Comunicação cliente-servidor de baixa latência com Mobile

Guilherme Freire Silva

Trabalho de Conclusão de Curso
Instituto de Matemática e Estatística
da
Universidade de São Paulo

Orientador: Prof. Dr. Marco Dimas Gubitoso

São Paulo, 2016

Lista de Figuras

Lista de Tabelas

Sumário

1 Motivação

Conhecer o desenvolvimento de um aplicativo para Android, Entender como se dá a comunicação Cliente-Servidor, Fazer uma transmissão de dados eficiente o aplicativo e o servidor.

2 MAPA GERAL DO QUE TA COM TESENO

TODOs: ¿ Explicar o ganho de utilizar JS. ¿ Estrutura Cliente-Servidor ¿ Network Socket ¿ Websocket ¿ Socket.io ¿ Explicar node.js

3 Comunicação Web

A comunicação de dados entre computadores através da Internet é algo muito comum, e necessita de uma arquitetura de rede adequada para ocorrer.

A mais utilizada normalmente é o modelo cliente-servidor. Neste modelo, existe o Servidor, que é o computador que fornece recursos

Há outras arquitetura muito utilizadas também, como peer-to-peer (P2P) por exemplo. Nessa arquitetura, não há um computador central para funcionar como servidor, cada computador conectado (Peer) na rede realiza funções tanto de cliente como de servidor na aplicação que está sendo executada. Essa aplicação tem suas tarefas organizadas e divididas entre os Peers. Essa arquitetura é conhecida principalmente por ser usada para a transmissão de arquivos grandes, como músicas e vídeos. Nessa transmissão, os Peers que tem o arquivo são conectados aos que não tem, e começam a transferência de pequenos pacotes de dados. Esses pacotes não precisam vir ordenados e o Peer receptor os armazena localmente. Uma vez que a transferência é completada, ele ordena os pacotes e monta o arquivo final. Uma vantagem dessa arquitetura é que a tranferência não é limitada pela capacidade de banda de Servidor, e Peers podem se conectar e desconectar sem que haja problemas para o receptor, o arquivo são será corrompido por eventuais problemas de conexão. Um ponto negativo é que, sem um Servidor, não há um controle de que tipos de arquivos estão sendo transferidos (o que abre uma porta para pirataria) e não é fácil interromper uma transferência, uma vez que ela pode ser composta de milhares de conexões. Outro ponto é que não é fácil de se conhecer a procedência dos dados recebidos, a segurança não pode ser garantida. Esta arquitetura não serve para as necessidades do projeto, a hipótese de uso foi descartada após o estudo de sua estrutura.

4 Websockets

5 Device

Device: O aplicativo é criado usando o Cordova. Cordova é uma IDE (?) que faz o porte de programas em Javascript para várias plataformas, como Android e iOS.

¿ Tiveram mil problemas até o Cordova ser escolhido.

O device se conecta ao servidor utilizando Socket.io

Estruturação de um código Web (JavaScript, CSS, HTML5)

6 Server

Server: Usa socket.io para fazer a comunicação com os clients. ¿ Usa node.js para something

7 Browser

Browser: Primeiro client, lançado pelo server; Recebe dados por Socket; Não envia dados;

Visualização de dados; Visualização 3D: Usa Three.js (Biblioteca 3D para js); Usa os dados recebidos para alteração de objetos 3D.

Foram feitas duas páginas de Three.js: 1) Um simples objeto é rotacionado e tem sua cor alterada de acordo com dados recebidos.

2) É a aplicação de um algoritmo de ruído para geração procedural de terreno em um grid. Utiliza os dados recebidos para alterar os parâmetros do algoritmo. No teste inicial ele altera em tempo real o grid, de forma que o envio constante de dados cria um movimento também constante.

¿ Talvez possa fazer um teste de performance, e rodar o algoritmo em um grid muito maior.

8 Parte Subjetiva

Parte Subjetiva:

Evolução do Projeto:

Esse projeto utiliza várias tecnologias que eu nunca tive contato.

Inicialmente a ideia inicial do TCC era usar um Arduino e um conjunto de sensores para coletar dados do meio e mandar de maneira síncrona para um servidor. O servidor então usaria esses dados para fazer algum controle.

Numa primeira discussão com o orientador, a ideia foi rapidamente descartada, pelos seguintes motivos:

Os sensores dele normalmente são imprecisos; A montagem de um dispositivo como o idealizado seria desnecessariamente custosa e muito propensa a erros; Sem uma montagem muito bem executada, o dispositivo ficaria muito frágil; É possível conseguir muitas leituras de sensores utilizando um smartphone.

Com os argumentos apresentados, a decisão foi criar um aplicativo para Android com o mesmo propósito. Todos os argumentos apresentados são resolvidos com essa solução. A implementação é extremamente mais simples; os sensores são muito mais precisos; a estrutura física já está pronta e é um objeto que já está presente em todo o mundo (o produto final atinge grande parte da população); e, por fim, é muito viável a adição ou remoção de funcionalidades, além de muito mais suporte para a plataforma.

Uma vez que a decisão de utilizar um smartphone, foi necessário saber como implementar um aplicativo que tenha acesso aos sensores. A primeira ideia foi utilizar serviço MIT App Inventor.

MIT App Inventor: É um serviço disponibilizado pelo MIT para a criação de aplicativos. Ele é extremamente didático e inclusivo, sua interface não é dada por linhas de código, e sim por blocos lógicos que se encaixam e formam um algoritmo (na prática, é bastante ruim não poder escrever linhas de código livremente). Sua API possui interface para o uso de sensores, além de outras funcionalidades que não foram exploradas neste TCC, como acesso à ferramenta de reconhecimento de voz e ferramentas sociais, como email, mensagem e Twitter. O serviço é bom, porém possui várias restrições, então é difícil de ser usado.

Com certa dificuldade um primeiro aplicativo de teste foi feito, para pegar valores do giroscópio. O próximo passo seria criar um servidor e estabelecer uma conexão entre ambos, porém as opções de conectividade do App Inventor também são muito restritas, então, dada a dificuldade prevista, eu achei melhor deixar essa plataforma de lado no momento e procurar outras alternativas para a criação de um app.

A ferramenta escolhida foi o Kivy.

9 Kivy

É um framework Open Source de Python. Seu objetivo é o desenvolvimento rápido de aplicações multiplataforma que fazem o uso de interfaces inovadoras, como telas com Multitouch e sensores de movimento. Sua API, por exemplo, lida com eventos de mouse, teclado e toques de tela. Ele

também possui um foco na criação de apps com NUI (Natural User Interface). NUI é uma metodologia de interface cuja proposta é ser invisível ao usuário, e apresentar uma experiência natural e intuitiva. Seu principal foco é evitar grandes barreiras durante o aprendizado. Através de decisões de design, o usuário tem um entendimento do software sem grandes dificuldades, à medida que a complexidade da interação aumenta.

[TODO] Para exemplificar a facilidade de desenvolvimento de lógica e de uma interface gráfica, abaixo tem o código para se criar um jogo de Pong.

10 Cordova

Cordova: O Apache Cordova é um framework open-source para desenvolvimento mobile. Ele permite o uso de tecnologias Web, como HTML5, CSS3 e JavaScript. Seu desenvolvimento é multiplataforma, então há uma API de alto nível para acessar módulos desejados, como sensores, arquivos e rede. Isso implica em uma interface abstrata para o uso das features, de forma que código desenvolvido será executado em todas as plataformas oferecidas e o desenvolvedor não precisa se preocupar com os detalhes de cada uma na hora da implementação.

A aplicação é implementada como uma página Web em um arquivo 'index.html' que referencia os recursos necessários, como CSS, JavaScript, imagens e arquivos de mídia. A parte lógica é feita em JavaScript, e a renderização em HTML5 e CSS3. O HTML é enviado para a classe "Wrapper" específica da plataforma escolhida, na qual estão definidos os detalhes de implementação. Ela também tem incorporada um browser nativo, o WebView, que executará o programa Web.

Para a comunicação entre o aplicativo e os componentes nativos de cada plataforma, é necessária a instalação de Plugins. Cada Plugin é uma biblioteca adicional que permite ao WebView se comunicar com a plataforma nativa na qual está rodando. Eles provêm acesso a funcionalidades que normalmente não estão disponíveis em aplicações Web. Essas ferramentas são disponibilizadas para o desenvolvedor através de uma expansão da API inicial, que serve de interface e toma para si os detalhes da implementação em cada plataforma, simplificando o uso para o desenvolvedor. Há um conjunto principal de Plugins, chamado de Core, que é mantido pelo próprio Cordova. Nele estão contidos os que acessam as principais funcionalidades de um dispositivo mobile, como acesso ao acelerômetro, câmera, bateria e geolocalização. Há também Plugins desenvolvidos por terceiros (em geral pela própria comunidade ativa) que trazem relações com outras funcionalidades, que podem ser exclusivas de uma plataforma. Enquanto no Core há apenas umas poucas

dezenas, a lista de Plugins criada pela comunidade oferece centenas com as mais diversas funcionalidades, como um para compras dentro do App e um para enviar notificações para dispositivos vestíveis (Smartwatches, por exemplo). A criação de um Plugin é simples e incentivada, no site do Cordova há um tutorial.

O método de desenvolvimento descrito até agora, focado em várias plataformas, é um dentre dois possíveis Workflows disponíveis no Cordova e seu nome é "Cross-platform" (CLI). Ele deve ser usado se o aplicativo desenvolvido pretende abranger a maior variedade de Sistemas Operacionais mobile, com pouca ou nenhuma ênfase em um desenvolvimento em uma plataforma específica. O segundo Workflow é chamado "Platform-centered", e deve ser usado se o projeto é focado para uma única plataforma e se pretende modificá-la em baixo nível.

A arquitetura interna está descrita no diagrama:

11 Bibliografia