos que tiveram como base a arquitetura REST.

A escolha por testes unitários para realizar a validação, se mostrou extremamente importante na etapa de desenvolvimento, uma vez que possibilitou a identificação de erros de forma precoce, assim alterações que venha a ocorrer são rapidamente

averiguadas. Deste modo, é possível garantir que os métodos desenvolvidos atendem por completo a necessidade que gerou sua criação.

A utilização dos testes de carga para verificação da escalabilidade da camada de comunicação se mostrou muito eficiente e prático, uma vez que possível criar diversos cenários com parâmetros diferentes. Outro ponto imporante é que mesmo subindo o nível da carga o tempo médio de resposta dos métodos testados ficaram abaixo de meio segundo.

Os objetivos definidos no anteprojeto deste trabalho foram atingidos parcialmente. O objetivo geral foi definido como: realizar a conexão entre o *front-end* e o *back-end* do projeto Health Simulator, utilizando princípios *REST* para realizar as trocas de dados e informações durante a utilização do simulador, de forma segura e eficaz. Conforme o capítulo 3, o desenvolvimento da camada de comunicação fazendo uso de princípios REST, proporciona um tráfego de informações robusto e eficiente. Foi apresentada a solução implementada para criar um controle de autenticidade de usuários, com a finalidade de aumentar a segurança durante as simulações realizadas.

Além disto, foram definidos objetivos específicos: Realizar um estudo bibliográfico sobre os temas pertinentes a este trabalho; Integrar o *front-end* com o *back-end* do projeto *Health Simulator*; Explorar e corrigir falhas de segurança que possam existir entre a comunicação do *front-end* com o *back-end*; Validar a comunicação entre o *back-end* e o *front-end*. O estudo bibliográfico referente ao tema do projeto serviu como uma base solida para o desenvolvimento, pois possibilitou a compreensão das necessidades do *front-end*, resultando em uma camada de comunicação assertiva.

A utilização de testes unitários, possibilitou a identificação de erros e falhas, assim como realizar a validação durante o desenvolvimento dos métodos. Os erros de codificação que foram identificados ao longo do período de implementação foram corrigidos.

Referente ao objetivo específico: integrar o *front-end* com o *back-end* do projeto *Health Simulator,* está em andamento, pois o simulador não está finalizado. Deste modo, o objetivo é atendido parcialmente.

Entretanto, conforme o desenvolvimento avança, está sendo realizada a integração com métodos necessários. Além disto, são realizadas reuniões semanais para discutir

como estão sendo consumidos os métodos desenvolvidos e se existe a necessidade de novos métodos ou de alterações nos já desenvolvidos.

Por fim, podem ser definidos como trabalhos futuros, o desenvolvimento de uma documentação *online* que fique acessível e que contenha informações detalhadas sobre os métodos implementados, como, por exemplo: quais parâmetros são necessários para realizar uma requisição, qual é a resposta esperada, entre outras informações relevantes, com a finalidade de simplificar o aprendizado para as equipes que atuam no desenvolvimento do simulador. Além disso, do aperfeiçoamento dos métodos gerados, assim como de seus respectivos testes unitários. A crição de novos cenários de testes tanto para testes unitários como para testes de carga. A aplicação de um algoritmo de criptografia do controle de usuário, visando uma maior segurança também deve ser implementado.

REFERÊNCIAS BIBLIOGRÁFICAS

AKPAN, J. P. Issues associated with inserting computer simulations into biology instruction: a review of the literature. Electronic Journal of Science Education, Southwestern University, v. 5, n. 3, 2001.

AUMONT, J.; MARIE, M. **Dicionário teórico e crítico de cinema**. Papirus Editora, 2006.

BASS, J. **Revolutionizing Engineering Science through Simulation**. A Report of the National Science Foundation Blue Ribbon Panel on Simulation-Based Engineering Science. Virginia, USA: National Science Foundation. may/2006. 66p.

BLAKE, C.; SCANLON, E. **Reconsidering simulations in science education at a distance: features of effective use.** Journal of Computer Assisted Learning, v. 23, n. 6, p.491–502, 2007.

BRADLEY, P. The history of simulation in medical education and possible future directions. Medical Education, v. 40, n. 3, p.254-262, 2006.

BEZ, M. R. Construção de um modelo para o uso de simuladores na implementação de métodos ativos de aprendizagem nas escolas de medicina. 2013.

BOEHM, B.; HANSEN, W. J. **Spiral development: Experience, principles, and refinements.** Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2000.

CARAPETO, J. L. X. Design Estratégico e Scrum - Suas Relações para Processos de Projeto de Websites de Comunicação. 2012. Unisinos, 2012.

COHN, M. Desenvolvimento de Software com Scrum: Aplicando Métodos Ágeis com Sucesso. Porto Alegre: Bookman, 2011.

DEITEL P., DEITEL, H.; et al. Java para programadores, uma abordagem baseada em aplicativos. Editora: Bookman, 2012.

Erl, T., Carlyle, B., Pautasso, C., & Balasubramanian, R. SOA with REST: Principles, Patterns &Constraints for Building Enterprise Solutions with REST. Prentice Hall Press. (2012).

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures.** Doctoral dissertation, University of California, Irvine, 2000.

FOX, B. Animação em 3Ds Max 6 : Criação de Filmes CG do Conceito a Conclusão. Rio de Janeiro: Ciência Moderna, 2004. p. 461.

FOWLER, M. et al. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2012.

FOWLER, M. UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004.

GOMES, E. et al, **Health Simulator: A Produção do Front End.** IX Gamepad: Senário de Games e Tecnologia, Universidade Feevale, 2016

GRAÇA, P. J. B. O ciberataque como guerra de guerrilha: o caso dos ataques DoS/DDoS à Estónia, Geoórgia e ao Google-China. 2013. Tese de Doutorado. Instituto Superior de Ciências Sociais e Políticas.

GURGEL, M. Projetando espaços - Guia de arquitetura de interiores para áreas residenciais. 6. Ed. São Paulo: Senac, 2012.

GURGEL, M. **Projetando espaços - design de interiores.** 4. ed. São Paulo: Senac, 2011.

HECKEL, G. et al. Projeto Health: Produção de Personagens Tridimensionais. GAMEPAD VIII, 29 a 30 maio 2015.

HIGGS, J., Jones M, Loftus S, Christensen, N. **Clinical Reasoning in the Health Professions - 3º Ed**.2000:223–234, 2008.

JONG, T.; JOOLINGEN, W. R. Scientific discovery learning with computer simulations of conceptual domains. Review of Educational Research, v. 68, n. 2,

p.179–201, 1998.

LACERDA, G. S. D.; WILDT, D. D. F.; RIBEIRO, V. G. **Uma Introdução às Metodologias Ágeis de Software**. 2004.

Lean Institute. **LeanInstitute**. Disponível em: http://www.lean.org.br/ Acesso em: 9 outubro. 2016.

LIMA, A.; MEURER, H. Projeto de Personagens Tridimensionais e Virtuais: Validação e Adaptação de Metodologias. 1. ed. Porto Alegre: Uniritter, 2011.

LIMA, A. et al. **Projeto para desenvolvimento do Simulador Health Simulator.** Anais do Computer on the Beach, Florianópolis, 2015. 279-288.

MARONI, V. Construção de um motor de inferência para análise de desempenho em ambientes virtuais de aprendizagem aplicados ao ensino da medicina de família e comunidade. Dissertação (Mestrado em Ciências da Saúde) – Universidade Federal de Ciências da Saúde de Porto Alegre, Porto Alegre, 2013.

MICROSFOT, **MICROSFOT**,disponível em https://msdn.microsoft.com/en-us/library/ms182561(v=vs.90).aspx Acesso em: 23 junho 2017

MILLER, R. **Testing With a Fake DbContext.** 2012. Disponível em: <https://romiller.com/2012/02/14/testing-with-a-fake-dbcontext/>. Acesso em 29 mai 2016

MORCHE, G. et al. **Health Simulator: Produção de Cenários.** Feira de Iniciação Científica 2014: ciência, tecnologia e inovação, 2014. 321-346.

NETO, A. **Introdução a Teste de Software**, Engenharia de Software Magazine, V. 1, 2007.

OED Online, **Oxford English Dictionary**. Disponível em: http://dictionary.oed.com. Acessado em 27 de Outubro de 2016.

ORTON, E.; MULHAUSEN, P. **E-learning virtual patients for geriatric education**. Gerontology & Geriatrics Education, v. 28, n. 3, p.73-88, 2008.

PINHEIRO, D. et al. Redes Bayesianas como geração de conhecimento para games. GAMEPAD VIII, 29 a 30 maio 2015.

PRESSMAN, R. S. **Engenharia de Software.** [S.l.]: MAKRON Books, 1995.

RICHARDSON, L.; RUBY, S. **RESTful Web Services**. O'Reilly Media, 2007.

SAUDATE, A. REST: Construa API's inteligentes de maneira simples. Editora Casa do Código, 2014.

SAHEKI, A. H. Construção de uma rede Bayesiana aplicada ao diagnóstico de doenças cardíacas. 2005. Tese de Doutorado. Universidade de São Paulo.

SILBERSCHATZ, A.; et. al. **Sistemas de bancos de dados.** Tradução de Marília Guimarães Pinheiro e Cláudio César Canhette. 3.ed. São Paulo: Makron Books, 1999.

SOMMERVILLE, I. Engenharia de Software. 8. ed. São Paulo: Pearson Addison-Wesley, 2007. p. 552

SOMMERVILLE, I. **Arquitetura orientada a serviços. Engenharia de Software.** 9th ed., p.355–368. São Paulo: Person Prentice Hall, 2011.

STALLINGS, W; BROWN, L. **Segurança de computadores:** princípios e práticas. 2. ed. [s.i.]: Elsevier Editora Ltda, 2013. 744 p.

STANFORD, P. G. **Simulation in Nursing Education: a review of the research. The Qualitative Report.** Nova Southeastern University – Florida – USA. v. 15, n. 14. 2010.Disponível em: www.nova.edi/ssss/QR/QR 15-4/stanford.pdf. Acesso em janeiro de 2012.

STEPHEN, T. HTTP Essentials: Protocols for Secure, Scalable Web Sites. John Wiley & Sons, 2001.

TANENBAUM, A. S; STEEN, M. V. **Distributed Systems: Principles and Paradigms**. 2 ed. Pearson Prentice Hall, 2006.

TALIGENT, **Building Object-Oriented Frameworks.** Disponível em: < https://lhcb-comp.web.cern.ch/lhcb-comp/Components/postscript/buildingoo.pdf>, Acesso em: 03/06/2017

TELES, V. M. Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. São Paulo: Novatec. 2006.

TSUJI, H.; SILVA, R. H. A. Aprender e ensinar na escola vestida de branco: do modelo biomédico ao humanístico. São Paulo: Phorte, 2010. 240p.

UNITY, **UNITY**, disponível em <https://docs.unity3d.com/Manual/AssetWorkflow.html>, Acesso em : 03/11/2016

VELLOSO, F. Informática: Conceitos Básicos. 9. ed. Rio de Janeiro, Elsevier, 2014.

WARD, A. **Game Character Development**. Boston: Cengage, 2008.

WEBBER, J.; PARASTATIDIS, S.; ROBINSON; I. **REST in Practice: Hypermedia and Systems Architecture**. O'Reilly Media, 2010.

WOMACK, J. P. A Mentalidade Enxuta nas Empresas: Elimine o Desperdício e Crie Riqueza. Rio de Janeiro: Elsevier, 2004. p. 408

ZIV, A.; BD., S.; ZIV, M. **Simulation Based Medical Education: an opportunity to learn from errors.** Medical Teacher, v. 27, n. 3, p.193-199, 2005.

APENDICE A – ARQUIVO DE DECUPAGEM HEALTH SIMULATOR

Servidor

Engine

Menu Inicial	Menu Inicial
	Clique no icone na desktop iniciarSimulacao Passa informações de IP, Browser, Hora, SO, Idioma Cadastro de Novos Usuários carregaTela e lista de universidades Envia dados ao servidor (Backend) público clicar em Login ValidaCredenciais( ) envia(login e senha) recebe confimaçao (FOTO,NOME,VALIDADO)ou mensagem = LOGIN ou senha incorretos. tela de hud - com configuraçoes do jogo (AVATAR, legenda,iluminaçao, audio,graficos) - IN PLAY Configuração de Avatar Iniciar Aplicação (Casos Clínicos) Sair

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_recepcao
Câmera	Todas (Cut Scene)
Ações Introdução	- Executar Cut Scene da Cena Recepção typeScene_name_class_category (rec_vendramini_a_introducao)

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano Geral Investigação	Cenário - Carregar cenário (typeScene_name_class_category) (ex. consultório_vendramini_a) Vem posicionado com uma referência em seu trilho (X:0, Y:0, Z:0) - Carregar mesa de atendimento social, com um target incluso (ex. helper dentro do mesmo prefab da mesa). Isso guia a direção das demais animações. Personagens (Criação) genderAvatar_categoryAvatar_ethnicityAvatar_ageAvatar_biotypeA

	vatar Ex. man_patience_asian_1_fat Variáveis Adicionais nameAvatar textureAvatar supportingAvatar amountSupportingAvatar Médico - Carregar Modelo (ex. m_medic_afro_1_slim) - Animação de sentar na cadeira (genderAction_stageAction_actionAction_numberAction) (man_investigation_sit_1) (Inicia em 2s) Paciente (sgenderAction_stageAction_actionAction_numberAction) - Carregar Modelo (genderAvatar_categoryAvatar_ethnicityAvatar_ageAvatar_biotype Avatar) (ex. man_patience_asian_1_fat) - Animação de sentar na cadeira (man_investigation_sit_1) (Inicia em 2s) Câmera

	- Carregar câmera nomeada e posicionada (**typecam_name_camera**) (ex. consultório_vendramini_camera1) - Visão de Plano Geral - Movimento de Câmera Doly (E > D ou D < E, conforme target de mesa de atendimento); Velocidade: 4s - Troca a câmera (corte seco), quando personagens terminarem execução de animação de sentar.

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Close Investigação	Câmera - Carregar câmera nomeada e posicionada para close no Médico (**typeCam_name_camera**) (ex. consultório_vendramini_camera2) - Carregar câmera nomeada e posicionada para close no Paciente (**typeCam_name_camera**) (ex. consultório_vendramini_camera2) Médico (genderAction_stageAction_actionAction_numberAction) - Animação do tipo idle de conversa inicial (man_investigation_talk_1)

	Paciente (genderAction_stageAction_actionAction_numberAction) - Animação do tipo idle de conversa inicial (man_investigation_talk_1) Interface HUD Jogador por ir avançando os diálogos através de cliques na HUD por meio de cliques em setas. Ao passar um diálogo, câmera muda foco de um personagem para outro. Corta para próxima câmera - Quando encerrar diálogos pré-digitados no Backend.

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Close na **Ficha do Paciente**	Câmera - Carregar câmera nomeada e posicionada para close na ficha do paciente (**typeCam_name_camera**) (ex. consultório_vendramini_camera2). Esta câmera tem de fazer baking to texture do que vê e usa isso como imagem de fundo do HUD de Ficha. Em cima disto, a HUD joga um elemento em que aparecem os textos digitados no Backend. Ficha - Identificar o cenário, e a ele carregar ficha em papel ou tablet.

	- Animação papel/tablet sobre a mesa com movimento de scroll D>E, E<D, C>B, B<C

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano Médio Investigação; **Anamnese**	Câmera - Carregar câmera nomeada e posicionada para close na ficha do paciente (**typeCam_name_camera**) (ex. consultório_vendramini_camera3). HUD - XML carrega interface de HUD para Anamnese (temporariamente). - HUD é construído com itens cadastrados no banco, dispostos para marcar um apenas. Ao fazer isso, executa animação de idle-conversa e diálogo desenrola em baixo entre paciente e médico (médico pergunta, paciente responde). Repetir processo para cada pergunta. Animação (genderAction_stageAction_actionAction_numberAction) Médico - Animação do tipo idle de conversa inicial (man_investigation_talk_1)

	Paciente - Animação do tipo idle de conversa inicial (man_investigation_talk_1) Troca de pergunta - Finaliza diálogo automaticamente entre as personagens, e retorna HUD de Anamnese. Troca de cena - Quando todas as perguntas permitidas sejam feitas, ou - Tempo esgota, mas não encerra jogo, conta como pontuação negativa.

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano Médio Investigação: **Exames**	Câmera - Carregar câmera nomeada e posicionada para close na ficha do paciente (**typeCam_name_camera**) (ex. consultório_vendramini_ficha). HUD - XML carrega interface de HUD para Exames

	(temporariamente). - HUD é construído com itens cadastrados no banco, dispostos para marcar mais de um. O jogador marca todos os exames que deseja e finaliza. Animação genderAction_stageAction_actionAction_numberAction) Médico - Animação do tipo idle de conversa inicial (man_examination_talk_1) - Animação de escrever algo no papel (man_examination_write_1) -Animação de entrega de papel e aperto de mãos (man_examination_handshack_1) - Animação de observar a Ficha do Paciente (man_examination_idlehands_1) Paciente - Animação do tipo idle de conversa inicial (man_examination_talk_1) -Animação de entrega de papel e aperto de mãos (man_examination_handshack_1) Troca de cena Quando ambos terminam a animação de aperto de mãos.

Tomada de Câmera: sz Plano Geral Exames	Cenário - Carregar cenário (typeScene_name_class_category) (ex. exame_raiox) - Cut scene mostrando paciente “indo começar o exame x,y,z”.

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_recepcao
Câmera	Todas (Cut Scene)
Ações Retorno	- Executar Cut Scene de retorno da Cena Recepção - **typescene_name_class_return** (rec_vendramini_a_return)

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano Geral Diagnóstico	Cenário - Carregar cenário (typeScene_name_class_category) (ex. consultório_vendramini_a_principal)

	Câmera - Carregar câmera nomeada e posicionada (**typeCam_name_camera**) (ex. consultório_vendramini_camera4) - Visão de Plano Geral - Movimento de Câmera Doly (E > D ou D < E, conforme target de mesa de atendimento); Velocidade: 4s Médico - Carregar Modelo (enderAvatarr_categoryAvatar_ethnicityAvatar_ageAvatar_biotypeAvatar) (ex. man_medic_afro_1_slim) - Animação de sentar na cadeira e pegar exames do paciente genderAction_stageAction_actionAction_numberActio (ex. man_diagnostic_sit_2) (Inicia em 2s) Paciente - Carregar Modelo (enderAvatarr_categoryAvatar_ethnicityAvatar_ageAvatar_biotypeAvatar) (ex. man_patience_asian_1_fat) - Animação de sentar na cadeira e entregar exames ao médico genderAction_stageAction_actionAction_numberAction (ex. man_diagnostic_sit_2) (Inicia em 2s) Troca de cena - Quando ambas as personagens trocam os exames um com

	outro. - Corte seco para próxima câmera.
Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano Geral Diagnóstico	Ações do Jogador Jogador pode pedir para o médico para checar exames. Esta checagem é feita por sobre a mesma do médico mesmo, acessando a mesma câmera que chama a Ficha do Paciente. Jogador por pedir ao médico que mude o exame em foco na mesa (a animação varia se com papéis ou se com tablet). Câmera - Carregar câmera nomeada e posicionada para close na ficha do paciente (**typeCam_name_camera**) (ex. consultório_vendramini_ficha). Médico - Movimento de mãos sobre sua mesa teatrizando que está observando e manuseando documentos ou tablet em suas mãos, caso seus equipamentos contenham o tablet. Animações (genderAction_stageAction_actionAction_numberAction) - man_diagnostic_idlehands_2 - man_diagnostic_idlehands_3 Troca de cena - Quando o jogador sair da visualização dos exames por meio

	de menu.
Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano Médio Diagnóstico	Câmera - Carregar câmera nomeada e posicionada para plano médio, com foco no médico. (**typeCam_name_camera**) (ex. consultório_vendramini_focomedico1). HUD - XML carrega interface de HUD para Diagnóstico (temporariamente). - HUD é construído com itens cadastrados no banco, dispostos para marcar apenas um. Animação (genderAction_stageAction_actionAction_numberAction) Médico - Animação de espera, segurando resultados de exames (man_diagnostic_idle_1) Paciente - Animação de espera por diagnóstico (man_diagnostic_idle_2) Troca de cena -Corte seco quando jogador escolhe o diagnóstico e finaliza a

	ação por meio de menu.
Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano Médio Diagnóstico	Câmera - Carregar câmera nomeada e posicionada para plano médio no médico (**typeCam_name_camera**) (ex. consultório_vendramini_focomedico). HUD - XML carrega interface de HUD para Diagnóstico (temporariamente). - HUD é construído com itens cadastrados no banco, dispostos para marcar um apenas. Ao fazer isso, executa animação de idle-conversa e diálogo desenrola em baixo entre paciente e médico. Repetir processo para cada frase de diálogo de uma personagem por vez. Animação (genderAction_stageAction_actionAction_numberAction) Médico - Animação do tipo idle de conversa para diagnóstico (man_diagnostic_talk_1) Paciente - Animação do tipo idle de conversa para diagnóstico

	(man_diagnostic_talk_id) Troca de pergunta - Finaliza diálogo automaticamente entre as personagens, e retorna HUD de Diagnóstico. Troca de cena - Quando o diálogo acabar entre as personagens, corte seco para próxima câmera.

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano Médio Conduta	Câmera - Carregar câmera nomeada e posicionada para plano médio, com foco no médico. (**typeCam_name_camera**) (ex. consultório_vendramini_focomedico1). HUD - XML carrega interface de HUD para Conduta (temporariamente). - HUD é construído com itens cadastrados no banco, dispostos para marcar apenas um.

	Animação (genderAction_stageAction_actionAction_numberAction) Médico - Animação de explicação de conduta (médico pode pegar algum remédio amostra e demonstrar ao paciente (esta ação depende do tipo de diagnóstico identificado) (man_conduct_explain_1) Paciente - Animação de espera por diagnóstico (man_conduct_idle_1) Troca de cena -Corte seco quando jogador escolhe o diagnóstico e finaliza a ação por meio de menu.

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano	Câmera - Carregar câmera nomeada e posicionada para plano médio no médico

Médio Conduta	(**typeCam_name_camera**) (ex. consultório_vendramini_focomedico). HUD - XML carrega interface de HUD para Diagnóstico (temporariamente). - HUD é construído com itens cadastrados no banco, dispostos para marcar um apenas. Ao fazer isso, executa animação de idle-conversa e diálogo desenrola em baixo entre paciente e médico. Repetir processo para cada frase de diálogo de uma personagem por vez. Animação (genderAction_stageAction_actionAction_numberAction) Médico - Animação do tipo idle de conversa para diagnóstico (man_conduct_talk_1) Paciente - Animação do tipo idle de conversa para diagnóstico (man_conduct_talk_1) Troca de pergunta - Finaliza diálogo automaticamente entre as personagens, e retorna HUD de Diagnóstico. Troca de cena

	- Quando o diálogo acabar entre as personagens, corte seco para próxima câmera.

Cena/ Cenário	typeScene_nameScene_classScene_categoryScene Ex.: consultorio_vendramini_a_principal
Tomada de Câmera: Plano Geral Resumo	Câmera - Carregar câmera nomeada e posicionada para plano geral no cenário (**typeCam_name_camera**) (ex. consultório_vendramini_fococenario). - HUD exibe página com informações estatísticas no fim do jogo. Jogador por escolher ações no final desta página.

ANEXO I – TESTES UNITÁRIOS DESENVOLVIDOS

```
[TestMethod]
public void Faz_enferencia_nodo_1_Respota_VOCETEMCERTEZADISTO()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var rede = @"d:\json_rede.json";
    var result = controller.json(idUser, idSession, 1,rede);
    Assert.AreEqual("VOCÊ TEM CERTEZA DISSO?", result.msg.FirstOrDefault().msg);
}

[TestMethod]
public void SalvaNodos_rede_1()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var rede = @"d:\json_rede.json";
    var result = controller.salvanodos(rede);
    Assert.AreEqual("Sucesso",result.status);
}
[TestMethod]
public void PacoteCaso_CasoEstudo_1()
{
    int idUser = 1;
```

```
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var result = controller.Pacotecaso(idUser, idSession, 1);
    var resultado = result.pacote.FirstOrDefault();
    Assert.AreEqual(1,resultado.idCaseStudy);
}
[TestMethod]
public void SolicitaCaso_Retorna_Caso_5()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var result = controller.SolicitaCaso(idUser, idSession, 5);
    var resultado = result.caso.idCaseStudy;
    Assert.AreEqual(5, result.caso.idCaseStudy);
}
[TestMethod]
public void Solicita_Caso_Anaminesia()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var result = controller.SolicitaCaso(idUser, idSession,5);
    Assert.AreEqual("Anaminésia", result.caso.nameCaseStudy);
}
[TestMethod]
public void PacoteCaso_Retorno_Quantidade_de_personagens_5()
{
```

```
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var result = controller.Pacotecaso(idUser, idSession, 5);
    var resultado = result.pacote.FirstOrDefault();
    Assert.AreEqual("5", resultado.amountSupportingCharacter.ToString());
}

[TestMethod]
public void ConsultaInstituicao_1()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var result = controller.ConsultaInstituicao(idUser, idSession, 1);
    Assert.IsTrue(result.listaInstituicao.Count() > 0);
}
[TestMethod]
public void ConsultaInstituicao_busca_feevale()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var result = controller.ConsultaInstituicao(idUser, idSession, 1);
    Assert.AreEqual("Universidade Feevale",
result.listaInstituicao.First().nameInstitution);
}
[TestMethod]
```

```
public void Faz_enferencia_nodo_1_Respota_VOCETEMCERTEZADISTO()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var rede = @"d:\json_rede.json";
    var result = controller.json(idUser, idSession, 1,rede);
    Assert.AreEqual("VOCÊ TEM CERTEZA DISSO?", result.msg.FirstOrDefault().msg);
}
[TestMethod]
public void SalvaNodos_rede_1()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var rede = @"d:\json_rede.json";
    var result = controller.salvanodos(rede);
    Assert.AreEqual("Sucesso",result.status);
}
[TestMethod]
public void VerificaExistencia_de_Resultado_1()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var result = controller.RetornaResultado(idUser, idSession);
    Assert.AreEqual(1, result.retornaresultado.idResult);
}
```

```
[TestMethod]
public void Verifica_conteudo_do_resultado()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    var result = controller.RetornaResultado(idUser, idSession);
    Assert.AreEqual("RELATÓRIO SOBRE O ALUNO", result.retornaresultado.report);
}
[TestMethod]
public void Salva_Resultado()
{
    int idUser = 1;
    Guid idSession = new Guid("B50E91DC-F0EE-4257-BC3A-36535D6B7148");
    DadosResultado dados = new DadosResultado();
    dados.idResult = 1;
    dados.idUser = idUser;
    dados.observation = "Observação";
    dados.report = "Seu Teste Funcionou";
    dados.idCaseStudy = 5;
    dados.idNetwork = 1;
    dados.Idsession = idSession;
    var result = controller.SalvaRelatorio(dados);
    Assert.AreEqual("Sucesso", result.status);
}
[TestMethod]
```

```
public void AutenticaUsuario_E_Cria_Sessao()
{
    string loginUser = "ADMIN";
    string senha = "1234";
    var result = controller.SolicitaAutenticacaoUsuario(loginUser, senha);
    Assert.IsNotNull(result.dados.idSession);
}
[TestMethod]
public void Altera_usuario_1()
{
    string loginUser = "adm";
    string senha = "adm";
    AlteraUsuarioDados dados = new AlteraUsuarioDados();
    dados.idUser = 1;
    dados.birthDayUser = Convert.ToDateTime("15/05/1992");
    dados.emailUser = "emailteste@usuer.com.br";
    dados.functionUser = "admin";
    dados.genderUser = "M";
    dados.loginUser = "ADMIN";
    ados.nameUser = "FELLIPE";
    dados.phoneUser = "51982090630";
    dados.idInstitution = 1;
    dados.photoUser = "caminhofoto";

    var result = controller.AlteraUsuario(dados);
```

```
    Assert.AreEqual("Sucesso", result.status);
}
[TestMethod]
public void Cadastra_Novo_Usuario()
{
  CadastraUsuarioDados dados = new CadastraUsuarioDados();
  dados.phoneUser = "51982090630";
  dados.emailUser = "meu@email.com.br";
  dados.birthDayUser = Convert.ToDateTime("15/05/1992");
  dados.photoUser = "caminhofoto";
  dados.functionUser = "Estudante";
  dados.loginUser = "0125788";
  dados.nameUser = "Felipe Fernadnes Lorenzoni";
  dados.passwordUser = "NãoTeconto";
  dados.genderUser = "M";
  dados.idInstitution = 1;
  var result = controller.CadastraUsuario(dados);
  Assert.AreEqual("Sucesso", result.status);
}
```

ANEXO II – INTEGRAÇÃO FRONT-END


```
[HttpGet]
[Route("solicitaAutenticacaoUsuario")]
public SolicitaAutenticacaoUsuarioResposta SolicitaAutenticacaoUsuario(string loginUser, string senhaUser)
```

loginUser "admin" senhaUser "admin"


```
[HttpGet]
[Route("consultaListaCaso")]
0 references | Felipe Lorenzoni, 24 days ago | 1 author, 1 change
public ConsultaCasoResposta ConsultaListaCaso(int idUser, Guid IdSession)
```

idUser 1 IdSession {66389497-72be-4708-acef-ef0f27ca4076}

ANEXO III – TESTES DE CARGA

Test	Scenario	Total	Passed	Failed	Test/sec	Test Time
AutenticaUsuario_E_Cria_Sessao	Cenário 1	50.526	50.526	0	168	0,016
Cadastra_Novo_Usuario	Cenário 1	50.838	50.838	0	169	0,026
AutenticaUsuario_E_Cria_Sessao	Cenário 2	38.537	38.537	0	128	0,051
Cadastra_Novo_Usuario	Cenário 2	37.682	37.682	0	126	0,082
AutenticaUsuario_E_Cria_Sessao	Cenário 3	35.063	35.063	0	117	0,072
Cadastra_Novo_Usuario	Cenário 3	34.801	34.801	0	116	0,11
AutenticaUsuario_E_Cria_Sessao	Cenário 4	30.298	30.298	0	101	0,067
Cadastra_Novo_Usuario	Cenário 4	30.328	30.328	0	101	0,088

Load Test Summary

Test Run Information

Load test name	25Alunos_Cadastra_Autentica
Description	
Start time	25/06/2017 13:55:50
End time	25/06/2017 14:00:50
Warm-up duration	00:00:00
Duration	00:05:00
Controller	Local run
Number of agents	1
Run settings used	Run Settings1

Overall Results

Max User Load	25
Tests/Sec	338
Tests Failed	0
Avg. Test Time (sec)	0,021
Transactions/Sec	0
Avg. Transaction Time (sec)	0
Pages/Sec	-
Avg. Page Time (sec)	-
Requests/Sec	-
Requests Failed	-
Requests Cached Percentage	-
Avg. Response Time (sec)	-
Avg. Content Length (bytes)	-

Load Test Summary

Test Run Information

Load test name	50Alunos_Cadastra_Autentica
Description	
Start time	25/06/2017 14:06:26
End time	25/06/2017 14:11:26
Warm-up duration	00:00:00
Duration	00:05:00
Controller	Local run
Number of agents	1
Run settings used	Run Settings1

Overall Results

Max User Load	50
Tests/Sec	254
Tests Failed	0
Avg. Test Time (sec)	0,066
Transactions/Sec	0
Avg. Transaction Time (sec)	0
Pages/Sec	-
Avg. Page Time (sec)	-
Requests/Sec	-
Requests Failed	-
Requests Cached Percentage	-
Avg. Response Time (sec)	-
Avg. Content Length (bytes)	-

Load Test Summary

Test Run Information

Load test name	100Alunos_Cadastra_Autentica
Description	
Start time	25/06/2017 14:15:07
End time	25/06/2017 14:20:07
Warm-up duration	00:00:00
Duration	00:05:00
Controller	Local run
Number of agents	1
Run settings used	Run Settings1

Overall Results

Max User Load	100
Tests/Sec	233
Tests Failed	0
Avg. Test Time (sec)	0,089
Transactions/Sec	0
Avg. Transaction Time (sec)	0
Pages/Sec	-
Avg. Page Time (sec)	-
Requests/Sec	-
Requests Failed	-
Requests Cached Percentage	-
Avg. Response Time (sec)	-
Avg. Content Length (bytes)	-

Load Test Summary

Test Run Information

Load test name	150Alunos_Cadastra_Autentica
Description	
Start time	25/06/2017 14:27:43
End time	25/06/2017 14:32:43
Warm-up duration	00:00:00
Duration	00:05:00
Controller	Local run
Number of agents	1
Run settings used	Run Settings1

Overall Results

Max User Load	150
Tests/Sec	202
Tests Failed	0
Avg. Test Time (sec)	0,077
Transactions/Sec	0
Avg. Transaction Time (sec)	0
Pages/Sec	-
Avg. Page Time (sec)	-
Requests/Sec	-
Requests Failed	-
Requests Cached Percentage	-
Avg. Response Time (sec)	-
Avg. Content Length (bytes)	-