Pedro Bernardo de Sousa	0030481711006
Vítor Andrade Marques da Silva	0030481511040
Weuller Júnior Souza Bessa	0030481621040

AViS – Alloy Virtual Space

Documentação de Desenvolvimento de Software

Laboratório de Engenharia de Software – 2º semestre/2019

Índice

1. Documentação de Visão do Problema

Um Ambiente Virtual de Aprendizagem, na área de educação, é uma plataforma, geralmente baseada em tecnologias web, que oferece suporte ao processo educativo (WELLER, 2007). Geralmente, plataformas AVA são desenvolvidas por instituições de ensino que veem benefícios em disponibilizar material didático em formato digital para os alunos, além de oferecer uma via suplementar de contato entre tutores e estudantes.

Ambientes virtuais de aprendizagem podem ser divididos em 4 categorias, formadas pela associação combinatória de duas grandezas: tempo e espaço. Quanto ao tempo, os participantes podem ou não interagir sincronamente, isto é, em um mesmo momento. Quanto ao espaço, os participantes podem ou não interagir em um mesmo lugar (LAZZAROTTO et al., 2011).

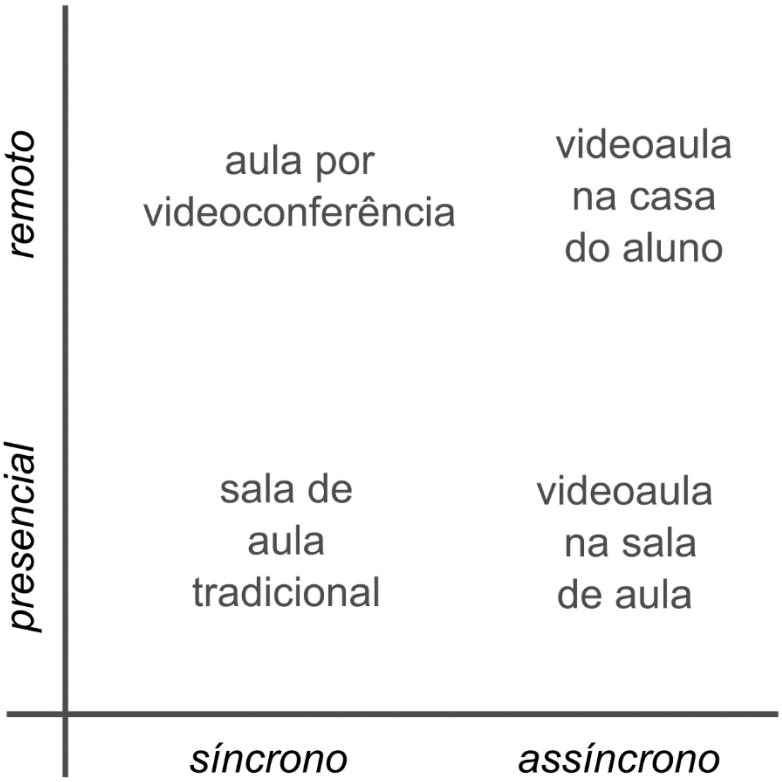
Quando os participantes interagem em um mesmo momento, diz-se que a aula é síncrona. Uma aula gravada, por outro lado, é chamada assíncrona, pois a interação entre o professor e o aluno acontece em momentos distintos: o professor grava o conteúdo em um momento e o aluno o consome em outro.

Quando os participantes interagem em um mesmo lugar, por exemplo em uma sala de aula tradicional, o aluno está em presença do professor. Isso classifica a aula como presencial. Oposto a isso, é possível ministrar aulas a distância, com o uso de tecnologias de comunicação, como cartas, telefone ou Internet.

Durante muitos séculos, só era possível aulas síncronas presenciais. Remotamente, não se podia gerar conteúdo educacional e esperar que alunos usufruíssem dele imediatamente. No estado tecnológico a que chegou nossa civilização, podemos observar as quatro combinações possíveis. Em salas de aulas tradicionais, temos interação direta entre tutores e alunos, o que se categoriza como educação presencial síncrona. Nessas mesmas salas de aula, o instrutor pode apresentar conteúdo gravado previamente por especialistas, o que se classifica como educação presencial assíncrona. O processo educacional pode se dar na residência do aluno, com conteúdo gravado previamente, o que se classifica como educação remota assíncrona. E, finalmente, alunos e professores podem se encontrar *online*, em plataformas de videoconferência, em horários acordados previamente, para gozar de um processo educacional remoto e assíncrono. Na Figura 1, observa-se em um plano cartesiano os

dois eixos descritos acima, acomodando exemplos dessas quatro modalidades de educação.

Figura 1 - Modalidades de educação



Fonte: elaborado pelos autores

Segundo os dados coletados ao longo de cinco anos de experiência em educação a distância da escola Pantoufle, demonstra-se que a associação das duas modalidades remotas de educação tem um enorme potencial de baixar custos sem sacrificar qualidade no processo educacional. É desta convicção que emana o apoio que a escola, assim como a Alloy City Linguistics, consagra ao presente projeto.

Contabilizar os custos da educação não é uma tarefa trivial. É preciso considerar gastos com transporte das pessoas que devem se deslocar até a escola, custos iniciais de investimento em infraestrutura, custos de manutenção das instalações da escola, entre muitos outros. Alguns desses custos são indiretos, como o custo do risco acrescido

pelos deslocamentos frequentes. Riscos de acidentes, por exemplo. É verdade que tais custos indiretos parecem extrapolações desnecessárias nessa análise. No entanto são custos reais, pagos pela sociedade como um todo, em uma época de otimizações de grande escala e esforços de transição tecnológica, a caminho de uma civilização verdadeiramente global.

Ao desenvolver sistemas que ofereçam a melhor experiência educacional possível a distância, se servindo para tanto de desdobramentos tecnológicos recentes, contribui-se, mesmo que modicamente, com esses esforços globais (GUTERRES, 2015).

Atualmente, atividades educativas remotas síncronas são realizadas com sistemas de videoconferência que se concentram, sobretudo, na transmissão de vídeo e áudio via protocolo TCP/IP. Via de regra, tais projetos ignoram considerações suplementares a respeito da importância da sensação de presença oferecida por um ambiente de aula físico, talvez por ignorar a importância desse aspecto, ou talvez para evitar os prováveis embaraços técnicos da empreitada. Esses benefícios são, entretanto, flagrantes demais para se evitar indefinidamente, sobretudo na área de aquisição de língua estrangeira para adultos, especialidade da escola Pantoufle.

Este trabalho lança, portanto, a empreitada de estudar tais questões e oferecer elementos de resposta, por meio de Tecnologias da Informação e Engenharia de Software.

2. Requisitos do Sistema

Segue uma descrição da convergência de circunstâncias que levou à definição dos requisitos, funcionais e não funcionais, do presente projeto.

2.1. Descrição da técnica utilizada para levantamento dos requisitos

Ao longo dos últimos 18 meses, desde o lançamento da terceira iteração da plataforma Pantoufle, professores e alunos da escola vêm compartilhando suas impressões sobre o sistema. É com base nesse retorno que a necessidade de mercado descrita na seção 3.1.1 – Problemas Encontrados – foi identificada. A seleção das funcionalidades do MVP, por sua vez, foi feita a partir da distinção entre casos de uso triviais, que o sistema já faz ou poderia fazer em um ou dois ciclos de desenvolvimento, e casos de uso atípicos, que talvez tenham o potencial de revolucionar o ensino síncrono remoto.

Por um lado, a plataforma Pantoufle, atualmente em desenvolvimento contínuo, com base em ciclos curtos de desenvolvimento, coleta continuamente informações sobre as necessidades da escola. Por outro lado, numa postura exploratória de pesquisa e desenvolvimento de software, este trabalho se apoiou no contexto e *savoir faire* da escola Pantoufle para definir os requisitos que, uma vez implementados, trazem elementos de resposta à problemática delimitada.

Engenharia de requisitos de qualidade é alcançada quando se conhece bem as necessidades do usuário (PRESSMAN, 2014). Nas circunstâncias deste trabalho, esse conhecimento existe por conta do envolvimento de um dos autores com a escola Pantoufle desde sua fundação, em setembro de 2014.

2.2. Situação Proposta

Numa videoconferência típica, transmite-se os fluxos de dados gerados por um microfone e por uma câmera. Uma abordagem simples, que provê uma experiência comparável a observar o interlocutor por um aparelho de TV. O projeto AViS pretende demonstrar a viabilidade de criar uma experiência mais rica e mais imersiva, em que um espaço tridimensional virtual simulado seja compartilhado entre os usuários. Espera-se que o sistema seja capaz de:

- simular um espaço virtual em três dimensões;
- gerar avatares que representem os usuários conectados;
- criar a sensação de que a voz de um determinado usuário remoto está emanando do ponto no espaço virtual onde se encontra a cabeça de seu avatar; e
- apresentar o rosto do usuário remoto no rosto de seu avatar.

Como veremos em maiores detalhes na seção 5.2.1 – Tecnologias, diversas tecnologias distintas foram associadas para implementar as funcionalidades acima. Dentre elas, as mais notáveis são Unreal Engine 4, mecanismo de jogo responsável por gerar e manter o ambiente 3D em tempo de execução, e OpenCV (*Open Computer Vision*), responsável por identificar o rosto do usuário via aprendizado de máquina.

Os demais processos envolvidos no funcionamento do MVP são descritos detalhadamente na seção 6 – Implementação.

2.3. 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 se olhe 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.

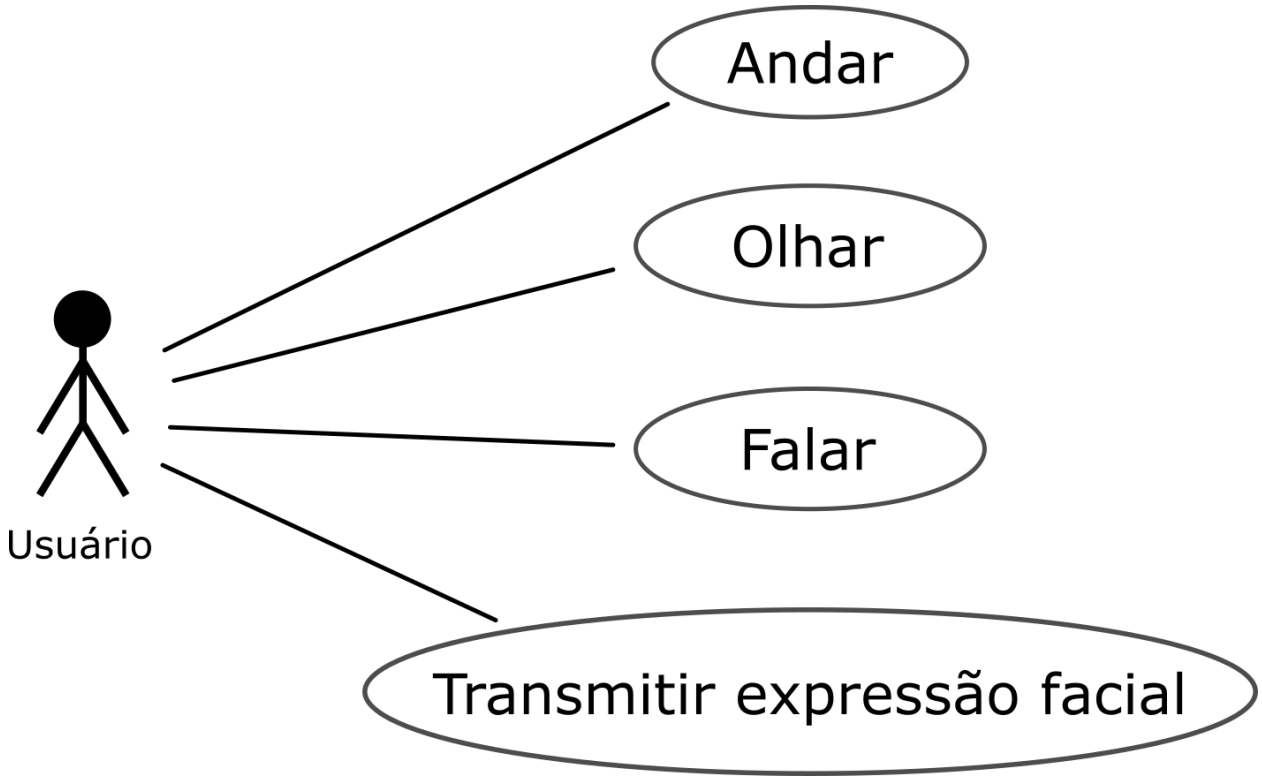
RF4 – Transmitir expressão facial

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.

2.3.1. Casos de Uso

A Figura 4 apresenta os Casos de Uso do MVP.

Figura 4 – Diagrama de Casos de Uso



Fonte: elaborado pelos autores

Quadro 3 – Caso de Uso FR1

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

Fonte: elaborado pelos autores

Quadro 4 – Caso de Uso FR2

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

Fonte: elaborado pelos autores

Quadro 5 – Caso de Uso FR3

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

Fonte: elaborado pelos autores

Quadro 6 – Caso de Uso FR4

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

Fonte: elaborado pelos autores

2.4. Requisitos Não Funcionais

Quatro requisitos não funcionais foram identificados e alcançados. Eles dizem respeito ao tempo de resposta entre clientes, ao uso de memória principal, ao uso de memória secundária pelos executáveis e ao uso de recursos de processamento na instância em servidor.

2.4.1. Tempo de resposta

Para oferecer as funcionalidades desejadas, é importante que a comunicação entre instâncias cliente seja a menor possível. Concluiu-se que 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.

2.4.2. 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*).

2.4.3. 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.

2.4.4. 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.

3. Análise e Design

A partir de uma análise preliminar, decidiu-se dividir o projeto em duas grandes fases. Durante a primeira fase, ocorrem os esforços de pesquisa e desenvolvimento de um protótipo funcional que demonstre a viabilidade da proposta: um Produto Mínimo Viável, ou MVP. 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 selecionadas 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.

Futuramente, em uma segunda fase, que está fora do escopo acadêmico do projeto, será desenvolvido um produto pronto para o mercado, onde será necessário, entre outros, um sistema de autenticação, navegação de cursos disponíveis, uma agenda, um sistema de updates automáticos, um processo bem definido de desenvolvimento contínuo, testes automáticos e compatibilidade com os principais sistemas operacionais. Tais funcionalidades se somam em um projeto de grande envergadura, estão fora do escopo da primeira fase e, portanto, do presente projeto.

O MVP desenvolvido neste trabalho demonstra a ideia central e prepara o caminho para que o projeto continue, no semestre seguinte e/ou após a graduação. Os detalhes do projeto são expostos nos subtópicos a seguir.

3.1. 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*.

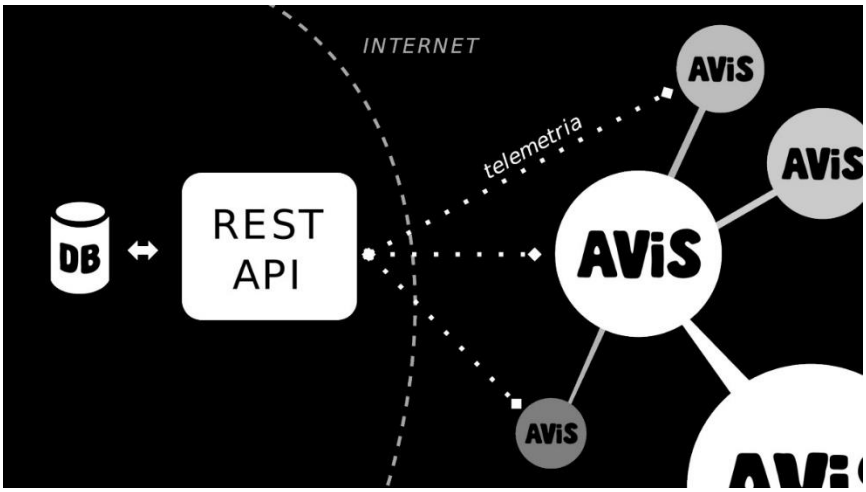
Seguindo o modelo cliente-servidor, é possível utilizar a *API Alloy*, que já está em produção e oferece uma parte essencial ao produto cuja viabilidade pretende-se demonstrar com este MVP.

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 5 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.

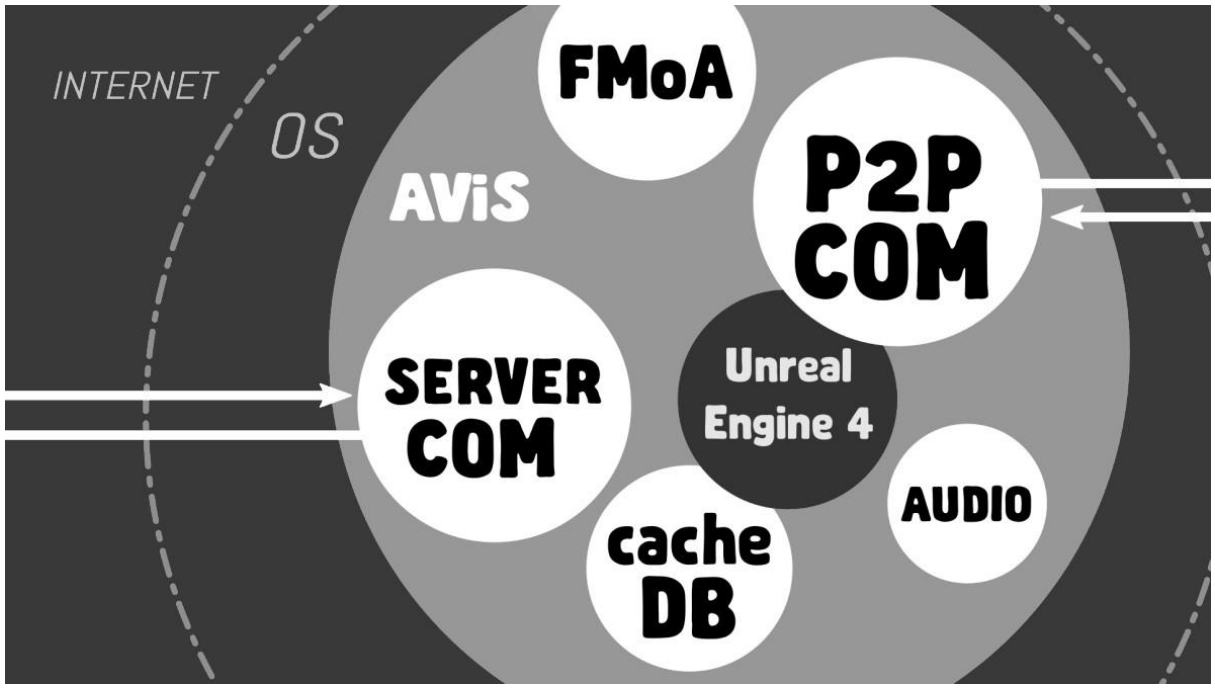
Figura 5 – Visão global da comunicação intra instâncias



Fonte: Elaborado pelos autores

A figura 6 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 6 - Arquitetura do cliente AViS



Fonte: Elaborado pelos autores

3.2. Tecnologias utilizadas e APIs

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

3.2.1. Tecnologias

C++

Uma linguagem de programação com 34 anos de amadurecimento, C++[1] 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 (último lançamento estável em dezembro de 2017, próximo previsto 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[2] 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

[1] Disponível em <https://isocpp.org>. Acesso em: 22 nov 2019.
[2] Disponível em <https://unrealengine.com>. Acesso em: 22 nov 2019.

mecanismo demanda consideravelmente menos tempo e esforço do que desenvolver a funcionalidade integralmente.

Segundo a empresa desenvolvedora, 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[3], também chamado de Open Source Computer Vision, é uma biblioteca de visão computacional. Inicialmente, foi desenvolvida pela Intel (KAEHLER, 2017), 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[4] 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.

3.2.2. APIs Utilizadas

A API do mecanismo de renderização é a interface entre o *software* do projeto e UE4. Esta API é exaustivamente documentada[5] e é acessível via C++, UE4 Blueprints ou Python. Neste projeto, o acesso será feito, majoritariamente, via C++.

[3] Disponível em <https://opencv.org>. Acesso em: 22 nov 2019.
[4] Disponível em <https://blender.org>. Acesso em: 22 nov 2019.
[5] Disponível em <https://docs.unrealengine.com>. Acesso em: 22 nov 2019.

Acesso à API Alloy[6], usada na plataforma Pantoufle[7], 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.

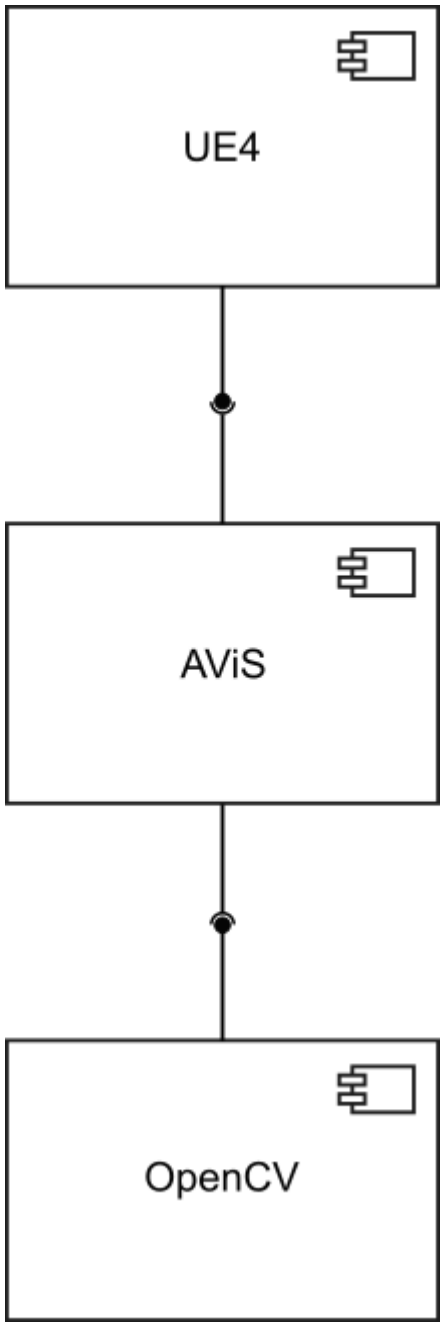
Ambas as APIs são de acesso local. A API UE4 é acessível por meio da inclusão de arquivos de interface nos programas desenvolvidos. A API Alloy é acessível por meio de chamadas HTTP locais, realizadas exclusivamente pela instância servidor.

[6] Disponível em <https://alloy.city>. Acesso em: 22 nov 2019.
[7] Disponível em <https://pantoufle.online>. Acesso em: 22 nov 2019.

3.3. Componentes do SW

O software cliente AViS é dependente de Unreal Engine 4 e OpenCV, que expõem interfaces de acesso local, via linkedição.

Figura 7 - Diagrama de Componentes

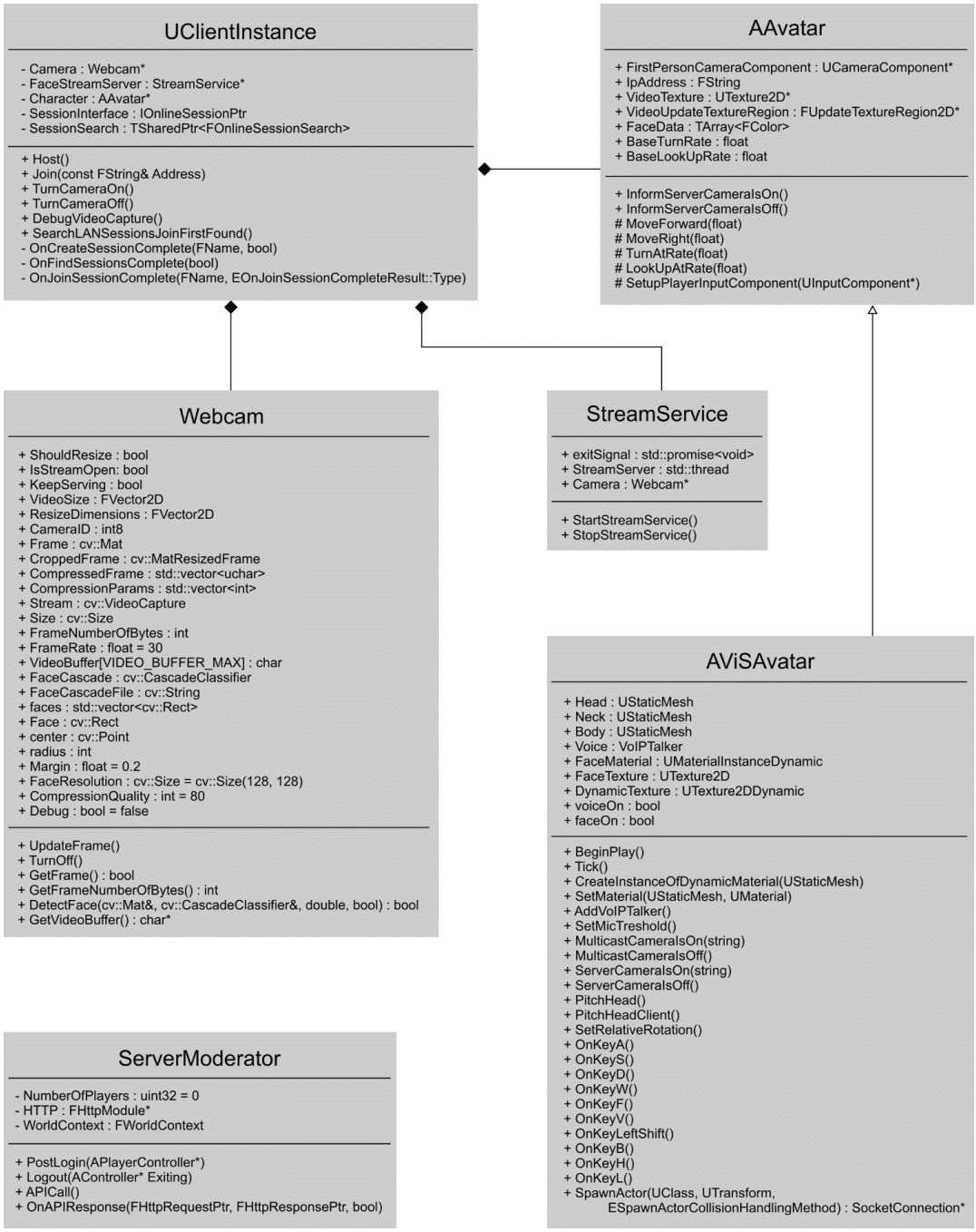


Fonte: elaborado pelos autores

3.4. Diagrama de Classes

As classes prefixadas com a letra A são descendentes de classes implementadas por UE4, omitidas por simplicidade. A classe ServerModerator é instanciada apenas no servidor, e é independente das demais.

Figura 8 – Diagrama de Classes



Fonte: elaborado pelos autores

3.5. Considerações sobre o Banco de Dados Utilizado

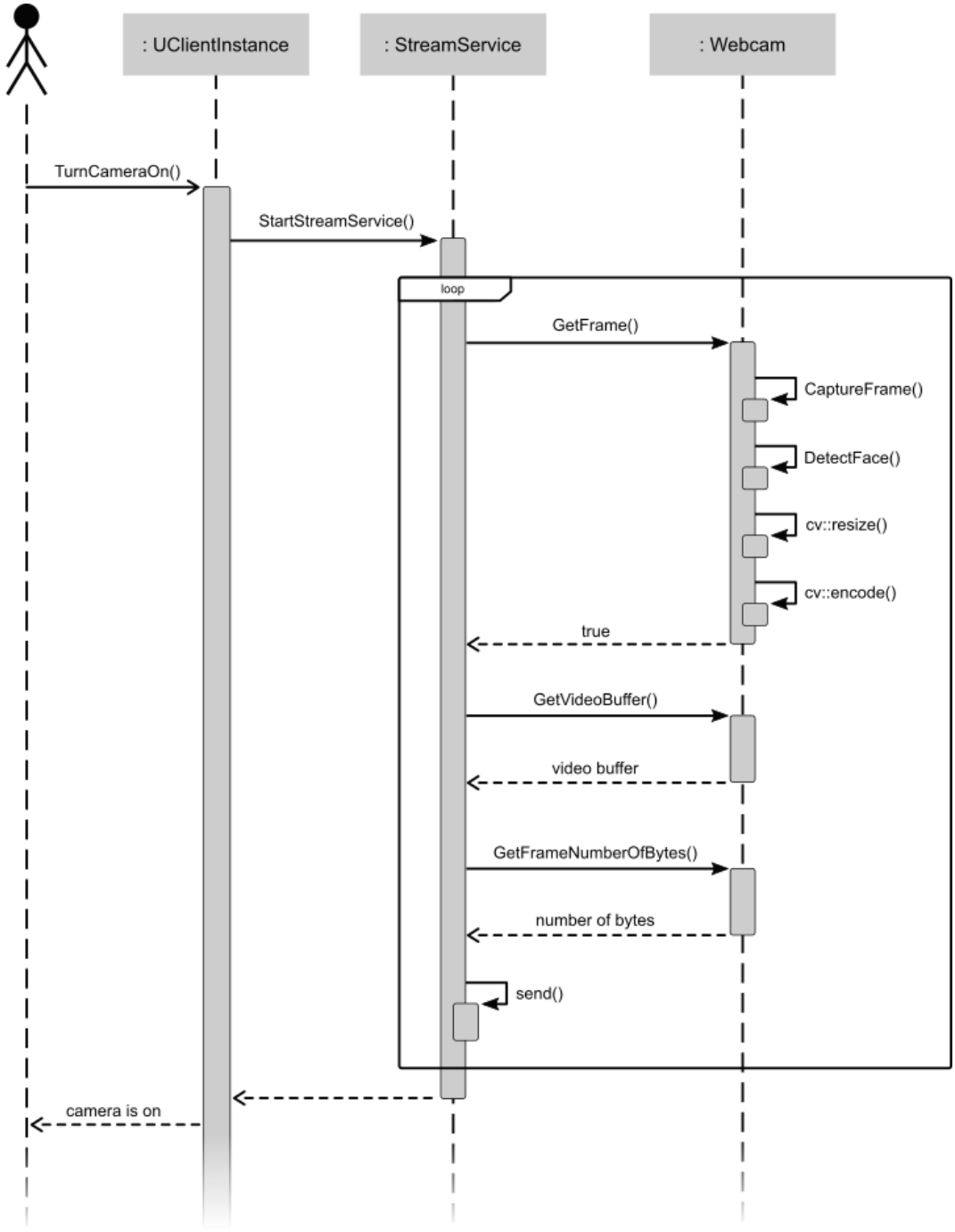
O sistema utilizado pala plataforma Pantoufle[8] atualmente se apoia em um banco de dados NoSQL MongoDB, conectado à API Alloy via Mongoose JS[9]. Futuramente, seria necessário acessar esse banco de dados pela referida API. Por isso, o MVP demonstra a habilidade de realizar essa conexão. Contudo, foi considerado desnecessário, tendo em vista o escopo do projeto, 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, conforme demonstra o MVP. Além disso, 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.

[8] Disponível em <https://pantoufle.online>. Acesso em: 22 nov 2019.
[9] Disponível em <https://mongoosejs.com>. Acesso em: 22 nov 2019.

3.6. Diagramas de Sequência

Na Figura 9, modela-se a ativação da câmera do usuário, assim como o processamento dos quadros que ela produz.

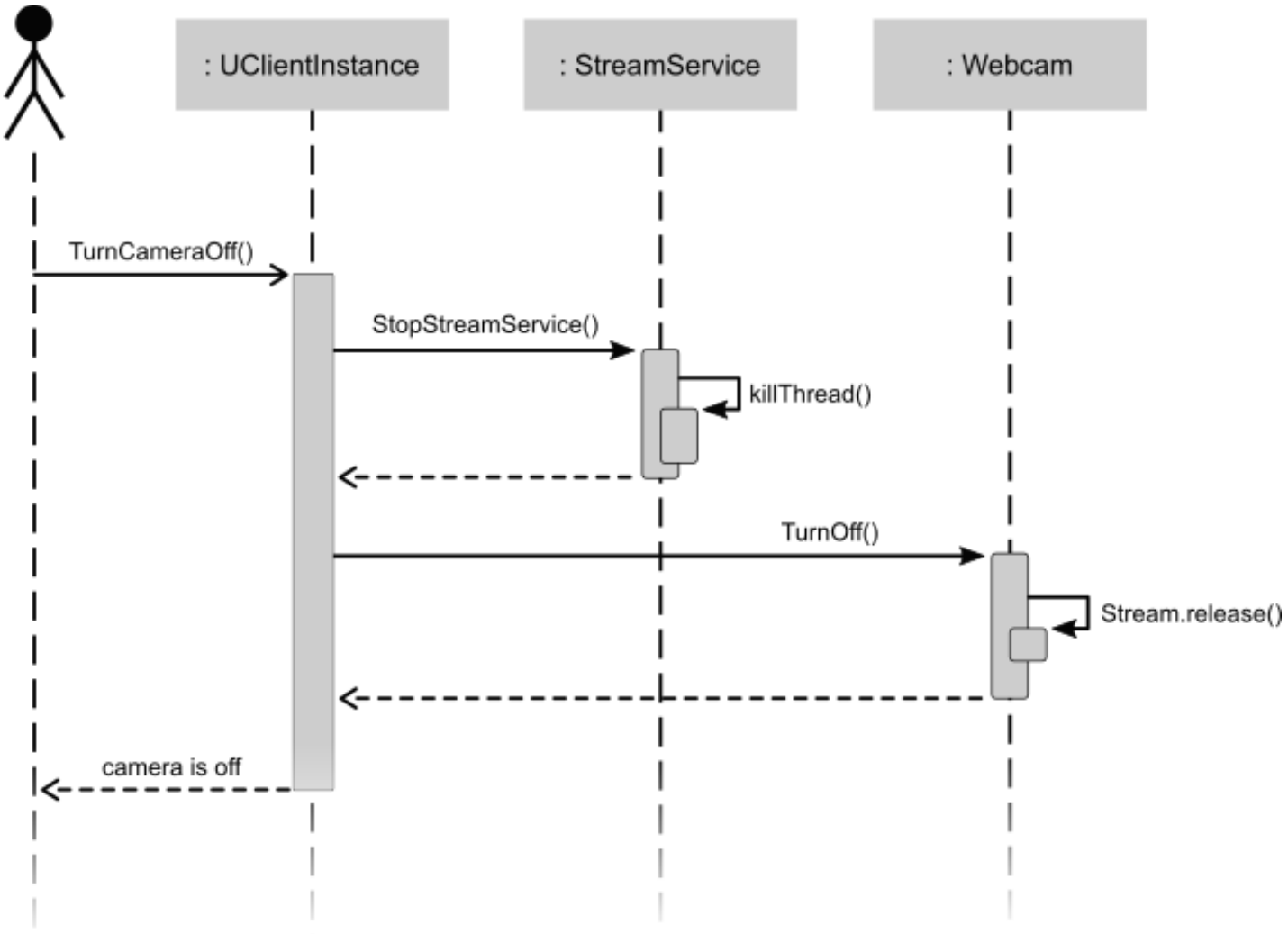
Figura 9 - Diagrama de Sequência: TurnCameraOn



Fonte: elaborado pelos autores

Na Figura 10, modela-se o desligamento da câmera.

Figura 10 – Diagrama de Sequência: TurnCameraOff

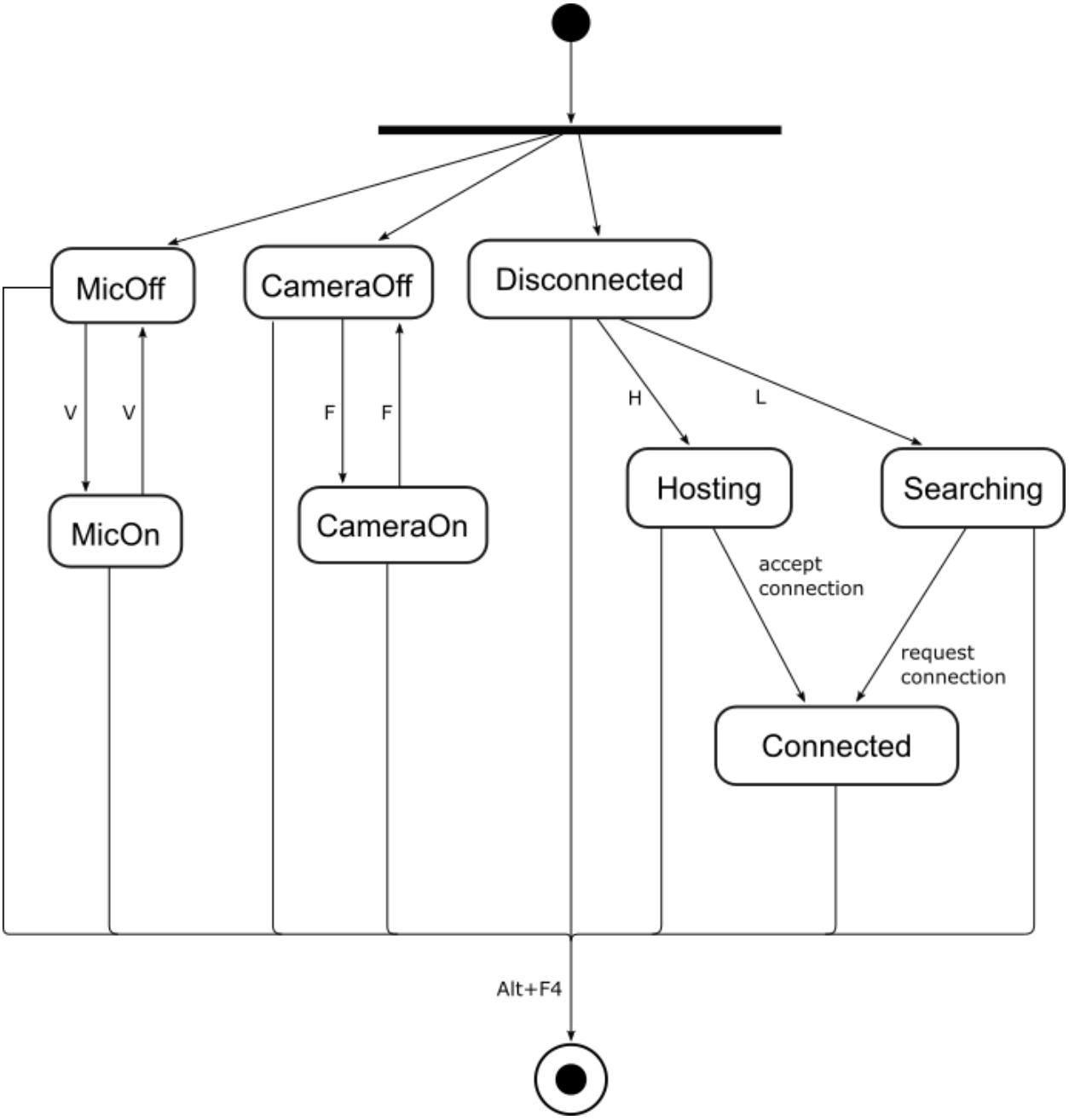


Fonte: elaborado pelos autores

3.7. Diagrama Estados

Na Figura 11, modelam-se os diferentes estados que o cliente AViS pode assumir. Alguns desses estados podem ser alcançados em paralelo. Por exemplo, a câmera e o microfone podem estar ligados ao mesmo tempo.

Figura 11 – Diagrama de Estados



Fonte: elaborado pelos autores

3.8. 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 12. Esta janela de inspeção pode ser instanciada com a tecla B ou com o comando *StartVideoCaptureDebugger*.

Figura 12 - Debugger do algoritmo de identificação de rostos

Fonte: elaborado pelos autores

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.

4. Implementação

O código fonte do projeto está integralmente disponível no GitHub[10], assim como *releases* pré-compiladas.

4.1. Modelagem 3D

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, foi necessária a utilização de métodos e atributos, as vezes classes inteiras, que não são mencionadas na documentação. 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 utilizado, a maneira mais efetiva se assegurar a compreensão sobre como UE4 funciona é através do código fonte. O programa AViS é, portanto, desenvolvido 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

[10] Disponível em <https://github.com/alloy-city/AViS>. Acesso em: 22 nov 2019.

128 por 128 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.

4.2. Captura e Processamento de Vídeo

Para ter acesso à webcam, integrou-se 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. A saber, como capturar o vídeo da webcam e apresentá-lo, sem modificação alguma, de volta ao usuário, em uma nova janela. Em seguida, ao tentar integrar o OpenCV a um projeto baseado em UE4, descobriu-se algumas colisões entre as duas ferramentas. Uma função OpenCV tem o mesmo nome de uma classe da biblioteca de funções matemáticas Kismet Math, utilizada pelo mecanismo UE4. 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, implementaram-se, um a um, os métodos responsáveis por tratar o fluxo de vídeo no cliente de origem. 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) redimensiona-se o rosto para 128 por 128 pontos;
5) compacta-se o rosto em JPG com índice de qualidade 80; e
6) envia-se o mapa de bits 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 potencial de estrangulamento de performance. 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 são enviados à máquina remota. Uma vez recebidos pela máquina de destino, uma função exposta pela API UE4 chamada ImportBufferAsTexture2D é utilizada para descompactar a imagem JPG. 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.

4.3. Captura e Processamento de Áudio

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, alterou-se 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, utilizou-se 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. O mecanismo 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, define-se 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.

4.4. Manual do Usuário

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 versão 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 em uma das máquinas disponíveis para a demonstração;
7) pressionar a tecla H para começar o serviço local de recepção;
8) lançar a segunda instância do executável, em uma segunda máquina;

9) em ambas as máquinas, pressionar a tecla F para iniciar o processo de captura de rosto; e

10) em ambas as máquinas, pressionar a tecla V para iniciar o processo de captura de voz.

5. Conclusão

Este projeto teve como objetivo estudar o desconforto frequentemente apontado por usuários de ferramentas de educação a distância síncronas, para então propor caminhos possíveis na busca de soluções para este problem. Quando o projeto começou a assumir sua forma final, tornou-se claro que a experiência de usuário proposta tem um enorme potencial. A comunidade de professores e alunos envolvidos com educação a distância receberia uma ferramenta baseada nos conceitos aqui explorados de braços abertos. As seções de teste realizadas até aqui com o MVP despertaram enorme entusiasmo em 5 professores de FLE da Pantoufle, o que valida a direção para a qual este breve projeto aponta.

O principal conceito aqui explorado está sendo estudado também por líderes da indústria, o que valida a ideia. A Oculus Research, incorporada pelo Facebook, anunciou em outubro de 2019 que está trabalhando em telepresença via realidade virtual (SHEIKH, 2019). Ainda em 2016, a Microsoft anunciou novas funcionalidades de comunicação em tempo real com sua proposta de realidade aumentada chamada Hololens (FOWERS, 2016). Infelizmente, essa validação da ideia vem acompanhada da declaração implícita de obsolescência do projeto AViS, o que proíbe os autores de recomendar a continuidade do projeto nos moldes atuais.

Em estudos futuros, entretanto, pode ser válido explorar a possibilidade de contribuir com a pesquisa em andamento na área de comunicação síncrona usando o mesmo conjunto de tecnologias aqui empregado. Mais precisamente, com aprendizado de máquinas e mecanismos de jogos 3D, é possível captar expressões faciais e aplicá-las em modelos 3D no intuito de criar uma forte sensação de presença entre usuários separados por vastas distâncias.

Além desta constatação da validade da proposta, o projeto se revelou suficientemente enriquecedor para os autores. O paradigma da Orientação a Objetos, a linguagem de programação C++, assim como a inteligência artificial estudada e empregada no projeto representam áreas de conhecimento de alta demanda no campo de engenharia de software, onde estudantes de Análise e Desenvolvimento de Sistemas almejam atuar.

Referências

FOWERS, Spencer; CUTLER, Ben; CHANG, Wayne. **Holoportation**. 2016. Disponível em: <https://www.microsoft.com/en-us/research/project/holoportation-3/>. Acesso em: 18 nov. 2019.

GUTERRES, João; SILVEIRA, Milene. **Desafios e Novas Possibilidades de Uso de Learning Management Systems**. Anais do Xxvi Simpósio Brasileiro de Informática na Educação (sbie 2015), [s.l.], p.21-30, 26 out. 2015. Sociedade Brasileira de Computação - SBC. http://dx.doi.org/10.5753/cbie.sbie.2015.21.

KAEHLER, Adrian. **Learning OpenCV 3**: Computer Vision in C++ with the OpenCV Library. Sebastopol: O'reilly Media, 2017. 1024 p. ISBN 978-1491937990.

KEENEY-KENNICUTT, Wendy. **Texas A&M Chemist Experiments with Potential of Online Learning**. 2013. Disponível em: <https://science.tamu.edu/news/2013/09/texas-am-chemist-experiments-with-potential-of-online-learning/>. Acesso em: 22 nov. 2019.

LAZZAROTTO, Lissandra Luvizão et al. **A educação em ambientes virtuais**: proposição de recursos computacionais para aumentar a eficiência do processo ensino-aprendizado. Revista Brasileira de Informática na Educação, [s.l.], v. 19, n. 02, p.42-55, 31 ago. 2011. Sociedade Brasileira de Computacao - SB. http://dx.doi.org/10.5753/rbie.2011.19.02.42.

PRESSMAN, Roger S. et al. **Software Engineering**: A Practitioner's Approach. 8. ed. Nova Iorque: Mcgraw-hill Education, 2014. 976 p. ISBN 9780078022128.

RIES, Eric. **Minimum Viable Product:** a guide. 2009. Disponível em: <http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html>. Acesso em: 02 nov. 2019.

SHEIKH, Yaser. **Facebook is building the future of connection with lifelike avatars**. 2019. Disponível em: <https://tech.fb.com/codec-avatars-facebook-reality-labs/>. Acesso em: 18 nov. 2019.

WELLER, Martin. **Virtual learning environments:** using, choosing and developing your VLE. Londres: Routledge, 2007. 192 p. ISBN 9780415414302.

Glossário

Admin: usuário com nível de acesso 4; ver seção 3.1 – Situação Atual.

Alloy City Linguistics: empresa especializada em software de cunho linguístico, criadora da plataforma Pantoufle, patrocinadora.

API Alloy: Interface de acesso ao back end da plataforma educativa utilizada pela escola Pantoufle, desenvolvida pela Alloy City Linguistics.

Chapter: conjunto de Lessons.

CL: Cliente AViS local.

Coordinator: usuário com nível de acesso 3; ver seção 3.1 – Situação Atual.

Course: conjunto de Meetings.

CR: Cliente AViS remoto.

Creator: usuário com nível de acesso 2; ver seção 3.1 – Situação Atual.

FLE: Francês Língua Estrangeira, especialização na área de educação.

Lesson: conjunto de Resources.

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

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

Pack: conjunto de Products (exceto outros Packs).

Pantoufle: *1.* Escola especializada em ensino remoto de FLE; *2.* Plataforma de ensino de FLE utilizada pela escola de mesmo nome.

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

Resource: menor unidade do material didático.

Student: usuário com nível de acesso 0; ver seção 3.1 – Situação Atual.

Teacher: usuário com nível de acesso 1; ver seção 3.1 – Situação Atual.

Torrent: protocolo de transmissão de dados P2P.

UE4: Unreal Engine 4.

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

WebSocket: protocolo de comunicação duplex via rede.