UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO

Plano de Testes

Projeto Corre Comigo!

Equipe 2

1. Identificação

Nome do Projeto: Corre Comigo!

Equipe: Alysson Rogério de Oliveira
Felipe Manfio Barbosa
Hiago de Franco Moreira
João Vitor Granzotti Machado
Leonardo Sensiate
Thales de Lima Kobosighawa
Vinícius Nakasone Dilda

Data de Criação do Documento: 10/05/2019

2. Introdução

A descrição dos aspectos a serem testados diz respeito às funcionalidades vitais do aplicativo, ou seja, àquelas que deverão obrigatoriamente estar presentes no MVP funcional apresentado ao final do período de projeto.

Funcionalidades opcionais, que forem adicionadas posteriormente, seguirão os mesmos padrões de teste aplicados às demais funcionalidades já implementadas.

Aspectos a serem testados:

Tendo extraído todos os requisitos da aplicação, estabelecemos dentro destes requisitos quais aspectos dessas funcionalidades deverão ser testados a

fim de validar a implementação. Os aspectos que estão cobertos pelo processo de teste estabelecido pela equipe são listados abaixo:

1 - Cadastro de usuário:

- Verificação de email válido;
- Verificação de senha conforme as especificações (tamanho e restrições de caracteres);
- Verificação de nome de usuário já existente;
- Verificação de inserção correta do cadastro no banco de dados.

2 - Login:

- Pelo cadastro do aplicativo;
- Verificação de senha incorreta;
- Verificação de email cadastrado;
- Pelo *Facebook.*

3 - *Feed* de eventos:

- Correspondência dos dados mostrados com o evento do *card*;
- Correspondência dos eventos exibidos com as preferências do usuário cadastradas;
- Redirecionamento para a tela com descrição completa do evento.

4 - Inserção de eventos:

- Verificação se o evento foi inserido com sucesso no banco de dados;
- Verificação se os campos de inserção obedecem às regras especificadas, como tamanho (qtde. de caracteres) e formato dos dados;
- Teste dos botões confirmar e cancelar;
- Teste para verificar se todos os campos foram preenchidos.

5 - Sistema de pesquisa de eventos:

- Filtragem de acordo com o que o usuário pesquisar;

- Apresentação dos resultados da pesquisa;
- Redirecionamento para tela do evento escolhido.

6 - Visualização das informações do evento selecionado no *feed*:

- Apresentação de todas as informações referentes ao evento, em uma tela separada;
- Redirecionamento para a página do evento.

7 - Sistema de Notificações:

- Garantia de que o usuário receba uma notificação com o evento mais recentemente adicionado que seja compatível com seus objetivos;
- O usuário pode aceitar visualizar o evento ou descartar a notificação.

8 - Edição de perfil:

- Recuperação e apresentação das informações de perfil do usuário;
- Possibilidade de alteração dos dados do perfil e a consequente mudança das informações no banco de dados;

9 - Recuperação cadastro em caso do usuário perder a senha:

- Recuperar a senha cadastrada no banco e enviar via email para o usuário;

10 - Sair da conta:

- O usuário pode se desautenticar do aplicativo a qualquer momento;
- O aplicativo deve retornar à tela de Login.

Aspectos que não serão testados:

Devido ao grande número de funcionalidades do aplicativo, aos diferentes cenários de utilização que este apresenta e também ao cronograma do projeto, existem algumas características que não serão diretamente testadas pela equipe.

Vale ressaltar que, durante o desenvolvimento, mesmo que alguns aspectos não sejam testados, características como qualidade e bom funcionamento serão uma preocupação constante da equipe, de modo que os aspectos não testados não venham a atrapalhar a utilização do aplicativo por parte dos usuários. Sendo assim, entre os aspectos que não serão testados durante o desenvolvimento, temos:

- Desempenho do aplicativo no sistema operacional iOS;
- Navegação no aplicativo sem a conexão com uma rede de internet;
- Tratamento de links quebrados ou expirados fornecidos pelos organizadores;
- Problemas referentes ao banco de dados - queda do servidor, por exemplo;
- Aspectos de acessibilidade do aplicativo - suporte à usuários com algum tipo de deficiência ou limitação;

Critérios para aceitação:

Para que o sistema seja aprovado, cada uma das funcionalidades deve ter passado pelo processo de testes individuais (durante seu desenvolvimento) e um teste integrado, ou seja, de todas as funcionalidades operando junto na aplicação “final”, de modo a verificar se o comportamento esperado foi atingido.

Além disso, o aplicativo deve apresentar o comportamento esperado para as diversas histórias de usuário, determinadas juntamente com o cliente.

Como processo final para aceitação do produto, haverá uma apresentação ao cliente, que cobrirá todas as funcionalidades apresentadas, a fim de demonstrar o que foi desenvolvido e o funcionamento do produto. O aplicativo só será dado como aprovado e finalizado caso o cliente esteja satisfeito com o que

lhe for apresentado, indicando que todos os requisitos estabelecidos por ele tenham sido atendidos.

3. Planejamento para realização dos testes

Cronograma de atividades:

Este cronograma de atividades contém a descrição dos passos que serão realizados na condução dos testes. Vale lembrar que o desenvolvimento deste aplicativo segue o modelo ágil SCRUM, e será composto por um total de 5 Sprints de desenvolvimento. Para melhor visualização, o cronograma referente a cada sprint está disposto em uma tabela. As tabelas com os cronogramas de testes de cada sprint estão dispostas abaixo:

Sprint 1		
Atividade	Data Início	Data Fim
Planejar Testes	14/05	15/05
Executar Testes	15/05	19/05
Avaliar Testes	20/05	21/05
Documentar Testes	15/05	21/05

Sprint 2		
Atividade	Data Início	Data Fim
Planejar Testes	30/05	31/05
Executar Testes	01/06	03/06
Avaliar Testes	03/06	04/06
Documentar Testes	01/06	04/06

Sprint 3		
Atividade	Data Início	Data Fim
Planejar Testes	08/06	09/06
Executar Testes	09/06	10/06
Avaliar Testes	10/06	11/06
Documentar Testes	09/06	11/06

Sprint 4		
Atividade	Data Início	Data Fim
Planejar Testes	13/06	15/05
Executar Testes	15/06	17/06
Avaliar Testes	17/06	18/06
Documentar Testes	15/06	18/06

Sprint 5		
Atividade	Data Início	Data Fim
Planejar Testes	21/06	22/06
Executar Testes	23/06	25/05
Avaliar Testes	26/06	26/06
Documentar Testes	23/06	26/06

Responsáveis pelos testes:

A tabela a seguir descreve as áreas mais gerais de teste, com uma breve descrição das mesmas, assim como os responsáveis pela execução de cada uma de tais áreas:

Teste	Descrição	Responsáveis
Integridade do Banco de Dados	Garantir que os métodos de gravação, modificação e exclusão de dados no banco funcionam de forma correta e sem corrupção dos mesmos.	João Vitor, Leonardo, Vinícius
Funcionalidades	Verificar se o comportamento de cada funcionalidade segue os casos de teste definidos, apresentando as saídas esperadas e possíveis notificações de erros, de modo a cumprir com os requisitos levantados.	Desenvolvedores da funcionalidade
Interface do Usuário	Verificar se as características de cada tela, como menus, tamanho e posição dos itens proporcionam uma interface amigável com o usuário. Além disso, verificar se as transições entre telas e botões de acesso seguem uma lógica bem definida de modo a proporcionar uma boa experiência ao usuário	Felipe, Leonardo e Vinícius

Segurança e Controle de Acesso	Verificar o acesso de um ator apenas às telas, funções e dados para os quais seu tipo de usuário (normal ou admin) tem permissão.	Alysson, Hiago, Thales

4. Projeto de casos de teste

Casos de uso considerados:

Os casos de uso considerados para os testes são baseados em histórias de usuários e "fluxos de navegação". Estes, por sua vez, foram elaborados juntamente com o cliente, para ilustrar como a aplicação deve funcionar durante a interação dos usuários.

Com base nesses fluxos esperados, a equipe irá desenvolver novas situações e consequentes novos casos de uso para auxiliar tanto no desenvolvimento quanto nos testes das funcionalidades implementadas.

Além disso, durante os sprints, o PO irá apresentar o protótipo do aplicativo desenvolvido até o momento para o cliente, possibilitando a criação de novas atividades de testes e a extração de mais detalhes técnicos, a fim de tentar cobrir os requisitos do aplicativo da melhor maneira possível, de maneira funcional.

Geração de casos de teste:

Utilizando a abordagem de teste funcional, podemos definir os critérios de teste baseados em classes de equivalência e análise do valor limite, apresentados nas tabelas a seguir.

Classes de equivalência:

Caso de teste	**Variáveis de entrada**	**Classes de equivalência válidas**	**Classes de equivalência inválidas**
Verificação de e-mail válido	Comprimento antes do caractere @ (t)	1<= t <= 50	t < 1 e t > 50
	Contém caractere @ (e)	Sim	Não
	Contém apenas letras, dígitos ou “_” (c)	Sim	Não
	Finaliza com “.com” ou “.br” (f)	Sim	Não
Verificação de senha válida	Comprimento (t)	6 <= t <= 15	t < 6 e t > 15
	Inicia com uma letra (i)	Sim	Não
	Contém apenas letras ou dígitos (c)	Sim	Não
Verificação dados de usuários e eventos	Dados de texto	1 <= t <= 300	0 >= t >= 300
	Dados com máscara de entrada	Formato máscara preenchido	Formato da máscara incompleto
	Dados com “checkbox”	Opções marcadas	Nenhuma marcação no checkbox

Análise do valor limite:

Caso de teste	**Entrada**				**Saída**
Verificação de e-mail válido	**t**	**e**	**c**	**f**	
	“” (tamanho 0)	“” (não contém @)	“” (caracteres diferentes de dígito, letra ou “_”)	“” (não finaliza com “.com” ou “.br”)	inválido
		a (não contém @)		a (não finaliza com “.com” ou “.br”)	inválido
	a@gmail.com (tamanho 1)	a@gmail.com(contém @)	a@gmail.com(contém apenas dígitos, letras ou “_”)	a@gmail.com (finaliza com “.com” ou “.br”)	válido
			abc#@gmail.com (caracteres diferentes de dígito, letra ou “_”)		inválido
				eu_atleta@algo.email (não finaliza com “.com” ou “.br”)	inválido

Caso de teste	**Entrada**			**Saída**
Verificação de senha válida	**t**	**i**	**c**	
	“” (tamanho 0)	“” (não inicia com letra)	“” (caracteres diferentes de dígito ou letra)	inválido
	a (tamanho 1)	a (inicia com letra)	a (contém apenas dígitos ou letras)	válido
	a123 (tamanho entre 1 e 15)	a123 (inicia com letra)	a123 (contém apenas dígitos ou letras)	válido
	abcdefgh1234567 (tamanho 15)	abcdefgh1234567 (inicia com letra)	abcdefgh1234567 (contém apenas dígitos ou letras)	válido
		1 (não inicia com letra)		inválido
			a@ (Contém caracteres diferentes de letras ou dígitos)	inválido

	abcdefgh12345678 (tamanho maior que 15)			inválido

5. Conclusão

Recursos utilizados:

O principal recurso utilizado para a realização dos testes será as histórias do usuário, baseadas no fluxo de navegação do aplicativo. Essa abordagem se dá por meio da determinação de cenários ideias, conhecidos como "*happy- flows*", e os cenários não ideias, ou "*unhappy flows*", que nos dão uma diretriz de como o usuário se comporta ao utilizar o aplicativo. Alguns desses fluxos foram determinados em conjunto com o cliente e outros criados pela equipe, a fim de explorar os pontos fortes e fracos da implementação de cada uma das funcionalidades.

Em posse de tais histórias, serão feitos os testes diretamente no aplicativo, com a visão de um usuário real, sendo que eventuais erros e comportamentos inesperados serão corrigidos no código à medida que forem identificados.

Além disso, os protótipos do aplicativo serão disponibilizados aos membros da equipe que não desenvolveram a funcionalidade a ser testada, para que estes interajam com a aplicação e encontrem eventuais erros que não foram identificados pelos desenvolvedores.

Quanto às ferramentas utilizadas para os testes, serão utilizados:

- O console do *Firebase*, para testar a inserção e presença dos dados no banco de dados do aplicativo;

- NPM (*Node Package Manager*), que é um repositório online para publicação de projetos de código aberto para o Node.js e também funciona como um utilitário de linha de comando que interage com este repositório online, auxiliando na instalação de pacotes, gerenciamento de versão e gerenciamento de dependências;

- O aplicativo Expo, que é uma ferramenta de desenvolvimento para criar experiências com gestos e gráficos interativos, usando JavaScript e React Native. Esta ferramenta permite visualizar a versão do aplicativo em tempo real, à medida que este está sendo desenvolvido;

O uso de uma ferramenta automática de testes não foi avaliado como necessário, mas pode ser incorporado durante o processo caso a equipe constate a necessidade.

Sendo assim, com estas ferramentas, somadas ao esforço dos desenvolvedores/testadores, e também à avaliação do cliente, espera- se testar e avaliar todas as funcionalidades do aplicativo a fim de garantir seu funcionamento correto, dentro dos cenários esperados e de modo a satisfazer todos os requisitos levantados.