APLICATIVO PROTÓTIPO PARA SOLICITAÇÃO DE REQUERIMENTOS DISCENTES

DOI: 10.37702/2175-957X.COBENGE.2021.3597

Renata Machado de Miranda - rmm@poli.br
Universidade de Pernambuco
Rua Professor Augusto Lins e Silva 80
51030-030 - Recife - PE

Lucas Abismael da Rocha Soares - lars@poli.br
Universidade de Pernambuco
RUA 85 QD 62 BL 03 APT 102 S/N
53441-310 - PAULISTA - PE

VERUSCA SEVERO DE LIMA - verusca.severo@poli.br
UNIVERSIDADE DE PERNAMBUCO
RUA FRANCISCO LACERDA 90
50741-150 - RECIFE - PE

Resumo: *Este projeto tem como objetivo desenvolver um aplicativo protótipo para smartphones com o React Native, em que o aluno poderá solicitar requerimentos da Escola Politécnica de Pernambuco - POLI, Universidade de Pernambuco - UPE, como por exemplo: atividade complementar, segunda chamada de exercício escolar, histórico escolar. A implementação dessa aplicação, além de modernizar, irá simplificar e otimizar o processo atual ao diminuir a quantidade de documentos impressos utilizados e eliminar etapas presenciais, fator que se demonstrou muito importante nos tempos atuais de isolamento social, por causa da COVID-19. Além disso, ao tornar virtuais todas as solicitações, o estudante poderá acompanhar o andamento de seu requerimento, sendo possível saber quando a solicitação foi concluída. Para isso foi feita a criação de um banco de dados MySQL para armazenar tanto as informações de cadastro dos alunos, como também as requisições realizadas.*

Palavras-chave: *Aplicativo Protótipo, React Native, Requerimentos, MySQL.*

APLICATIVO PROTÓTIPO PARA SOLICITAÇÃO DE REQUERIMENTOS DISCENTES

1 INTRODUÇÃO

A popularização dos aplicativos móveis se deu com a chegada dos sistemas operacionais iOS e *Android* em 2007 e 2008, respectivamente (COUTINHO, 2014). Essa disseminação é consequência da praticidade que os aplicativos fornecem na execução de inúmeras atividades, como por exemplo: interação social, operações financeiras, telemedicina, entre outros.

É importante ressaltar, ainda, que uma das principais funcionalidades que os aplicativos possuem é possibilitar que o usuário consiga realizar tarefas de forma remota (TAROUCO, 2013). Essa funcionalidade é algo que se tornou indispensável, especialmente no período em questão de isolamento social, devido à pandemia de COVID-19 (do inglês, *Coronavirus disease 2019*).

É impossível apontar um setor da sociedade que não tenha sido amplamente influenciado pelo uso de aplicativos móveis; a saúde, a educação, o comércio, a indústria e as relações interpessoais, por exemplo, têm sido diretamente beneficiados. Na perspectiva do âmbito escolar, o uso de aplicativos é uma tendência notória nos últimos anos, pois é um recurso que gera facilidade na comunicação, no armazenamento de dados e na execução de procedimentos (PAES; SOUZA; COSTA, 2018). Dessa maneira, é inevitável não pontuar aspectos a melhorar em algumas estruturas administrativas da Escola Politécnica de Pernambuco – POLI, Universidade de Pernambuco – UPE, sendo um deles a forma utilizada para solicitação de requerimentos por parte dos discentes, tais como, segunda chamada de exercício escolar, banca examinadora, declaração de matrícula, entre outros. A solicitação desses requerimentos exige do discente a execução de diversas etapas e procedimentos presenciais, o que torna o processo mais complexo e demorado do que o esperado.

Diante do exposto, este projeto tem como principal objetivo o desenvolvimento de um aplicativo protótipo para *smartphones*, em que o aluno conseguirá solicitar requerimentos, como também acompanhar de forma remota o andamento de suas solicitações, eliminando etapas presenciais, e tornando possível a economia de papel na universidade, já que a maioria do processo será feito virtualmente, com exceção da retirada do documento final, em alguns casos.

O restante do artigo encontra-se organizado como segue. A Seção 2 apresenta a metodologia do trabalho, em que são detalhadas todas as ferramentas utilizadas para a implementação do protótipo. A Seção 3 apresenta os resultados, em que é possível observar o sistema desenvolvido. Por fim, a Seção 4 com as conclusões do trabalho.

2 METODOLOGIA

Para a implementação do protótipo, foi utilizado o *Visual Studio Code* como base para edição do código, como também o *React Native*, que é um *framework* que torna

possível a criação de aplicativos multiplataforma utilizando apenas a linguagem *JavaScript.* Para a utilização do *React Native*, optou-se por fazer a instalação da ferramenta *Expo CLI*, facilitando o desenvolvimento do protótipo. Foi utilizado o *Android Studio*, que possui emuladores de celulares *Android,* para realizar os testes e simulações.

No lado do *back-end* (parte que contém a implementação das regras do sistema) foi utilizado o *Node.js*, que serviu como API (do inglês *Application Programming Interface*), junto com o *nodemon* (para monitoramento de alterações no código), *Express.js*, obtendo-se a administração das requisições, e o *Sequelize,* que ofereceu um melhor gerenciamento do *MySQL* (banco de dados escolhido para a aplicação). Por fim, foi utilizado o *Cloudinary* como serviço de nuvem.

Para realizar a programação do protótipo também foram utilizadas: a documentação do *React Native* (REACT NATIVE, 2021), documentação do *Expo CLI* (EXPO, 2021), documentação do *Sequelize* (SEQUELIZE ORM, 2021).

2.1 *React Native* e *Expo CLI*

O *React Native* é uma série de ferramentas que viabilizam a criação de aplicações móveis para sistemas operacionais *iOS* e *Android* (ROMERO; SAAD; BASTOS, 2018). É um projeto desenvolvido pelos engenheiros do *Facebook*, que transforma o código *JavaScript* em código nativo do sistema operacional do aparelho, o que possibilita o desenvolvimento de aplicativos híbridos, em que rodam tanto no *Android*, como no *iOS*. O uso do *React Native* apresenta diversas vantagens, dentre as quais pode-se citar: o carregamento mais rápido da aplicação; a experiência do usuário se torna mais fluida; se obtém melhor integração entre funções nativas do celular como câmeras, GPS; entre outros (ESCUDELARIO; PINHO, 2020).

Antes do surgimento do *React Native*, era mais complexo e mais caro desenvolver aplicações nativas que fossem simultâneas para *iOS* e *Android*, pois o desenvolvedor, além de ter que aprender duas plataformas e linguagens diferentes (*Objective-C/Swift* para *iOS* e *Java/Kotlin* para *Android*), não conseguia aproveitar partes do código de uma plataforma para a outra. Com o *React Native*, o código pode ser aproveitado em até 100%, permitindo uma redução no custo e na duração do desenvolvimento (ESCUDELARIO; PINHO, 2020)

Quando ainda não existia a opção de iniciar o desenvolvimento da aplicação pelo *Expo CLI*, para iniciar um projeto com *React Native* era necessário baixar e instalar diversas dependências para preparar o ambiente antes de iniciar o desenvolvimento do código. Além disso, desenvolvedores que utilizam computadores com sistema operacional *Windows* não conseguem fazer a instalação das ferramentas necessárias para desenvolver projetos para *iOS*, precisando utilizar um *Mac* (Computador da empresa *Apple Inc*) para isso. Com um computador com sistema operacional *macOS*, é possível desenvolver projetos tanto para *iOS*, como para *Android*, precisando baixar as ferramentas adequadas: *JDK, Android Studio* (para *Android*), *Xcode, CocoaPods* (para *iOS*), *Node.js* e *Watchman*. Já nos computadores *Windows,* como só é possível instalar ferramentas necessárias para desenvolver aplicativos *Android*, então necessitam de menos programas, precisando instalar: *Node.js, JDK* e *Android Studio* (REACT NATIVE, 2021).

Atualmente, o desenvolvedor tem a opção de começar a desenvolver a aplicação com o *Expo CLI*, em que não faz-se necessário ter ambientes configurados separadamente para *Mac* ou para *Windows*, tudo que precisa ser instalado para começar o desenvolvimento da aplicação é o *Node.js* e o próprio *Expo* (esta sendo feita no *prompt* de comando do computador), trazendo uma maior facilidade e rapidez no processo de inicialização da aplicação (REACT NATIVE, 2021).

Para realização de testes em tempo real, é preciso fazer a instalação do aplicativo móvel do *Expo*, que pode ser encontrado tanto no *Play Store* (*Google*), como na *App Store* (*Apple*). É possível utilizar um *smartphone* (podendo ser *Android* ou *iOS*), basta escanear o *QR Code* gerado pelo *Expo*, como também pode-se optar por um emulador de celular. Com um computador com sistema operacional *Windows* só é possível ter acesso a um emulador *Android*.

2.2 *Node.js*

Uma API faz a comunicação entre sistemas, funcionando como uma ponte para troca de informações entre aplicativo e banco de dados. Como forma de integrar o banco de dados com o aplicativo, trazendo benefícios como segurança de dados e facilidade na troca de informações, foi utilizado o *Node.js* como API (JACOBSON; BRAIL; WOODS, 2011).

O *Node.js* é um ambiente que possui multiplataforma, o que permite a possibilidade de criar qualquer tipo de aplicativo e ferramenta do lado servidor (*back-end*) em *JavaScript* (SILVA; SOBRAL, 2017; NODE.JS, 2021). Porém, algumas tarefas comuns no desenvolvimento de uma aplicação não são possíveis utilizando apenas o *Node*. Para se obter um melhor gerenciamento de diferentes requisições *HTTP,* como também mostrar respostas do banco de dados de maneira dinâmica, foi feita a instalação do *framework Express.js*. Para fazer a monitoração de todas as alterações no código da aplicação e reiniciar automaticamente a API quando necessário, foi feita a instalação do *nodemon* (RUBENS, 2017).

Também foi utilizado o *Sequelize*, que oferece uma maior facilidade no gerenciamento do banco de dados. Com ele foi possível criar, migrar e buscar dados no banco de dados (SEQUELIZE ORM, 2021).

2.3 *MySQL*

O *MySQL* é um servidor de banco de dados que gerencia e armazena os elementos e os organiza em forma de tabelas (ORACLE, 2021). Para utilizá-lo foi preciso fazer a instalação da ferramenta *XAMPP* (DVORSKI, 2007), que tornou possível a ativação do *MySQL* e do *Apache* (*software* multiplataforma que facilita e garante a comunicação entre cliente e servidor). Com isso, permitiu-se a utilização da aplicação *phpMyAdmin* (plataforma administradora de banco de dados em *MySQL*), possibilitando o acesso e o gerenciamento das informações que foram armazenadas no banco de dados (PHPMYADMIN, 2021).

Para a aplicação do projeto foi necessário criar um banco de dados, tanto para armazenar as informações dos alunos cadastrados no aplicativo, como também para receber os dados das solicitações de requerimentos e enviar para o protótipo o conteúdo das solicitações finalizadas.

2.4 *Cloudinary*

O *Cloudinary* é um serviço de nuvem que tem como objetivo gerenciar as imagens que forem carregadas na aplicação (CLOUDINARY, 2021).

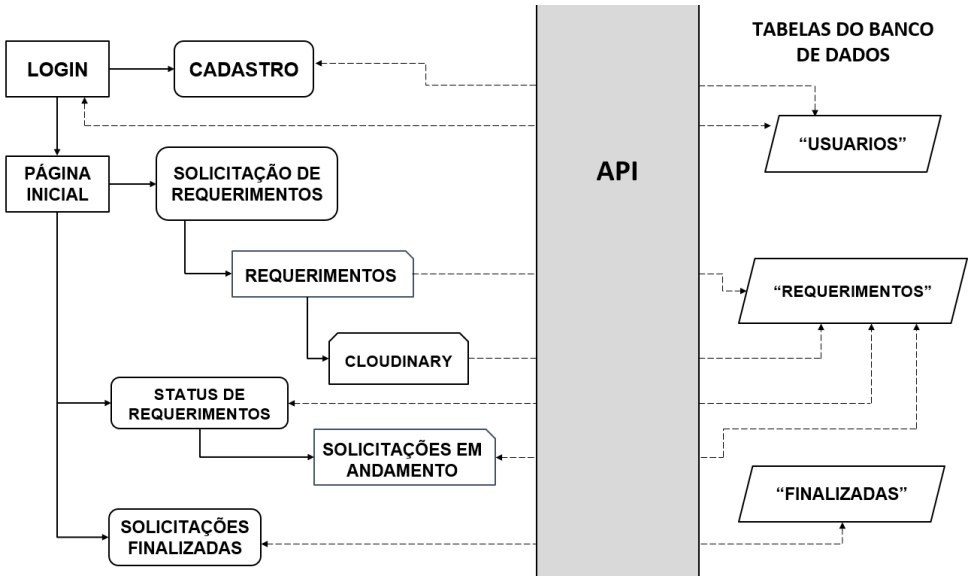
Para não ocorrer um sobrecarregamento do banco de dados do protótipo, foi feita a utilização do *Cloudinary* para armazenar os arquivos anexados pelos estudantes, como: comprovante de pagamento, documentação, em que ao invés de enviar os arquivos para o banco é enviado apenas o caminho destes.

No desenvolvimento do protótipo foi utilizada a versão gratuita do *Cloudinary,* já que os testes realizados no processo de criação da aplicação não exigiam uma grande quantidade de armazenamento na nuvem. Porém, essa ferramenta possui versões pagas com maiores capacidades para armazenar as documentações necessárias (CLOUDINARY, 2021).

3 RESULTADOS

Esta seção apresenta o aplicativo protótipo desenvolvido. Na Figura 1 pode ser observado o fluxograma funcional do sistema.

Figura 1 – Fluxograma funcional do sistema.



Fonte: Autoria própria.

Para o desenvolvimento do protótipo fez-se necessário a criação de três tabelas no banco de dados:

- tabela “usuarios”, para guardar informações de cadastro do usuário;
- tabela “requerimentos”, para armazenar o que foi preenchido na solicitação de requerimentos.
- tabela “finalizadas”, responsável por enviar seus dados para a API, que, por sua vez, encaminha para o protótipo (a título de simulações, seus dados foram preenchidos manualmente).

As tabelas “requerimentos” e “finalizadas” possuem uma coluna chamada “usuarioId”, com o objetivo de fazer conexão com a tabela “usuarios”.

No primeiro acesso, é necessário fazer o cadastro, em que o usuário deve informar seu primeiro nome, sobrenome, curso, telefone, *e-mail*, senha, confirmação de senha e CPF, como ilustra a Figura 2(b), evitando o preenchimento desses dados sempre que precisar fazer a solicitação de um requerimento. Cada estudante só poderá fazer o cadastro uma vez, pois o sistema não permite CPFs repetidos.

Após o aluno realizar corretamente o cadastro, suas informações são enviadas para a API, que encaminha para a tabela “usuarios”, como pode ser visto na Figura 2(d). Além dos dados de cadastro, é criado de forma automática um *id* (número de identificação) para cada estudante, de forma a facilitar o gerenciamento.

No desenvolvimento da aplicação foi utilizado o *bcrypt*. Com ele é gerado o *hash*, que será a senha cadastrada pelo usuário juntamente com um *salt* (uma sequência de caracteres que fornece aleatoriedade ao resultado da senha criptografada, tornando-a mais complexa). Assim, quando a API encaminha os dados de cadastro para o banco, a senha é criptografada, enviando para a tabela “usuarios” o *hash*. Com isso, ocorre o aumento da segurança das contas dos alunos. Na Figura 2(d) é possível verificar as senhas criptografadas.

Para ser realizado o *login* (ver Figura 2(a)), o aplicativo envia para a API o *e-mail* e senha informados pelo aluno, que são checados na tabela “usuarios” do banco para conceder, ou não, o acesso ao aplicativo. Efetuando o *login*, a aplicação salva as informações do usuário no *AsyncStorage*, um componente do *React Native* que armazena dados de forma local. Então, o usuário será redirecionado para a página inicial do protótipo (ver Figura 2(c)).

Sempre que o protótipo se comunica com a API para encaminhar ou buscar elementos do banco de dados, ele procura no *AsyncStorage* o *id* do aluno, sendo possível, assim, identificar qual usuário está *logado* no momento naquele dispositivo, repassando essa informação para o *back-end* (exceto no momento de busca do *status* de requerimento, em que é enviado para a API o número de protocolo específico, que foi digitado no aplicativo pelo estudante, para ser procurado nas tabelas “requerimentos” e “finalizadas” do banco). Essa busca no armazenamento local também foi feita para identificar o nome do estudante e colocar na página inicial da aplicação, como pode ser visto na Figura 2(c). Quando o *logout* é feito, esses dados são apagados do *AsyncStorage*.

Na guia inferior da página inicial, o usuário tem a opção de verificar seu perfil, em que será mostrado os dados que ele preencheu em seu cadastro. Nessa tela, o aluno tem a possibilidade de editar algumas de suas informações, tais como número de telefone e *e-mail*.

Figura 2 – Telas de *login*, cadastro e inicial do protótipo.

(a) Tela de *login.*

(b) Tela de cadastro.

(c) Tela inicial.

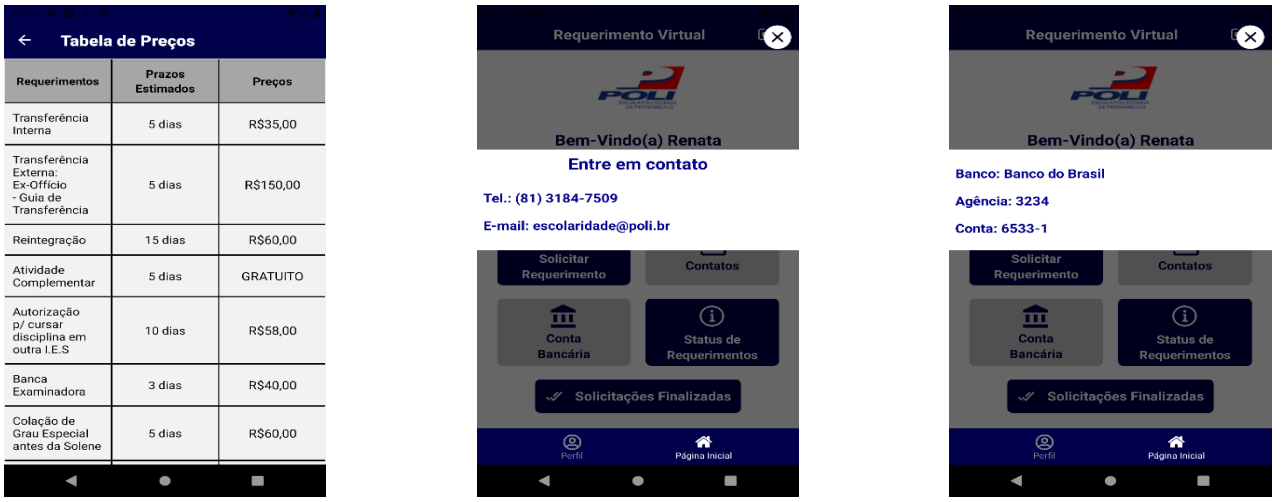
id	nome	sobrenome	cpf	curso	telefone	email	senha
7	Renata	Machado de Miranda	096.979.474-65	Engenharia Elétrica de Telecomunicações	(81) 99711-1960	rmm@poli.br	$2b$10$.Jn3FIC.OWIGnnOZ2eFzA.AVMgbx4TvBfo.WXC/GHI5...
8	Lucas	Abismael da Rocha Soares	125.784.050-93	Engenharia Elétrica de Telecomunicações	(81) 99785-6636	lars@poli.br	$2b$10$d6UXq.yF0JC2kOEfnmKXlekjBS6780kCp1RYKCnzh3l...

(d) Tabela de usuários do banco de dados.

Fonte: Autoria própria.

Na tela inicial, o usuário poderá verificar a tabela de preços (ver Figura 3(a)), que também apresenta os prazos de entrega para cada requerimento. Além disso, pode-se conferir os contatos da escolaridade (ver Figura 3(b)) e a conta bancária em que o aluno precisa depositar o valor do requerimento solicitado (ver Figura 3(c)).

Figura 3 – Telas de informações.



(a) Tabela de preços. (b) Contatos. (c) Conta bancária.

Fonte: Autoria própria.

No botão “Solicitar Requerimentos” (ver Figura 2(c)), o estudante será redirecionado para uma tela que mostrará as opções de requerimentos, como ilustra a Figura 4(a), sendo elas: requerimentos gerais, segunda chamada de exercício escolar, colação de grau, banca examinadora, autorização para cursar disciplina em outra I.E.S e programa e ementa de disciplina. Todas essas opções possuem um botão que leva diretamente para a tela de tabela de preços, para melhorar a experiência do usuário. Os requerimentos gerais (Figura 4(b)) são, em sua maioria, os requerimentos que necessitam apenas do comprovante de pagamento para serem solicitados (no caso da solicitação de Atividade Complementar, por ser um requerimento gratuito, não haverá comprovante de pagamento). Existem algumas exceções que exigem o preenchimento de um formulário no protocolo, porém, como esse formulário não é disponibilizado neste protótipo, foi colocada uma observação avisando exatamente quais são esses requerimentos que apresentam essa exceção. Ainda na seção de “requerimentos gerais”, é encontrado um botão de acesso ao calendário acadêmico, que leva o aluno diretamente para a página virtual da universidade.

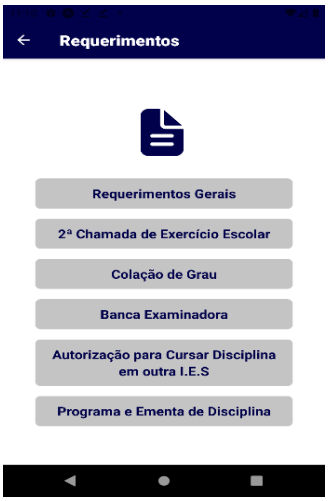
As outras opções de requerimentos foram colocadas à parte por necessitarem de dados e arquivos extras, quando comparados com os requerimentos gerais. A aplicação foi programada dessa forma para se obter uma melhor organização. Na colação de grau, por exemplo, o aluno precisará escolher entre colação de grau comum (só precisando anexar sua cópia de documento) e colação de grau especial – antes da solene, em que ele deverá anexar a mais o comprovante de pagamento e uma justificativa (ver Figura 4(c)). Outra opção que exige requisitos diferenciados é a segunda chamada de exercício escolar, em que é obrigatório o preenchimento dos espaços para informar o motivo da falta, a data em que ocorreu a prova, o nome e a turma da disciplina, o nome do professor, selecionar qual o exercício escolar ele deseja pedir a segunda chamada e, como nos requerimentos gerais, deverá ser anexado um comprovante de pagamento (ver Figuras 4(d) e (e)). Se o aluno precisar, também é possível anexar um atestado. Na tela para solicitação de banca examinadora (ver Figura 4(f)), o usuário deverá preencher o nome e a turma da disciplina, o nome do professor, selecionar o exercício escolar e anexar o comprovante de pagamento. Em caso de dúvidas, pode-se consultar o manual do aluno, em que mostra a seção do guia do estudante com informações sobre a revisão de prova. Já na autorização para cursar disciplina em outra I.E.S (Instituição de Ensino Superior) o usuário precisa anexar o comprovante de pagamento e o programa da disciplina solicitada com carimbo e assinatura

da outra I.E.S. Por fim, tem-se programa e ementa de disciplina, em que, além de anexar o comprovante de pagamento, precisa informar a disciplina desejada.

O botão “enviar arquivos”, que pode ser encontrado em todas as telas de solicitação de requerimentos, tem a funcionalidade de encaminhar tudo que for anexado para o *Cloudinary*, serviço de nuvem que está sendo utilizado nesse projeto. Na Figura 3 é possível observar algumas das telas desses requerimentos.

Assim que o usuário pressiona o botão para encaminhar os anexos para o *Cloudinary*, um botão de confirmar é exibido (ver Figura 5(a)), sua função é enviar as informações da solicitação para a API que, imediatamente, as envia para o banco de dados (ver Figura 5(c)). No caso dos anexos, só serão enviados seus caminhos, para que o banco não fique sobrecarregado. Após pressioná-lo, uma alerta de confirmação aparece na tela (ver Figura 5(b)), para indicar que a solicitação foi efetuada com sucesso e informar ao aluno um número de protocolo.

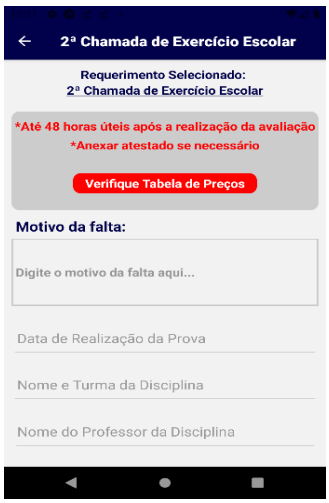
Figura 4 – Telas de solicitação de requerimentos



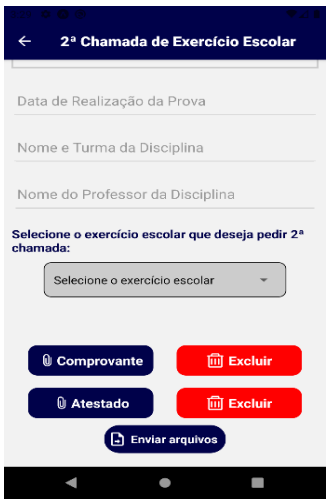
(a) Opções de requerimentos.

(b) Requerimentos gerais.

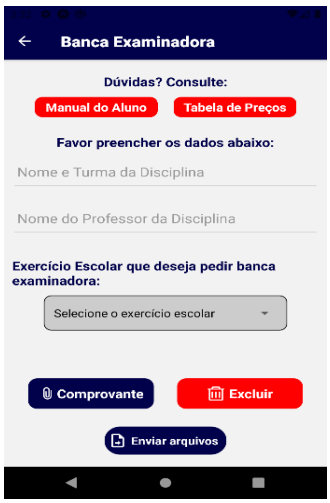
(c) Colação de grau.



(d) 2ª chamada tela 1.



(e) 2ª chamada tela 2.

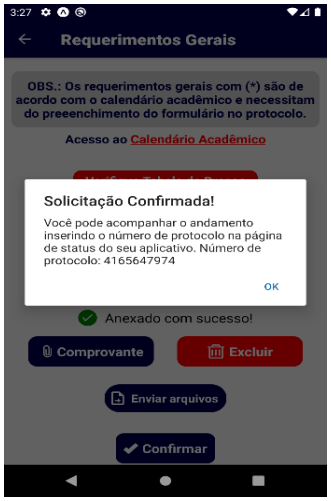


(f) Banca examinadora.

Fonte: Autoria própria.

Figura 5 – Solicitação de requerimento realizada.

(a) Requerimentos gerais.



(b) Tela de confirmação.

id	requerimento	protocolo	status	disciplina	professor	exercicio	motivoFalta	dataProva	comprovante	atestado	documento	justificativa	programa	usuariold
31	Comprovante de Matrícula - com grade horária	5100117257	está pronto para retirada	NULL	NULL	NULL	NULL	NULL	http://res.cloudinary.com/projeto-tcc/image/upload...	NULL	NULL	NULL	NULL	7
32	Comprovante de Vínculo	7263583667	está em andamento	NULL	NULL	NULL	NULL	NULL	http://res.cloudinary.com/projeto-tcc/image/upload...	NULL	NULL	NULL	NULL	7
33	Histórico Escolar - Graduando	4165647974	está em análise	NULL	NULL	NULL	NULL	NULL	http://res.cloudinary.com/projeto-tcc/image/upload...	NULL	NULL	NULL	NULL	7

(c) Tabela de requerimentos do banco de dados.

Fonte: Autoria própria.

Como pode ser observado na Figura 5(c), a tabela “requerimentos” possui uma coluna “*status*”, em que é preenchida automaticamente com uma mensagem inicial (“está em análise”) assim que o usuário realiza uma solicitação de requerimento. Essas mensagens foram alteradas manualmente na tabela a título de simulações.

No protótipo foi criada a tela “*Status* de Requerimentos” (ver Figura 7(a)), em que o aluno poderá se informar sobre a situação de sua solicitação. Para isso, basta digitar o número de protocolo que foi disponibilizado no espaço determinado.

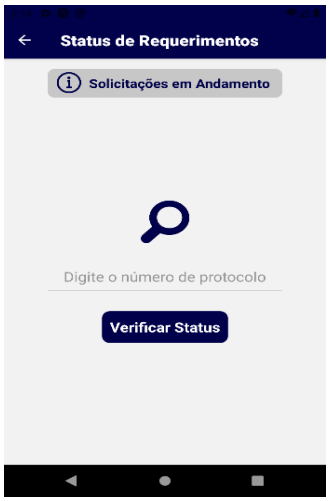
Para se obter o *status* da solicitação, o protótipo se comunica com a API, que vai buscar na tabela “requerimentos” do banco de dados o número de protocolo informado pelo estudante, caso ela não encontre nenhuma resposta, a API procura na tabela “finalizadas”. Desse modo, o usuário pode encontrar diferentes mensagens (ver Figura 6), como por exemplo, a informação de que o requerimento ainda está em análise, que é a primeira mensagem após a solicitação ser confirmada e indica que a escolaridade está verificando se todos os dados foram enviados corretamente; a informação de que o requerimento ainda está em andamento, o que significa que não ocorreu nenhum erro no envio dos arquivos da solicitação. Também tem-se o aviso “pronto para retirada”, a esse ponto o aluno já deve ir buscar seu requerimento solicitado, e o aviso “solicitação não encontrada”, caso o aluno tenha digitado o número do protocolo errado.

Figura 6 – Mensagens de *status* de requerimentos.

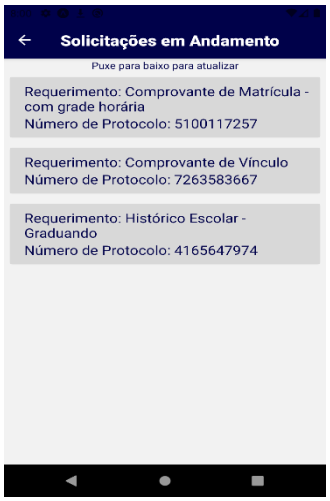
Fonte: Autoria própria.

Com o objetivo de evitar que o aluno perca o número de protocolo e fique impossibilitado de acompanhar suas solicitações, foi criada uma página de informações de solicitações, podendo ser acessada pelo botão “Solicitações em Andamento” que pode ser visto na Figura 7(a). A tela de “Solicitações em Andamento” apresenta o tipo de requerimento e seu número de protocolo, como pode ser observado na Figura 7(b). E para o usuário ter um melhor controle de seus pedidos concluídos, o protótipo possui uma área onde são mostradas todas as solicitações finalizadas: é possível ter acesso ao nome do requerimento, o número de protocolo, a data em que foi feita a solicitação e a data que ela foi finalizada, como pode ser visto na Figura 7(c). Para isso, o aplicativo se comunica com a API que busca essas informações da tabela “finalizadas” do banco de dados (ver Figura 7(d)).

Figura 7 – Informações de solicitações e solicitações finalizadas.



(a) Status de requerimentos.



(b) Informações de solicitações.

(c) Solicitações finalizadas.

id	requerimento	protocolo	datalnicio	dataTermino	usuariold
1	Comprovante de Vínculo	8562036521	26/03/2021	29/03/2021	7
2	Atividade Complementar	6365987420	01/04/2021	06/04/2021	8

(d) Tabela “finalizadas” do banco de dados.

Fonte: Autoria própria.

4 CONCLUSÕES

O procedimento de solicitação de requerimentos discentes, na Escola Politécnica de Pernambuco - POLI, Universidade de Pernambuco – UPE, exige a execução de muitas etapas, deixando o processo mais complexo e desgastante do que o necessário.

Visando contribuir para a modernização e melhoria dos serviços fornecidos pela universidade, o alvo deste trabalho é realizar o desenvolvimento de um aplicativo protótipo para *smartphones*, em que o aluno, além de solicitar seus requerimentos, poderá acompanhar o andamento de suas solicitações, eliminando etapas presenciais. Dessa forma, espera-se contribuir no processo de solicitação de requerimentos através da sua simplificação com a implementação do requerimento virtual, resultando em uma solução moderna e prática para os discentes e os setores administrativos da universidade.

Outro ponto importante a citar seria a diminuição da quantidade de papéis utilizados que essa aplicação pode gerar, tendo em vista que, atualmente, é necessário o aluno entregar um termo de requerimento único preenchido, como também, em alguns casos, cópia de alguns documentos. Com a ferramenta proposta, tudo isso será feito virtualmente, diminuindo significativamente o volume de documentos impressos no setor responsável pelos requerimentos, contribuindo assim com a sustentabilidade do planeta.

Como trabalhos futuros, pretende-se implementar uma conexão entre o aplicativo protótipo e um servidor criado para a universidade gerenciar toda a demanda de requerimentos discentes. Outro ponto que poderia ser trabalhado é a viabilização do *download* e o *upload*, no próprio aplicativo, do formulário que deve ser preenchido para a solicitação dos seguintes requerimentos: dispensa de disciplina, transferência interna, reintegração e transferência externa: *ex officio*. Desenvolver, também, uma forma de realizar o pagamento dos requerimentos no próprio aplicativo, facilitando ainda mais o processo para os estudantes.

REFERÊNCIAS

CLOUDINARY. **Image and Video Upload, Storage, Optimization and CDN**. Disponível em: https://cloudinary.com/. Acesso em 15 mar. 2021

COUTINHO, Gustavo Leuzinger. **A Era dos Smartphones: Um estudo exploratório sobre o uso dos Smartphones no Brasil.** 2014. Monografia - Curso de Publicidade e Propaganda. Universidade de Brasília, Brasília, 2014.

DVORSKI, Dalibor D. **INSTALLING, CONFIGURING, AND DEVELOPING WITH XAMPP**. Skills Canada, 2007, Ontario. Disponível em: http://dalibor.dvorski.net/downloads/docs/installingconfiguringdevelopingwithxampp.pdf. Acesso em 2 fev. 2021.

ESCUDELARIO, Bruna; PINHO, Diego. **React Native: Desenvolvimento de aplicativos mobile com React**. São Paulo: Editora Casa do Código. 2020.

EXPO. **Introduction to Expo.** Disponível em: https://docs.expo.io/. Acesso em 5 fev. 2021.

JACOBSON, Daniel; BRIAL, Greg; WOODS, Dan. **APIs: A Strategy Guide: Creating Channels with Application Programming Interfaces**. O'Reilly Media. 2011.

NODE.JS. **About Node.js**. Disponível em: http://nodejs.org/about/. Acesso em 23 abr. 2021.

ORACLE. **MySQL Database Service.** Disponível em: https://www.oracle.com/mysql/ Acesso em 15 mar. 2021.

PAES, Denyse.; SOUZA, Maria.; COSTA, Rosane. **Redes Ubíquas para que? O Uso de Aplicativos Digitais no Cenário Acadêmico.** In*:* XX Seminário Nacional de Bibliotecas, 2018, Salvador.

PHPMYADMIN. **Bringing MySQL to the web**. Disponível em: https://www.phpmyadmin.net/. Acesso em 12 mar. 2021.

REACT NATIVE. **React Native Docs**. Disponível em: https://reactnative.dev/docs/getting-started. Acesso em 5 fev. 2021.

ROMERO, B. A.; SAAD, E. F.; BASTOS, G. B. **React-native, avaliação de performance comparando com código java nativo com listview.** JORNAL DE ENGENHARIA, TECNOLOGIA E MEIO AMBIENTE-JETMA, v. 2, n. 2, p. 23, 2018.

RUBENS, João. **Primeiros passos com Node.js**. São Paulo: Editora Casa do Código, 2017.

SEQUELIZE ORM. **Sequelize**. Disponível em: https://sequelize.org/. Acesso em: 10 mar. 2021.

SILVA, D.; SOBRAL, L. **Um estudo em larga escala sobre a estrutura do código-fonte de pacotes javascript**. 2017. Monografia - Curso de Sistemas de Informação. Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

TAROUCO, Fabrício. **A Metrópole Comunicacional e a Popularização dos Apps para Dispositivos Móveis.** In: V Seminário Internacional de Pesquisa em Comunicação (Sipecom), 2013, Santa Maria.

PROTOTYPE APPLICATION FOR REQUEST OF STUDENT REQUIREMENTS

Abstract: *This project aims to develop a prototype application for smartphones with React Native, which the student will be able to request requirements from the Polytechnic School of Pernambuco – POLI, University of Pernambuco – UPE, for example: complementary activity, make-up test, school records. The implementation of this application, in addition to modernizing, will simplify and optimize the current process by reducing the amount of printed documents used and eliminating presencials steps, a fator that has proved to be very importante in the current times of social isolation, because of COVID-19. In addition, by making all requests virtual, the student will be able to monitor the progress of his requirements, making it possible to know when the request was completed. For this purpose, a MySQL database was created to store both the students' registration information, as well as the requests made.*

Keywords: *Prototype Application, React Native, Requirements, MySQL.*