

Curso de Análise e Desenvolvimento de Sistemas

Pedro Bernardo de SOUSA 0030481711006

Weuller Júnior Souza Bessa 0030481621040

Vítor Andrade Marques da Silva 0030481511040

AViS – Alloy Virtual Space

Documentação de Desenvolvimento de Software

Sorocaba

Dezembro – 2019



Pedro Bernardo de SOUSA 0030481711006

Weuller Júnior Souza Bessa 0030481621040

Vítor Andrade Marques da Silva 0030481511040

AViS – Alloy Virtual Space

Documentação de Desenvolvimento de Software

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Sorocaba – FATEC, sob orientação de **M.ª** **Maria Angélica Calixto de Andrade Cardieri**, como parte dos pré-requisitos para obtenção do título de *Tecnólogo em Análise e Desenvolvimento de Sistemas*

Sorocaba

Dezembro – 2019

Dedicatória

Agradecemos às nossas famílias pelo tenaz apoio oferecido generosamente ao longo deste curso e, mais precisamente, durantes os meses de esforço que renderam este trabalho. Sem o alto apreço por ciência e educação compartilhado pelas nossas três famílias, esta grande empreitada não teria sido possível.

Agradecimentos

Muito obrigado a todo corpo docente, por ter sido em nós uma força transformadora ao longo desses três anos. Agradecimentos muito especiais a Me. Ana Carolina Camargo, Me. Cesar Munari, Prof. Jefferson Blaitt e Me. Maria Angélica Cardieri, nossa orientadora, pelos esforços de mentoria muito além de suas funções. Obrigado por terem notado nosso módico potencial, quando jazia escondido por nossa incipiência.

Resumo

A Alloy City Linguistics desenvolveu e mantém uma plataforma de ensino e aprendizado de francês baseada em tecnologias web. O objetivo do grupo AViS até o final do curso ADS dos participantes é desenvolver um cliente para essa plataforma existente. O cliente vai se chamar AViS - Alloy Virtual Space, e vai se apoiar na RESTful API disponibilizada pela Alloy para o projeto. O cliente AViS será capaz de simular um ambiente 3D onde professor e aluno poderão interagir de maneira mais próxima de interações humanas presenciais, isto é, com a impressão de compartilhar o lugar, e não apenas o momento.

**Palavras-chave:** AViS. Plataforma. EAD. Ambiente.Tri-dimensional.

Lista de Figuras

Figura 1 – Visão geral da gestão de pessoas 10

Figura 2 – Enfoque sistêmico nas organizações 12

Figura 3 – Esquema da estratégia empresarial e gestão de pessoas 13

Figura 4 – Fronteiras virtuais nas organizações 24

Figura 5 – Gestão das tecnologias da informação e gestão de pessoas 25

Lista de Tabelas

Tabela 1 – Principais mudanças na Área de Recursos Humanos (PricewaterhouseCoopers) 17

Tabela 2 – Principais mudanças na Área de Recursos Humanos (Chiavenato) 22

Tabela 3 – Resumo do Estudo de Caso 1 29

Tabela 4 – Ferramentas de T.I. e seu relacionamento com os processos da ARH (estudo de caso 1) 30

Tabela 5 – Ferramentas de T.I. e seu relacionamento com os processos da ARH (estudo de caso 2) 36

Índice

[1. Introdução 6](#_Toc22563831)

[2. Embasamento teórico. 7](#_Toc22563832)

[3. Planejamento Inicial do Software 8](#_Toc22563833)

[3.1. Situação Atual 8](#_Toc22563837)

[3.1.1. Problemas Encontrados 8](#_Toc22563838)

[3.1.2. Aplicativos Disponíveis no mercado (estado da arte) 8](#_Toc22563839)

[3.2. Objetivos do Projeto 9](#_Toc22563840)

[3.3. Riscos do Projeto AViS 9](#_Toc22563841)

[3.3.1. Limitações Operacionais 9](#_Toc22563842)

[3.3.2. Considerações Legais 9](#_Toc22563843)

[3.3.3. Considerações de Hardware / Software / Rede 10](#_Toc22563844)

[3.3.4. Políticas Organizacionais 10](#_Toc22563845)

[3.3.5. Riscos 10](#_Toc22563846)

[4. Análise de Requisitos 12](#_Toc22563850)

[4.1. Descrição da técnica utilizada para levantamento dos requisitos 12](#_Toc22563852)

[4.2. Situação Proposta 12](#_Toc22563853)

[4.3. Requisitos Funcionais 12](#_Toc22563854)

[4.3.1. Casos de Uso 13](#_Toc22563855)

[4.4. Requisitos Não Funcionais 16](#_Toc22563856)

[4.4.1. Tempo de resposta 16](#_Toc22563857)

[4.4.2. Uso de memória 17](#_Toc22563858)

[4.4.3. Uso de espaço em disco 17](#_Toc22563859)

[4.4.4. Uso de recursos de processamento no servidor 17](#_Toc22563860)

[5. Projeto Detalhado do Software 18](#_Toc22563861)

[5.1. Arquitetura da aplicação proposta 18](#_Toc22563863)

[5.2. Tecnologias utilizadas e APIs 21](#_Toc22563864)

[5.2.1. Tecnologias 21](#_Toc22563865)

[5.2.2. APIs Utilizadas 22](#_Toc22563866)

[5.3. Componentes do SW 23](#_Toc22563867)

[5.4. Diagrama de Classes 24](#_Toc22563868)

[5.5. Considerações sobre o Banco de Dados Utilizado 25](#_Toc22563869)

[5.6. Diagrama de Sequência 26](#_Toc22563870)

[5.7. Diagrama Pacotes 26](#_Toc22563871)

[5.8. Diagrama Estado 26](#_Toc22563872)

[5.9. Interfaces com o usuário 27](#_Toc22563873)

[6. Implementação 31](#_Toc22563874)

[7. Projeto de Teste 32](#_Toc22563875)

[8. Instalação do Software 32](#_Toc22563876)

[9. Análise dos Resultados 32](#_Toc22563877)

[10. Conclusão 32](#_Toc22563878)

# Introdução

Com o propósito de proporcionar uma experiência de aprendizado à distância mais abrangente, a plataforma do projeto AViS, busca fornecer aos usuários ferramentas em que os mais diversos aspectos da comunicação possam ser explorados durante a experiência de uso.

A comunicação humana natural se dá, não apenas através das palavras ditas e escritas. Aspectos da linguagem corporal e da entonação complementam este processo entre emissor e receptor. Assim, oferecer uma ferramenta capaz de abordar, da forma mais abrangente possível tais aspectos no processo de ensino à distância, proverá aos usuários, um significativo avanço na qualidade dos estímulos sensoriais, proporcionando que mais das informações oferecidas possam ser absorvidas.

Dois objetivos iniciais do projeto são, oferecer aos alunos um ambiente virtual em que a experiência de aprendizado aconteça como se aluno e professor partilhassem da mesma localização e, oferecer aos professores a possibilidade de receber dos alunos um volume maior de informações para avaliá-los e assim poder aprimorar suas aulas e conteúdo.

Como possibilidades a serem implementadas a longo prazo, o projeto AViS, poderá se tornar uma plataforma unificada capaz de oferecer suporte ao ensino de diversos idiomas e de forma universal, compatível com todos os principais sistemas operacionais.

# Embasamento teórico.

# Planejamento Inicial do Software



## Situação Atual

TODO: Vítor

### Problemas Encontrados

TODO: Vítor

### Aplicativos Disponíveis no mercado (estado da arte)

Atualmente há diversos sistemas utilizados para ensino à distância, muitos deles são amplamente utilizados, outros estão ainda em desenvolvimento. Esses sistemas são categorizados como AVAs (Ambiente Virtual de Aprendizagem), ou seja, sistema de apoio ao ensino a distância proporcionando conteúdos e recursos necessários para total ou parcial aproveitamento dos cursos. Alguns dos utilizados no mercado são citados a baixo:

* Amadeus LMS

O Amadeus é um LMS - do inglês, “Learning Manager System”, é um software de ensino a distância, EaD, propõe o conceito de blended learning, ou b-learning, é um derivado do e-learning, o que inclui situações presenciais.

* Moodle

Moodle é um software que permite criação de cursos num contexto de b-learning ou e-learning amplamente utilizado em de EaD.

* Blackboard Collaborate

Blackboard é uma plataforma para ensino a distância e presencial permitindo que os alunos se comuniquem com seus professores utilizando chat em vídeo

* Second Life

Second Life é um jogo que simula a vida real, em 2014, pesquisadores da Texas A&M University e da Florida Institute of Technology, utilizaram do jogo como forma de engajar os alunos no aprendizado em um ambiente virtual.

## Objetivos do Projeto

O principal objetivo acadêmico do projeto é estudar a viabilidade, técnica e comercial, da implementação de uma ferramenta de interação remota e síncrona, que ofereça melhor sensação de presença em contextos educacionais, através da renderização tridimensional de um ambiente compartilhado simulado.

Os principais desafios encontrados podem ser divididos em duas categorias: **desafios tecnológicos de implementação** e **problemáticas interdisciplinares**.

De um ponto de vista técnico, a solução que se propõe é bastante desafiadora. Especialmente em relação aos estudos de caso tipicamente explorados no curso de ADS. Não é comum

## Riscos do Projeto AViS

Partindo da premissa de que a viabilidade de um projeto depende de uma boa avaliação dos riscos e das formas de vencê-los, para o projeto AViS, consideramos como riscos:

### Limitações Operacionais

Entre as limitações operacionais que podem interferir no correto funcionamento da aplicação, podemos citar:

* Oscilações na largura de banda e/ou interrupção da conexão com a internet de um ou mais usuários ativos em um ambiente virtual durante suas interações;
* Falta de energia que afete o servidor ou um dos usuários ativos no ambiente virtual;
* Incompatibilidade de hardware ou software (sistema operacional) após uma atualização do sistema operacional;
* Indisponibilidade do servidor de hospedagem dos serviços necessários à aplicação.

### Considerações Legais

Sob aspectos legais, o uso da aplicação será baseado nos termos de licenças de código aberto. Entretanto, a aplicação proverá a cada usuário, acesso à imagem e voz de cada um dos demais usuários que partilhem do mesmo ambiente virtual. Assim, nos termos e condições de uso, cada usuário deverá tomar ciência e aceitar sua total responsabilidade em relação à captura e/ou divulgação indevidas dos fluxos de áudio e/ou vídeo de outros usuários.

TODO: nossa licença

TODO: licenças dos diferentes componentes de software utilizados no projeto (UE4, OpenCV, etc.)

### Considerações de Hardware / Software / Rede

A aplicação foi desenvolvida com foco no sistema operacional Windows 10 de 64 bits, as demandas em relação ao que esse sistema operacional exige, especificam o mínimo necessário ao bom funcionamento da aplicação. Entretanto, para o pleno uso das funcionalidades da aplicação, exige-se também um microfone e uma webcam para capturar a voz e a imagem do usuário, além desses itens de hardware, a estação de trabalho deve dispor de uma conexão com a internet, quanto à largura de banda da conexão, um mínimo de 1Mbps para download e 1Mbps de upload.

Para o funcionamento do MVP, o projeto AViS deve contar com a porta 7777 aberta no servidor da aplicação para os protocolos UDP e TCP. Não há a necessidade de manipulação das configurações do roteador ou das políticas do firewall nas estações de trabalho dos usuários.

### Políticas Organizacionais

TODO: estudar e descrever implicações/impactos em organizações que vierem a adotar uma ferramenta baseada nas tecnologias demonstradas neste trabalho.

### Riscos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PLANO DE RISCOS** | | | | |
|
| **Risco** | **Probab (P)** | **Impacto (I)** | **RiscoTotal (P x I)** | **Tratamento** |
|
| Oscilações / interrupções do provedor de internet | 5 | 4 | 20 | Buscar a opção mais estável dentre os provedores disponíveis. |
| Falta de energia | 2 | 4 | 8 | Compra de nobreaks. |
| Problemas de compatibilidade com atualizações do SO | 4 | 3 | 12 | Antes de cada atualização, criar um ponto de restauração do sistema. |
| Indisponibilidade do servidor de hospedagem | 2 | 5 | 10 | Verificar a opção mais confiável dentro do orçamento. |
| Uso de imagem indevida entre os usuários | 2 | 5 | 10 | Estabelecer um documento de termos e condições de uso. |
| Especificações incompatíveis da estação de trabalho | 1 | 5 | 5 | Realizar as adequações necessárias de hardware, software e acesso à internet. |
| Indisponibilidade da porta 7777 para os protocolos UDP e TCP | 1 | 5 | 5 | Realizar a configuração necessária para a operação. |

|  |  |
| --- | --- |
| **Probabilidade e Impacto** | |
| **Valor** | **Descrição** |
| 1 | Muito baixa |
| 2 | Baixa |
| 3 | Média |
| 4 | Alta |
| 5 | Muito alta |



# Análise de Requisitos



## Descrição da técnica utilizada para levantamento dos requisitos

## Situação Proposta

*Descrever o que será feito sem descrever como.*

*Descrever a proposta do que será desenvolvido em alto nível.*

Este MVP tem como objetivo melhorar a forma de comunicação e interação entre os utilizadores do sistema destinado a posteriormente atender necessidades na área de ensino de idiomas, em um ambiente tridimensional onde os usuários podem se comunicar e se ver utilizando de avatares para suas representações de posicionamento bem como expressão facial no ambiente virtual.

## Requisitos Funcionais

**RF1 – Andar**

O usuário poderá movimentar seu avatar no espaço virtual da sala de aula, utilizando as teclas W, A, S e D do teclado. A posição de todas as instâncias remotas de um determinado avatar deverão ser sincronizadas conforme os movimentos registrados pela instância local, de forma que todos os usuários conectados à sala virtual vejam os demais em suas devidas posições.

**RF2 – Olhar**

O usuário poderá modificar os ângulos, em dois eixos, para ajustar o ponto de vista em que observa o ambiente virtual através de movimentos do mouse. Por exemplo, ao movimentar o mouse para frente, o ângulo de visão no eixo horizontal diminuirá, para que olhe-se para baixo. Ao movimentar o mouse para um lado, o ângulo de visão no eixo vertical será ajustado. Os ajustes serão sincronizados entre todas as instâncias conectadas à sala virtual.

Este requisito atende a necessidade que o usuário terá de observar o ambiente virtual à sua volta.

**RF3 – Falar**

O usuário terá sua voz capturada pelo sistema e transmitida diretamente para as instâncias remotas. Cada instância remota reproduzirá o som recebido ajustando continuamente a posição de origem do som, no mecanismo de áudio posicional do ambiente virtual, para que ela coincida com a posição da cabeça do avatar correspondente a instância onde o som fora capturado.

**RF4 – Transmitir expressão facial**

A expressão facial do usuário será continuamente capturada pelo sistema, enquanto ele estiver com a webcam ligada. As imagens correspondentes ao rosto do usuário serão transmitidas diretamente às instâncias remotas. Cada instância remota aplicará a imagem mais recente disponível no rosto do avatar correspondente à instância onde a imagem fora capturada.

### Casos de Uso



|  |  |  |
| --- | --- | --- |
| Caso de Uso | FR1: Andar | |
| Ator Principal | Usuário local | |
| Atores Secundários | Usuários remotos | |
| Pré-Condição | O usuário está conectado a uma sala virtual | |
| Pós-Condição | Posição do avatar sincronizada entre todas as instâncias conectadas | |
| Ações do Ator | | Ações do Sistema |
| 1 – Pressiona teclas de movimento (W, A, S, D) | |  |
|  | | 2 – Envia comando ao servidor |
|  | | 3 – Servidor calcula nova posição do avatar |
|  | | 4 – Servidor envia novas coordenadas às instâncias conectadas |
|  | | 5 – Posição do avatar é atualizada |

|  |  |  |
| --- | --- | --- |
| Caso de Uso | FR2: Olhar | |
| Ator Principal | Usuário local | |
| Atores Secundários | Usuários remotos | |
| Pré-Condição | O usuário está conectado a uma sala virtual | |
| Pós-Condição | 1 - Posição da cabeça do avatar sincronizada entre todas as instâncias conectadas  2 – Ponto de vista do ator principal no ambiente virtual ajustado | |
| Ações do Ator | | Ações do Sistema |
| 1 – Movimenta o mouse | |  |
|  | | 2 – Envia comando ao servidor |
|  | | 3 – Servidor calcula novos ângulos, nos dois eixos |
|  | | 4 – Servidor envia novos ângulos às instâncias conectadas |
|  | | 5 – Instância local atualiza ângulos do ponto de vista do ator principal |
|  | | 6 – Instâncias remotas atualizam ângulo de inclinação da cabeça do avatar |
|  | | 7 – Instâncias remotas atualizam ângulo do avatar no eixo vertical |

|  |  |  |
| --- | --- | --- |
| Caso de Uso | FR3: Falar | |
| Ator Principal | Usuário local | |
| Ator Secundário | Usuários remotos | |
| Pré-Condição | O usuário está conectado a uma sala virtual | |
| Pós-Condição | Todos os usuários conectados à sala virtual ouvem a voz do ator principal | |
| Ações do Ator | | Ações do Sistema |
| 1 – Usuário fala | |  |
|  | | 2 – Captura áudio com microfone padrão |
|  | | 3 – Fragmenta áudio com base em limiar de volume |
|  | | 4 – Compacta o fragmento de áudio |
|  | | 5 – Envia o fragmento de áudio às instâncias remotas |
|  | | 6 – Instância remota descompacta o áudio |
|  | | 7 – Instância remota reproduz o áudio, utilizando a posição da cabeça do avatar correspondente à instância de origem do som, para configurar o mecanismo de áudio posicional |

|  |  |  |
| --- | --- | --- |
| Caso de Uso | FR4: Transmitir expressão facial | |
| Ator Principal | Usuário local | |
| Ator Secundário | Usuários remotos | |
| Pré-Condição | O usuário está conectado a uma sala virtual | |
| Pós-Condição | Usuários conectados à sala virtual veem os rostos uns dos outros | |
| Ações do Ator | | Ações do Sistema |
| 1 – Usuário aciona Webcam | |  |
|  | | 2 – Captura imagem |
|  | | 3 – Identifica rosto do ator principal |
|  | | 4 – Recorta imagem utilizando as coordenadas do rosto identificado |
|  | | 5 – Compacta a imagem |
|  | | 6 – Envia a imagem às instâncias remotas |
|  | | 7 – Instância remota descompacta a imagem |
|  | | 8 – Instância remota aplica a imagem no rosto do avatar correspondente à instância onde a imagem fora capturada |

## Requisitos Não Funcionais

### Tempo de resposta

Para oferecer as funcionalidades desejadas, é importante que a comunicação entre instâncias cliente seja a menor possível. Não é razoável decidir um valor fixo para o tempo de viagem dos pacotes, já que, num cenário real, este tempo vai depender de uma rede probabilística (a Internet), e da localização geográfica dos usuários conectados pelo sistema.

Entretanto, no quadro da demonstração que se pretende fazer ao final do projeto, espera-se que a latência entre os dois computadores conectados esteja na ordem de dezenas de milissegundos, enquanto a latência entre as instâncias cliente e a instância no servidor remoto, localizado em um *datacenter*, em São Paulo, seja de até 200 milissegundos.

### Uso de memória

O programa, tanto no servidor, quanto no cliente, não deve utilizar mais do que 500 MB de memória principal.

Além disso, dada a liberdade de manipulação de memória oferecida por C++, medidas específicas devem ser tomadas para evitar vazamento de memória(*memory leak*).

### Uso de espaço em disco

O arquivo de instalação não deve ultrapassar 200 MB.

O espaço ocupado pelo sistema cliente, uma vez instalado, não deve ultrapassar 400 MB.

### Uso de recursos de processamento no servidor

A instância em execução no servidor não deve ultrapassar 20% dos recursos de processamento da máquina virtual, exceto durante a instanciação do programa.

# Projeto Detalhado do Software

A partir de uma análise preliminar da ideia, decidiu-se dividir o projeto em duas grandes fases, que correspondem a dois semestres. A fase academicamente mais relevante é a primeira, durante a qual ocorrem os esforços de pesquisa e desenvolvimento. O principal objetivo dessa primeira fase é desenvolver um Produto Mínimo Viável, ou MVP, para demonstrar a viabilidade da ideia, de um ponto de vista técnico. Nessa fase, deve-se demonstrar que as tecnologias escolhidas para o projeto são apropriadas e serão suficientes. Além disso, o MVP deverá conter funcionalidades centrais da ideia, da forma mais minimalista possível, para que o escopo permaneça concentrado nos desafios técnicos menos usuais e mais relevantes.

As funcionalidades centrais de um futuro produto são a transmissão de voz e expressão facial, assim como a simulação do ambiente em 3D. Essas são, portanto, as funcionalidades que devem estar no MVP. Por outro lado, em um produto pronto para o mercado, será necessário um sistema de autenticação, navegação de cursos disponíveis, uma agenda, um sistema de updates automáticos, um processo bem definido de desenvolvimento contínuo, com testes automáticos e lançamento, compatibilidade com os principais sistemas operacionais etc. Um projeto dessa magnitude precisa de mais do que 3 desenvolvedores de software inexperientes e 6 meses de trabalho, mesmo que com orientação docente. O MVP, desenvolvido em um semestre letivo, vai demonstrar a ideia central e preparar o caminho para que o projeto continue, no semestre seguinte e após a graduação.



## Arquitetura da aplicação proposta

O modelo arquitetural proposto para a solução AViS é híbrido. São associados o modelo cliente-servidor, tipicamente utilizado por *webapps*, e um modelo P2P distribuído, como em aplicativos *torrent*.

Seguindo o modelo cliente-servidor, é possível utilizar a *API Alloy*, que já está em produção e oferece uma parte essencial ao produto cuja viabilidade pretende-se demonstrar com este MVP. Além disso, um servidor remoto, imparcial face ao cliente AViS, vai arbitrar o diálogo de dados entre clientes.

Já com o que um modelo P2P oferece, o sistema AViS será capaz de transmitir dados sensíveis ao tempo com mais agilidade. *Buffers* de áudio e vídeo serão transmitidos, via protocolo UDP, seguindo o caminho mais curto, determinado pela infraestrutura de redes, entre um cliente e outro.

Na figura 1, observa-se uma representação global da comunicação entre diferentes instâncias do cliente AViS. A figura apresenta também a infraestrutura disponibilizada pela Alloy City Linguistics. O servidor assume um papel de árbitro entre os clientes conectados, mantendo-os atualizados quanto a dados relevantes para a experiência do usuário. Os dados mais importantes a serem considerados aqui são os IPs e portas de cada instância cliente conectada. Uma lista de endereços essencial para a comunicação direta (P2P) entre instâncias clientes.

Figura – Visão global da comunicação intra instâncias

A figura 2 apresenta uma visão global da arquitetura empregada no aplicativo cliente. Unreal Engine 4, tecnologia central no projeto, é responsável tanto pela renderização do ambiente virtual quanto pela sincronização de endereços. O módulo VoIP é responsável pela captura e envio de áudio entre clientes. O módulo FMoA é responsável pela captura da expressão facial do usuário.



Figura 2 - Arquitetura do cliente AViS

## Tecnologias utilizadas e APIs

Seguem as tecnologias chave utilizadas no projeto, assim como as principais razões por trás da decisão de utilizá-las.

### Tecnologias

* C++

Uma linguagem de programação com 34 anos de amadurecimento, C++ é o padrão de algumas das indústrias mais exigentes em matéria de *software*. Ela foi escolhida para este projeto pelas razões listadas abaixo:

* Oferece acesso de baixo nível aos recursos de sistema, particularmente à memória;
* Oferece abstrações de nível mais alto, como classes e mecanismos de iteração;
* Alta performance em tempo de execução (na mesma ordem de C e Rust);
* Desenvolvimento ativo (última *release* estável em dezembro de 2017, próxima prevista para 2020);
* Escolha padrão da maior indústria de entretenimento do mundo
* Rico legado.

Unreal Engine 4

Originalmente um mecanismo de jogo (*game engine*), Unreal Engine 4 é um motor de *renderização* 3D em tempo real. Ele costuma ser manipulado via C++ e via UE Blueprints (um formato proprietário de programação visual). Essa ferramenta será responsável pela *renderização* tridimensional do ambiente de interação entre os usuários, abstraindo do projeto as complexidades matemáticas e físicas inerentes a simulações 3D.

Além disso, Unreal Engine contém um módulo responsável pela comunicação via UDP entre usuários, tecnologia essencial ao projeto. Aprender a usar essa parte do mecanismo demanda consideravelmente menos tempo e esforço do que desenvolver a funcionalidade integralmente.

Unreal Engine 4 se apoia em 21 anos de amadurecimento e é usada hoje por desenvolvedores de jogos, artistas 3D, estúdios de arquitetura, estúdios de efeitos especiais, pela indústria automobilística, por estudantes de C++, entre outros. A tecnologia é desenvolvida por Epic Games, sob uma licença de código fonte acessível e de uso educacional livre.

OpenCV

OpenCV (Open Source Computer Vision) é uma biblioteca de *computer vision*. Inicialmente, foi desenvolvida pela Intel, mas hoje é mantida por uma ampla comunidade de programadores independentes, empresas e universidades, sob a licença aberta BSD. O desenvolvimento está ativo, com o último lançamento estável em julho de 2019.

No projeto AViS, OpenCV será usada, sobretudo, para definir as coordenadas do rosto do usuário em cada quadro do fluxo de vídeo.

Blender

Programa de modelagem 3D de código fonte aberto, Blender pode ser usado para a criação de modelos estáticos 3D (*meshes*). Especificamente neste projeto, Blender é utilizado para a criação do avatar utilizado pelo programa para representar os usuários no ambiente virtual.

### APIs Utilizadas

A API do mecanismo de renderização é a interface entre o *software* do projeto e UE4[[1]](#footnote-1). Esta API é exaustivamente documentada no domínio <https://docs.unrealengine.com> e é acessível via C++, UE4 Blueprints ou Python. Neste projeto, o acesso será feito, majoritariamente, via C++.

Acesso à API Alloy (<https://alloy.city>), usada no aplicativo web <https://pantoufle.online>, será demonstrado, mas o uso da API está fora do escopo deste projeto, visto que só será necessário futuramente, caso seja demonstrada a viabilidade tecnológica da ideia aqui explorada.

Ambas as APIs são de acesso local. A API UE4 é acessível por meio da inclusão de arquivos de interface nos programas desenvolvidos. A API Alloy é acessível por meio de chamadas HTTP locais, realizadas exclusivamente pela instância servidor.

## Componentes do SW



## Diagrama de Classes

## Considerações sobre o Banco de Dados Utilizado

O sistema utilizado pelo aplicativo <https://pantoufle.online> atualmente se apoia em um banco de dados NoSQL MongoDB, conectado à API via Mongoose JS. Futuramente, seria necessário acessar esse banco de dados através da API exposta. Por isso, demonstraremos a habilidade de realizar essa conexão, mas julgamos desnecessário, tendo em vista os objetivos do MVP, mapear ou modelar o banco de dados, tal qual ele existe em produção hoje. Primeiramente, porque o contato com o banco é intermediado pela API. Basta conhecer a interface da API e o acesso é feito sem maiores complicações. Segundamente, o MVP vai demonstrar capacidades técnicas efêmeras, que utilizam a memória principal, apenas. Não será necessário, no escopo do MVP, persistir dados gerados ou coletados durante a execução do programa, tanto na instância em servidor quanto nas instâncias clientes.

## Diagrama de Sequência

## Diagrama Pacotes

## Diagrama Estado

## Interfaces com o usuário

A interface gráfica do MVP consiste no ambiente 3D propriamente dito, já que as funcionalidades que se pretende demonstrar podem ser acessíveis via linha de comando, com atalhos de teclado provisórios, ou mesmo automatizadas e otimizadas para o ambiente de demonstração. Juntamente com esse ambiente simulado, o MVP conta com uma janela para inspecionar o trabalho do algoritmo de identificação de rostos, conforme figura 3. Esta janela de inspeção pode ser instanciada com a tecla B ou com o comando *StartVideoCaptureDebugger*.

Figura 3 - Debugger do algoritmo de identificação de rostos

Após a implementação do MVP, na segunda fase do desenvolvimento, será necessário implementar interfaces que cubram o mesmo conjunto de funcionalidades oferecido pelo aplicativo Web atualmente em produção. Seguem alguns exemplos:

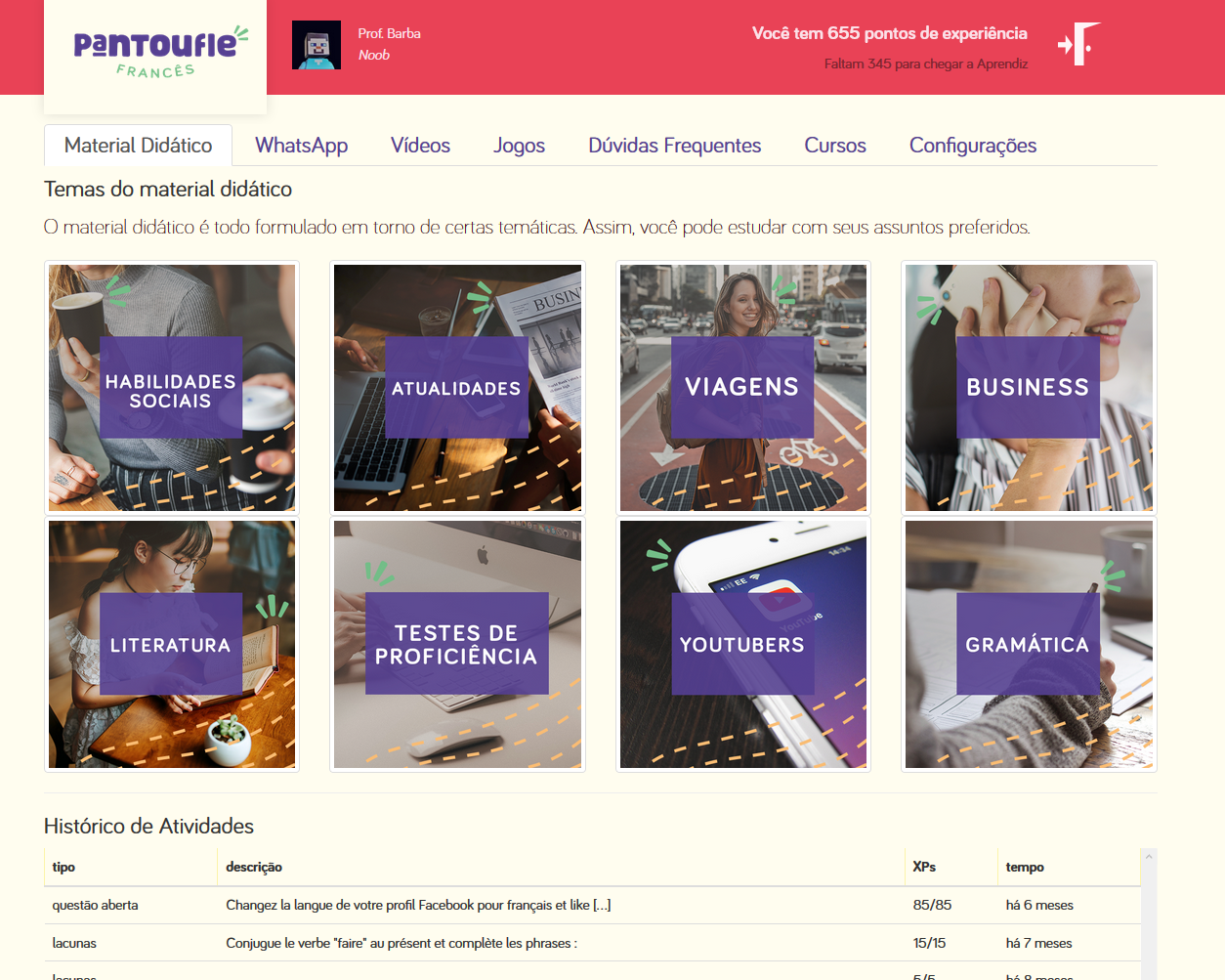


Figura 4 - Navegador de conteúdo



Figura 5 - Lista de perguntas frequentes



Figura 6 - Navegador de cursos disponíveis

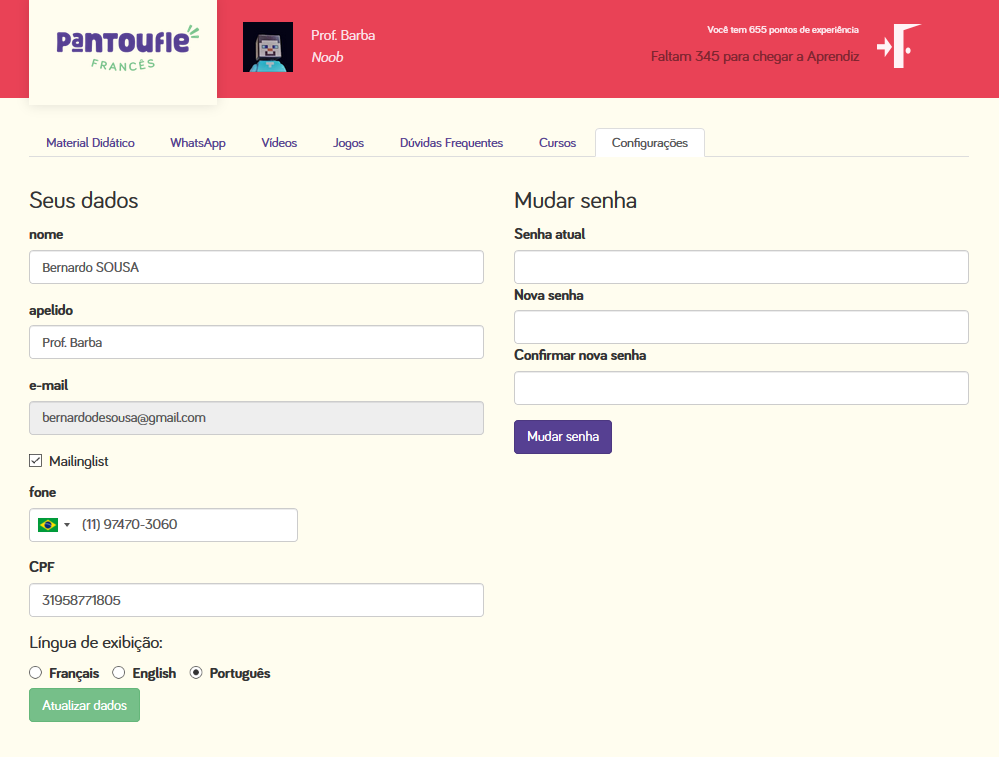


Figura 7 - Formulário de configurações

# Implementação

O código fonte do projeto está integralmente disponível no GitHub, assim como *releases* pré-compiladas, no seguinte endereço: <https://github.com/alloy-city/AViS>

Para implementar o protótipo de demonstração, ou MVP, foi necessário estudar as partes relevantes da vasta documentação de Unreal Engine 4. Essa documentação cobre quase toda a API exposta pelo mecanismo de jogo. Eventualmente, nos deparamos com métodos e atributos, as vezes até mesmo classes inteiras, que não são mencionadas. Nesses casos raros, é sempre possível consultar o código, diretamente, que, embora não seja totalmente aberto, é acessível para consulta. Mesmo quando a documentação cobre o recurso que estamos utilizando, a maneira mais efetiva se assegurar que compreendemos como UE4 funciona é através do código fonte. O programa AViS é, portanto, desenvolvido com ao lado do código do mecanismo UE4, em uma única solução no Visual Studio.

A modelagem do ambiente virtual, que representa uma sala de aula, foi feita no próprio Editor UE4, utilizando modelos geométricos simples, disponíveis em qualquer instalação recente do mecanismo. Para essa tarefa da implementação, foi necessário estudar como UE4 simula geometria 3D, iluminação, texturas e materiais.

Em seguida, implementou-se um avatar, controlável pelo usuário, com base na classe *Pawn* exposta pelo mecanismo de jogo. Essa classe conta com um ponto de vista na sala virtual, que pode ser manipulado pelos mecanismos de entrada de comandos, como mouse e teclado.

Para que a esfera, que representa a cabeça do avatar, seja devidamente capaz de comportar o mapa de bits correspondente a um quadro de rosto, recebido pela rede, foi necessário realizar um procedimento chamado *UV unwrapping*. Este procedimento associa coordenadas de renderização de texturas a um modelo 3D. Neste caso em particular, o modelo 3D é uma esfera, mas as texturas são mapas de bits quadrados, de 64 por 64 pontos. Foi preciso portanto definir as coordenadas das texturas na esfera de forma que o rosto sofra a menor deformação geométrica possível. Esta tarefa foi realizada no software de modelagem 3D Blender.

Para ter acesso à webcam, integramos ao projeto o software de visão de máquina OpenCV. O primeiro desafio dessa tarefa foi aprender a utilizar as funcionalidades mais elementares do OpenCV, isoladamente. No nosso caso, escolhemos capturar o vídeo da webcam e apresentá-lo, sem modificação alguma, de volta ao usuário, em uma janela. Em seguida, ao tentar integrar o OpenCV a um projeto baseado em UE4, descobrimos algumas colisões entre as duas ferramentas. Uma função OpenCV tem o mesmo nome de uma classe da biblioteca de funções matemáticas chamada Kismet Math. Em iterações futuras do projeto, conflitos como este poderão ser evitados através do mecanismo de subsistemas de UE4. Este mecanismo é usado internamente para estruturar os diferentes componentes UE4. É possível utilizar essa mesma estrutura para integrar OpenCV ao projeto, criando para ele um espaço de memória privado, diferente do espaço global. Essa solução, entretanto, requer a reestruturação de uma parte substancial da implementação. Optou-se por renomear a função no código fonte do OpenCV e utilizar essa versão alterada do programa. É preciso manter em vista essa modificação e implementar uma solução definitiva antes de tentar atualizar OpenCV para uma versão mais recente.

Uma vez que OpenCV e UE4 estavam trabalhando juntos, em um mesmo projeto, implementou-se, um a um, os métodos responsáveis por tratar o fluxo de vídeo no cliente que o captura. A saber:

1. Captura-se um quadro
2. Identifica-se as coordenadas e o raio de um rosto no Quadro
3. Recorta-se o Quadro utilizando as coordenadas encontradas
4. Redimenciona-se o rosto para 64 por 64 pontos
5. Compacta-se o rosto em JPG com índice de qualidade 80
6. Envia-se o rosto compactado ao objeto responsável por comunicação em rede.

Este processo foi implementado na classe Webcam

A classe StreamService é responsável por enviar os quadros processados pela rede. Essa classe foi projetada para processar 30 quadros por segundo, o que a classifica como ponto de estrangulamento de performance em potencial. Ela deve iniciar um loop de observação, para aguardar conexões de rede. Para que o programa não pare até que uma conexão seja estabelecida, o loop de observação é executado em um thread paralelo ao programa principal.

Os bits correspondentes ao rosto, já compactado, são enviados à máquina remota. Nela, uma função exposta pela API UE4 chamada ImportBufferAsTexture2D. Essa função recebe um buffer correspondente a uma imagem, expande a imagem caso ela esteja compactada, e cria um objeto da classe nativa Texture2D. A cada quadro recebido pela rede, essa função é chamada. Se ela retornar um objeto da classe Texture2D válido, aplica-se a textura ao avatar correspondente ao usuário que gerou o quadro em questão.

Todo esse processo cria a impressão de que um determinado avatar, presente na sala de aula virtual, tem o rosto do usuário que o controla.

Finalmente, para implementar a funcionalidade de transmissão de voz, foi necessário aprender a utilizar os recursos de processamento de áudio do mecanismo de jogo. O primeiro passo é ganhar acesso ao microfone do usuário através da ativação do plugin Voice, desenvolvido também pela Epic Games. Ao assegurar que os usuários estão em uma mesma sessão (USession), UE4 se encarrega de enviar os pacotes de áudio compactados, para todos os usuários conectados à mesma sessão. Entretanto, para aumentar a sensação de presença, alteramos o comportamento do mecanismo para que a voz, no usuário remoto, seja reproduzida a partir de um ponto específico na sala de aula virtual, e não como um som ambiente independente da simulação. Para isso, utilizamos a classe nativa USoundCue, que pode ser posicionada numa cena virtual e emitir sons. Os sons emitidos por um objeto dessa classe contam com diversos recursos de processamento de som em tempo de execução, entre eles atenuação diferenciada entre os dois canais. Notemos que UE4 reproduz os sons do programa em execução em dois canais. Esquerdo e direito. Ao atenuar por exemplo o lado direito mais do que o esquerdo ao reproduzir um determinado som, o usuário tem a sensação de que o som está vindo do seu lado esquerdo.

Para usufruir dos recursos de atenuação diferenciada oferecidos pela classe USoundCue, instancia-se um objeto dessa classe juntamente com o avatar, mantendo os atributos de coordenadas espaciais do objeto sincronizados com as coordenadas da cabeça do avatar. Durante a instanciação do avatar e do seu componente de som, definimos uma relação de parentalidade espacial entre os dois objetos. Dessa forma, os cálculos dessa sincronização são delegados ao mecanismo de jogo.

Alcançou-se todos esses passos de implementação na versão 0.5 do cliente AViS. Tanto o código fonte quanto um arquivo compactado contendo todos os arquivos necessários para a demonstração da tecnologia estão disponíveis no repositório do projeto.

# Projeto de Teste

# Instalação do Software

O MVP exige que os binários do OpenCV estejam disponíveis no Path do Windows. Além disso, o arquivo cascada de aprendizado de máquina para identificação da posição do rosto no quadro precisa estar em um caminho específico. Portanto, os passos seguintes devem ser seguidos precisamente, em um ambiente Windows 10, para que a demonstração ocorra normalmente.

1. Baixar o arquivo zip da release 0.5
2. Descompactar o arquivo zip
3. Criar a estrutura de pastas C:/AViS/Plugins/OpenCV/Resources/Data/haarcascades/
4. Copiar o arquivo haarcascade\_frontalface\_default.xml para a pasta criada no passo 3.
5. Incluir as bibliotecas OpenCV opencv\_videoio\_ffmpeg411.dll e opencv\_videoio\_ffmpeg411\_64.dll ao Path do Windows
6. Lançar a primeira instância do executável AViS.exe emu ma das máquinas disponíveis para a demonstração
7. Precionar a Tecla H para começar o serviço local de demonstração
8. Lançar a segunda instância do executável, em uma segunda máquina
9. Em ambas as máquinas, precioar F para iniciar o processo de captura de rosto
10. E mambas as máquinas, precioar V para iniciar o processo de captura de voz.

# Análise dos Resultados

# Conclusão

Referências

Glossário

**Student**: usuário com nível de acesso 0;

**Teacher**: usuário com nível de acesso 1;

**Creator**: usuário com nível de acesso 2;

**Coordinator**: usuário com nível de acesso 3;

**Admin**: usuário com nível de acesso 4;

**Resource**: menor unidade do material didático;

**Lesson**: conjunto de Resources;

**Chapter**: conjunto de Lessons;

**Meeting**: par de momentos no tempo, definidos em UTC, que representam uma aula ao vivo;

**Course**: conjunto de Meetings

**Pack**: conjunto de Products (exceto outros Packs)

**Product**: unidade comercializável (Meetings, Courses, Lessons, Chapters and Packs);

**CR**: Cliente AViS remoto

**CL**: Cliente AViS local

**Webapp**: aplicativo projetado para funcionar em um ambiente provido por um navegador web.

**P2P**: *Pear to Pear*; estratégia de comunicação em rede que envolve duas instâncias remotas equivalentes, sem a intermediação de um servidor

**Torrent**: protocolo de transmissão de dados *P2P*.

**API Alloy**: Interface de acesso à base dados da empresa Alloy City Linguistics. Esta API foi desenvolvida por um dos integrantes do presente trabalho e está na base do principal aplicativo mantido pela empresa, o webapp Pantoufle (https://pantoufle.online).

Apêndice

1. [↑](#footnote-ref-1)