Implementação de um Sistema de Quiz no Aplicativo WebLectures

O. MIranda *R. Azevedo*

Technical Report - IC-PFG-19-02 - Relatório Técnico

June - 2019 - Junho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Implementação de um Sistema de Quiz no Aplicativo WebLectures

Otávio Vansetti Miranda[1], Rodolfo Jardim de Azevedo[2]

[1] Aluno de graduação do Instituto de Computação Universidade Estadual de Campinas (UNICAMP), CampinasSP, Brasil. o158319@dac.unicamp.br

[2] Professor Doutor do Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Campinas-SP, Brasil. rodolfo@ic.unicamp.br

Resumo. Este relatório técnico descreve um projeto final de graduação, com o objetivo de detalhar o projeto e implementação de um sistema de questionários para o aplicativo para dispositivos móveis WebLectures. O foco deste projeto foi aprimorar o aplicativo ao aplicar a metodologia de ensino Peer Instruction (FAGEN *et al*, 2000), de forma a introduzir a possibilidade a um professor usuário de fazer perguntas de múltipla escolha a serem respondidas por seus alunos em uma página WEB, durante ou após a aula. Os resultados são, então, adicionados ao *whiteboard* do aplicativo, permitindo sua análise de forma fluida durante o decorrer da aula.

As principais ferramentas utilizadas neste projeto foram o Android Studio para criação do código e testes (Debug com simulador), a plataforma Angular 6 para a criação da página WEB de acesso para respostas e o sistema Firebase, na qual o Banco de Dados foi implementado.

Este projeto contemplou a criação da página web em que os questionários serão respondidos (design de front-end e back-end, incluindo envio e recuperação de dados ao servidor, controle de concorrência, etc.) e atualização do sistema de reprodução (adição dos questionários, interatividade, etc.) do aplicativo WebLectures.

Palavras-Chave: Peer Instruction, Quiz, aplicativo Android, Angular, Firebase, Banco de Dados, metodologia de ensino.

SUMÁRIO

1. INTRODUÇÃO

O avanço da tecnologia tem propiciado a adoção de ferramentas que possibilitam a melhoria do ensino e da aprendizagem, estimulando a ampliação do alcance das aulas, uma maior participação dos alunos, a consolidação de conteúdos apresentados, entre outras vantagens.

Dentre estas ferramentas, o WebLectures é um aplicativo para uso acadêmico destinado à gravação e exibição de aulas, possibilitando que o conteúdo das sessões possa ser divulgado posteriormente na forma de páginas WEB. Esta gravação das aulas contempla desde o áudio do professor expondo o conteúdo, passando pelo slide projetado pelo professor e ainda anotações adicionadas ao próprio slide ou feitas em um quadro branco virtual. A captura de vídeo não é necessária, ficando mais simples o processo de exportação da aula.

A adoção do WebLectures requer um tablet onde o professor carrega os slides da aula e faz suas anotações, explicações, exemplos e correções. Utilizando um projetor, o professor expõe a aula para os alunos na sala e, ao mesmo tempo, grava a sessão para uso posterior. O WebLectures constitui, assim, uma aplicação da tecnologia que é útil para aperfeiçoar o ensino, favorecendo alunos e professores com um acervo de mais de mil aulas já gravadas.

Tendo em vista as vantagens do emprego do aplicativo WebLectures, considerou-se realizar um aprimoramento, integrar a metodologia Peer Instruction ao aplicativo. A ideia é a incorporação de perguntas formuladas com o propósito de aumentar a participação do aluno na aula e permitir a identificação de eventuais dificuldades com o conteúdo (FAGEN et al, 2000). A utilização da Peer Instruction resultou em melhoria no desempenho dos alunos em aulas de física na Universidade de Harvard (CROUCH; MAZUR, 2001), bem como em aulas de no curso de Sistemas de Informações na Universidade Estadual de Campinas (CACEFFO et al, 2018).

2. JUSTIFICATIVA

A metodologia Peer Instruction pode trazer consideráveis benefícios ao aprendizado e, nesse sentido, aprimorar o aplicativo WebLectures através da implementação do sistema de questionário ajuda a otimizar a aplicação de questões durante aulas. Adicionalmente, o aplicativo assim aprimorado pode vir a se tornar uma ferramenta valiosa para a aplicação do Concept Inventory desenvolvido para a disciplina de Algoritmos e Programação de Computadores pela Universidade Estadual de Campinas (CACEFFO *et al*, 2016).

3. OBJETIVOS

O objetivo deste Projeto Final de Graduação foi desenvolver e implementar um sistema para responder questionários gerados no aplicativo WebLectures de forma intuitivo para usuários e funcional para a metodologia Peer Instruction. Adicionalmente, foi implementada a opção de responder questionários durante sua reprodução, após o término da aula. Dessa forma possibilitando a alunos em sala de aula ou em casa possam responder às questões aplicadas.

4. DESENVOLVIMENTO DO TRABALHO

A página web para acesso e resposta dos questionários foi desenvolvida utilizando padrão Angular 6. Utilizando a linguagem Typescript, uma variante de Javascript, foi desenvolvida uma página de acesso simples e clara para permitir a escolha de um quiz específico e uma página de resposta igualmente resumida, a fim de evitar problemas de escala em dispositivos móveis.

Em seu front-end, a página de acesso consiste de um cabeçalho, com o título da página e um logotipo da Unicamp, uma caixa de texto, que aceita o código numérico de um questionário, e um botão de submissão. O back-end inclui um teste de acesso que verifica se o número inserido é válido, ou seja, se um quiz existe com o número adquirido, antes de liberar o uso do botão de submissão. Este, por sua vez, executa a transição para a página de resposta, sendo necessário encapsular o número inserido na caixa de texto no URL da págino, a fim de permitir que seja utilizado na página de respostas.

A página de resposta tem em seu front-end o mesmo cabeçalho que a página de acesso, adicionando a este o número e título do questionário selecionado. Além disso, entre 2 e 6 botões serão mostrados, nos quais o usuário aluno pode clicar para fazer sua escolha. Cada botão tem uma letra, de “A” a “F”, sendo que os primeiros dois sempre são exibidos, já que pelo menos duas opções são necessárias para um questionário, enquanto os demais são condicionalmente renderizados de acordo com o número de escolhas decidido na criação do questionário. Por fim, existe um botão de retorno, que cancela o processo e retorna à página de acesso.

O back-end da página de resposta adquire o número do quiz, enviado da página de acesso pelo URL, e inicializa uma estrutura de dados para conter as informações do questionário. Uma função é executada para efetuar a requisição de dados do quiz, enviando o número e recebendo os dados de título e número de opções, que serão, por sua vez, utilizados no front-end, como mencionado no parágrafo acima. Assim que o usuário seleciona um dos botões de resposta, uma requisição de incrementação é enviada ao servidor e o sistema retorna à página de acesso.

Além da página de acesso e resposta de questionários, esse projeto incluiu a modificação do sistema de reprodução de aulas (player) a fim de fornecer suporte às novas funcionalidades de questionário. Esse sistema se baseia em um relatório de ações, que registra todo toque ou traço feito pela caneta e quaisquer comandos usados durante a gravação de uma

aula, como mudanças de páginas ou a criação de novos whiteboards. Foi necessário, portanto, adicionar entradas para as ações de criação e finalização de um questionário.

Ao final da gravação de uma aula o sistema de reprodução gera um arquivo HTML, que permite a reprodução da classe em uma grande variedade de plataformas. Para permitir que os questionários sejam não apenas vistos mas também respondidos por espectadores, foi adicionada uma página interativa ao sistema de reprodução, que apresenta botões para cada opção do questionário ao mesmo tempo que a reprodução é pausada, permitindo que o usuário responda a pergunta calmamente, de forma similar a como ocorreria durante a aula. A aula é retomada quando uma escolha é selecionada, mostrando os resultados do quiz.

5. FERRAMENTAS UTILIZADAS

5.1 Android Studio

O Android Studio é um Ambiente Integrado de Desenvolvimento, da sigla em inglês IDE, para desenvolvimento de aplicativos Android, utilizando as linguagens Java, Javascript e, mais recentemente, Typescript e Kotlin, originalmente baseado no IntelliJ IDEA. As modificações do sistema de reprodução foram feitas utilizando ambiente, assim como seus testes utilizando suas interfaces de debug. A possibilidade de usar um dispositivo móvel conectado via USB ao computador para depuração foi uma funcionalidade extremamente útil para o projeto, permitindo mais rápidas iterações e correções.

5.2 Angular

A plataforma Angular utiliza a linguagem Typescript para facilitar o desenvolvimento de aplicações WEB, mobile e para desktops, fornecendo ferramentas e padrões para permitir a criação mais organizada e eficiente de web-apps. Ela fornece novas opções para desacoplar componentes, permitindo uma maior modularização e facilita a comunicação de páginas e componentes utilizando um sistema mais prático de roteamento. Mesmo adicionando várias novidades, o Angular continua compatível com a muitas ferramentas conhecidas, como por exemplo o Node.JS, permitindo uma rápida adaptação de desenvolvedores experientes.

Além de adicionar novas ferramentas de desenvolvimento, o Angular inclui um suporte nativo para testes unitários e de integração, utilizando ferramentas como Jasmine e Protractor, além de sua própria plataforma de testes, o Karma. Esses fatores, somados à rápida atualização e desenvolvimento ativo da plataforma, implicam em um sistema moderno e ágil para o desenvolvimento de aplicações WEB da mais alta qualidade.

5.3 Firebase e Firestore

O Firebase é uma plataforma para a implantação de aplicativos WEB, fornecendo hospedagem, estatísticas e suporte a várias ferramentas avançadas como análise de performance e erros, aprendizado de máquina, testes e projeções de escala. Porém, o mais importante serviço é o Firestore, um banco de dados não relacional, flexível e escalonável para desenvolvimento de dispositivos móveis, webs e servidores. O funcionamento se dá, resumidamente, através de documentos (conjuntos de strings, números, etc) e coleções desses documentos que são

armazenados no banco de dados, no qual existe a possibilidade de consulta, leitura e escrita entre diferentes sistemas para o mesmo banco de dados.

O Firebase é usado neste projeto para hospedagem e para utilizar o Firestore. Sua API tem ótima compatibilidade com aplicações Android e Angular, permitindo uma eficiente integração entre ambos os clientes e o servidor na nuvem. Além disso, as funções de acesso ao Firestore são assíncronas, permitindo que inúmeros usuários enviem as respostas de seus questionários simultaneamente, sem risco de problemas de concorrência ou de performance. Além disso, o serviço é gratuito para um sistema da escala como a que este projeto espera ter quando for implantado.

6. RESULTADO DA IMPLEMENTAÇÃO EM DETALHES

6.1 Acesso e Resposta de Questionários

Ambas as páginas WEB para acesso e resposta dos questionários contam com um cabeçalho comum, que contém o título do aplicativo, “Web Lectures Mobile Quiz Answers”, e um logotipo da Unicamp. Esse cabeçalho está implementado no componente pai de ambas as páginas, de forma que não é necessário implementá-lo duas vezes. Além disso, o estilo da página é definido pelo padrão Material, implementado na aplicação como um todo através do arquivo “styles.css”.

6.1.1 Página de Acesso aos Questionários

A página de acesso aos questionários é composta por uma caixa de texto e um botão de confirmação. A caixa de texto é implementada utilizando a ferramenta de formulários da plataforma Angular, utilizando as funções “FormBuilder”, “FormGroup” e “Validators” da biblioteca “angular/forms”. Ela contém o texto “Quiz ID” enquanto não é preenchido e alerta ao usuário que seu preenchimento é obrigatório caso seja ignorado, ambas funcionalidades existentes no Angular.

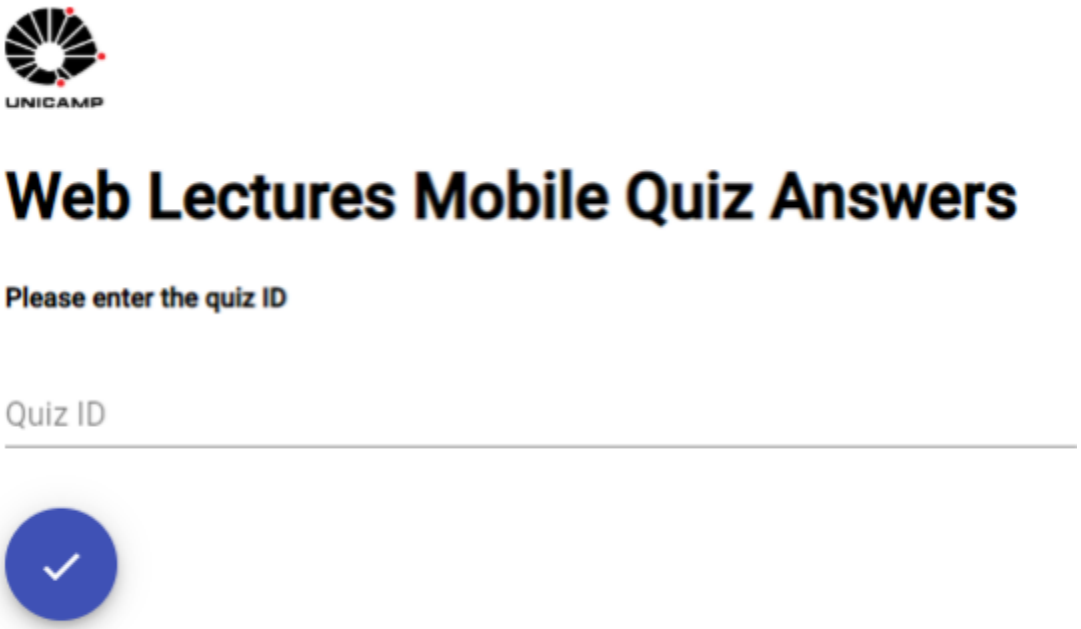

Figura 1: Página de Acesso de Questionários Inicial

O botão de confirmação executa a função “OnFormSubmit”, que executa o comando “router.navigate” do roteador, um sistema do Angular que permite a comunicação entre diversas páginas de forma prática e eficiente. Este comando adquire o valor preenchido no formulário e

carrega a página de resposta, enviando a esta o identificador do questionário que foi inserido na caixa de texto. A função não executa se o campo do formulário não for preenchido.

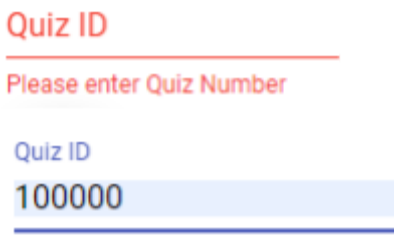

Figura 2: Caixa de texto com Alerta ou Preenchida

6.1.2 Página de Resposta de Questionários

A página de resposta de questionários contém, visualmente, o título do questionário e até seis opções de resposta, além de um botão para retornar para a página de acesso. Assim que a página é carregada a função “getQuizOptions” é invocada, recebendo o identificador do questionário, adquirido pelo roteador, como parâmetro. Essa função, por sua vez, chama a função “getBoard” que recebe do servidor Firestore as informações do questionário escolhido. Graças às bibliotecas do Firebase e Firestore, só é necessário inicializar a conexão com o banco correto (“Quizzes”) utilizando o comando “firebase.firestore().collection('Quizzes')”, já que o aplicativo estará sendo executado na própria plataforma do Firebase.

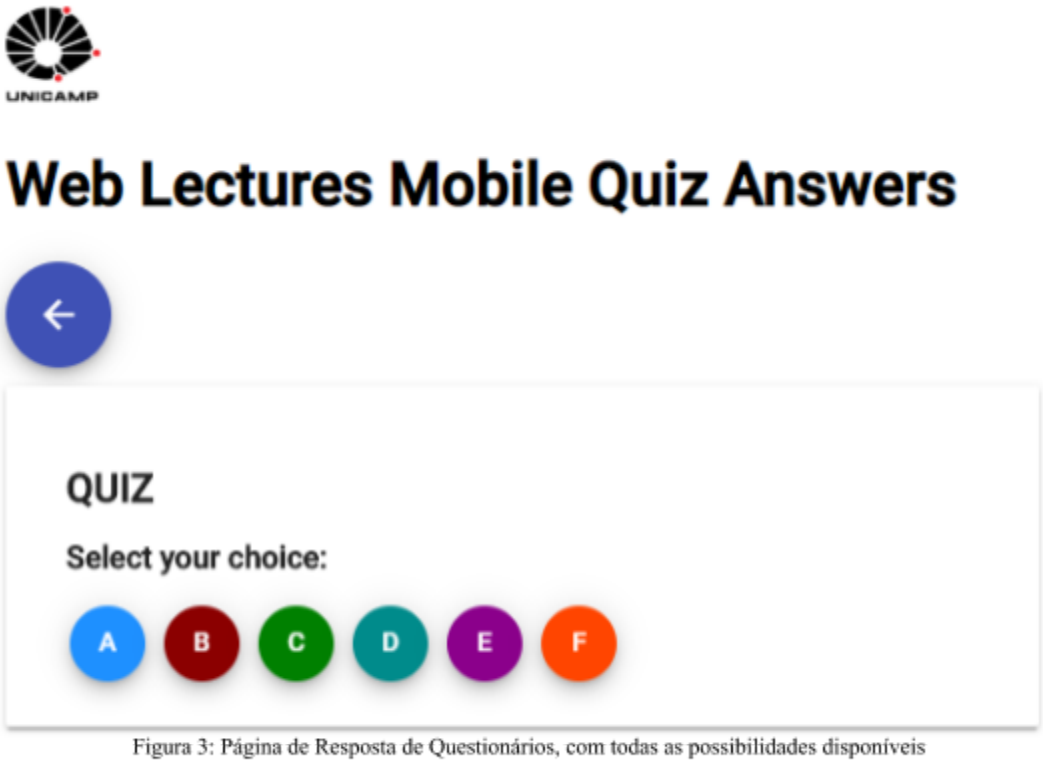

Figura 3: Página de Resposta de Questionários, com todas as possibilidades disponíveis

Uma vez carregadas as informações, o número adequado de botões para respostas é ativado, renderizando-os para o usuário, junto com o título do quiz. A partir desse momento o usuário pode selecionar uma das opções, o que invoca a função “increment()”. Essa função começa o processo de incremento do contador da alternativa selecionada chamando a função “incNum(num, id)”, que inicializa a conexão com o Firestore e executa uma transação com o comando “firebase.firestore().runTransaction(function (transaction)”. A transação adquire o valor atual do contador, o incrementa e, através da função “transaction.update”, o envia de volta ao banco de dados. Vale notar que a transação é assíncrona, garantindo que não haverão problemas de concorrência entre usuários. Após a atualização do servidor, o roteador é invocado para carregar a página de seleção de questionários novamente, retornando o aplicativo a seu estado original.

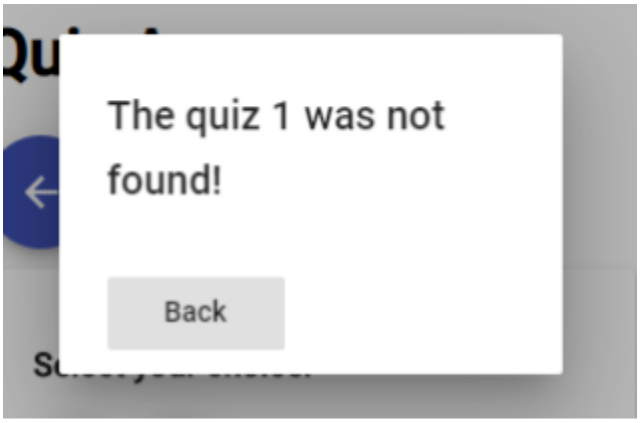

Figura 4: Alerta de Questionário Não Encontrado

Caso o questionário com o identificador recebido não seja encontrado uma mensagem é exibida para o usuário, utilizando um “modal.component” do Angular. Esse componente é essencialmente uma página separada que é exibida sobre a página de resposta, contendo o alerta de erro, com o identificador do quiz não encontrado, além de um botão que retorna o usuário para a página de seleção. Sua implementação é baseada na função “dialog.open()” das bibliotecas do Angular, com elementos da biblioteca Material auxiliando a manter seu funcionamento e estilo consistentes com o resto da aplicação.

6.2 Introdução de Questionários no Sistema de Reprodução

6.2.1 Alterações no Sistema de Gravação

Para implementar os questionários no sistema de reprodução, foi primeiro preciso modificar o sistema de gravação. Esse sistema se baseia no registro de mensagens de texto simples salvas em um relatório, guardando cada ação feita, como o carregamento de novos slides ou traços com a caneta, para permitir sua reprodução posterior. A função responsável pelo registro é "logActivity()" da biblioteca "Utils", que recebe uma cadeia de caracteres específica para cada tipo de ação.

Foi necessário adicionar uma nova chamada da função logActivity para registrar a criação de um questionário e uma para seu término, com o intuito de permitir que a pausa na aula para permitir que os alunos respondam o questionário sejam cortadas da gravação final. No momento em que o quiz é criado é preciso registrar o número de alternativas, além de seu identificador. Com o término acionado pelo professor, é importante marcar não apenas o momento da ação, mas também qual é o slide que contém os resultados, de forma a permitir sua exibição durante a reprodução. O slide em si deve ser salvo utilizando a função "saveToFile()", que cria um arquivo PNG do novo slide, o qual será guardado com os demais arquivos para a reprodução.

Adicionalmente, é preciso pausar o registro de outras ações durante o período de execução do questionário, usando a função "pauseLog()", que elimina novas entradas no relatório conforme são inseridas. Isso é necessário para garantir a sincronização do áudio e slides pós-questionário durante a reprodução, como discutido na seção 6.2.2.

6.2.2 Alterações no Sistema de Reprodução

O sistema de reprodução sofreu alterações similares, mas mais complexas, que o de gravação. Em contrapartida ao sistema descrito acima, o player lê e parseia o relatório, a fim de reproduzir cada ação registrada em sincronia com o áudio gravado. Para implementar a operação descrita na seção 6.2.1, foi preciso implementar uma nova função, "quiz()", utilizando diversas funções existentes no arquivo "player.js", além de novas funcionalidades de interação criadas especialmente para a resposta dos questionários.

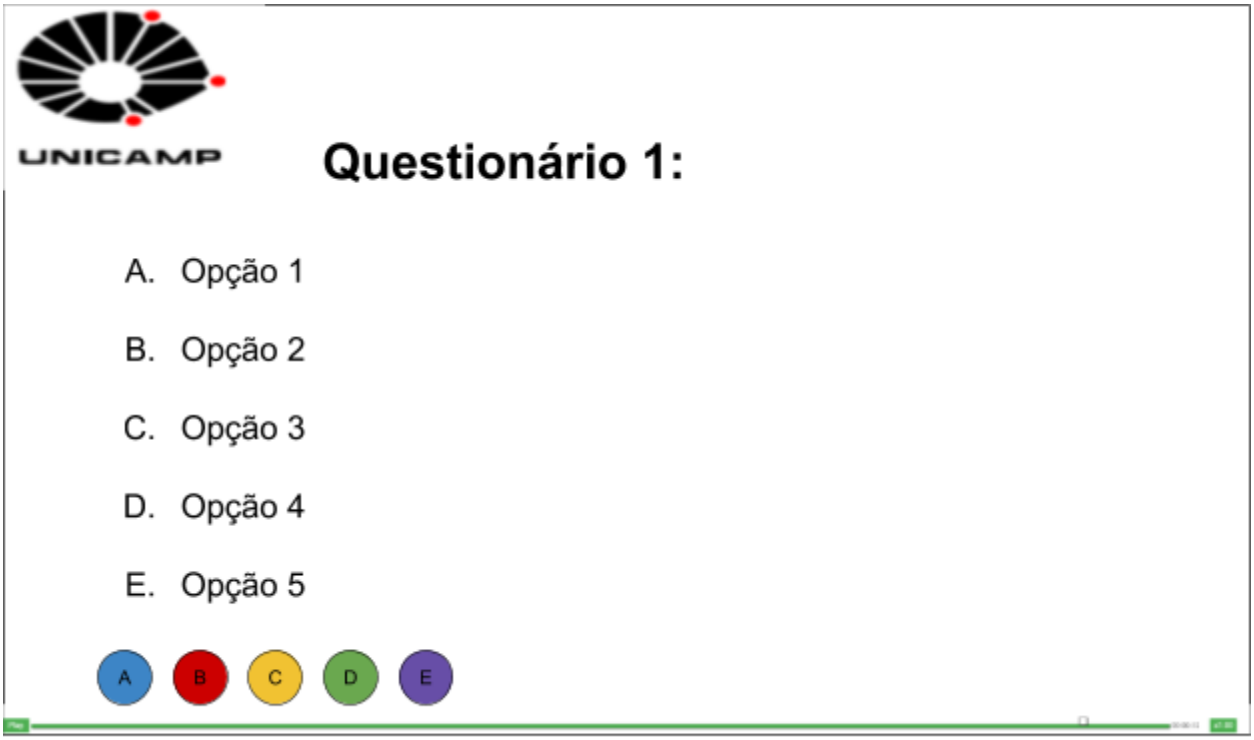

Figura 5: Sistema de Reprodução exibindo um Questionário

A função quiz é invocada quando o sistema de reprodução lê o início de um quiz, imediatamente chamando a função “pause()” para pausar a reprodução. Em seguida, botões são renderizados na tela de reprodução, em número condizente com o questionário recebido, permitindo que o espectador interaja com a reprodução de forma similar a um aluno no momento da aula, com a exceção de poder aguardar quanto tempo julgar necessário para responder. Uma vez escolhida a resposta, a reprodução retoma com o slide que contém os resultados do questionário, carregado com as funções “loadBackGround()”, que adquire o arquivo PNG do slide, “context.clearRect()”, que limpa a tela para permitir a exibição do slide e “context.drawImage()”, que exibe o slide carregado para o espectador. Adicionalmente, o áudio é sincronizado com o momento de término do quiz, adquirido com a próxima entrada do relatório de gravação, usando o comando “audio.currentTime” para isso.

7. LIMITAÇÕES DA SOLUÇÃO

Este projeto conta com algumas limitações, sejam questões que estão além de seu escopo ou simplesmente que dependem de fatores externos fora do controle do time de desenvolvimento. Uma delas é a possibilidade de distração dos alunos ao utilizarem dispositivos móveis em sala de aula, já que seria difícil garantir que estes fossem apenas usados para responder os questionários. Existe a possibilidade de se estabelecer uma rede Wi-Fi com acesso limitado à internet, a fim de limitar o potencial de distração de um dispositivo, mas isso não evitaria o uso de redes de dados móveis ou de conteúdo offline.

Adicionalmente, existe uma limitação de tráfego no sistema Firebase, que limita o acesso ao banco de dados para 50.000 operações de leitura, 20.000 de escrita. Em uma sala hipotética com 50 alunos, seria possível em média criar 385 questionários diariamente, sendo 51 operações de leitura e 52 de escrita para cada. Caso a base de usuários cresça ao ao ponto de exceder esses limites, é possível aumentar a quota diária para permitir até 2300 questões por apenas 7,2 dólares por mês, ou 24 centavos de dólar por dia.

Finalmente, não é possível criar mais de um quiz em simultâneo. Para manter o uso do aplicativo simples e intuitivo, foi decidido que a habilidade de se fazer questionários em paralelo não seria implementada. Além de não ser um caso de uso particularmente comum, a possibilidade de se fazer dois questionários em rápida sequência deve ser suficiente para compensar essa limitação

8. CONCLUSÃO

Este projeto desenvolveu um sistema de questionários para o aplicativo WebLectures, com o intuito de facilitar a aplicação dos modelos de Peer Instruction (CROUCH; MAZUR, 2001) (CACEFFO et al, 2018) em sala de aula de forma integrada e intuitiva. Foi possível aprimorar o WebLectures para permitir que professores apliquem perguntas de múltipla escolha pela internet, através do aplicativo Android, de forma que podem ser rapidamente respondidas por seus alunos, via um web-app. Sua implementação priorizou a integração clara e intuitiva dos novos elementos, a fim de minimizar o impacto das mudanças na produtividade de professores e alunos e maximizar os benefícios oferecidos. Adicionalmente, o projeto foi uma excelente oportunidade para explorar novos métodos de implementação e implantação de aplicações WEB, fazendo uso das mais recentes técnicas de hospedagem na nuvem.

9. REFERÊNCIAS BIBLIOGRÁFICAS

CACEFFO, R.; GAMA, G.; AZEVEDO, R. **Exploring Active Learning Approaches to Computer Science Classes.** SIGCSE, Baltimore, MD, p. 922-927, fev. 2018.

CACEFFO, R.; WOLFMAN, S.; BOOTH, K. S.; AZEVEDO, R. **Developing a Computer Science Concept Inventory for Introductory Programming.** SIGCSE, Memphis, TN, p. 364-369 mar. 2016.

CROUCH, C. H.; MAZUR, E. **Peer Instruction: Ten years of experience and results.** American Journal of Physics, Melville, NY, v. 69, n. 9, p. 970-977, set. 2001.

FAGEN, A. P.; YANG, T.; CROUCH, C. H.; MAZUR, E. **Factors That Make Peer Instruction Work: A 700-User Survey.** AAPT Winter Meeting, Kissimmee, FL, 2000.