**Plano de Teste -**

# Introdução

Esse Plano de teste corresponde aos testes realizados no funcionamento do Node SYOS.

# Arquitetura

O framework utilizado para a implementação do back-end do Node é o C++, que possui uma arquitetura orientada a objetos.

Para o armazenamento, consulta e alteração de dados da aplicação, será usada uma API que disponibiliza um conjunto de serviços REST. Funciona como na imagem, em que o cliente (aplicação) envia uma requisição à uma URL (endpoint) para acessar os serviços da API, e tem como retorno um objeto na forma de um JSON.

# Funcionalidades

|  |  |  |  |
| --- | --- | --- | --- |
| **Funcionalidades** | **Comportamento Esperado** | **Verificações (Ação)** | **Critérios de Aceite** |
| Encontrar os beacons | Ao comissionar o Node em uma loja via aplicativo “Instalação de mercado SYOS” ele deverá encontrar os beacons próximo a ele em um raio de 20 metros.  Se ele encontrar os beacons, ele piscará o led verde (1 vez, para X beacons, 2 vezes para X beacons e 3 vezes para X beacons). | Caso, ele esteja procurando os beacons ele deverá piscar vermelho X vezes.  Assim, que o dispositivo encontrar os beacons, ele deverá piscar o led verde, a quantidade de vezes será de acordo com a quantidade de beacons. |  |
| Login | Ao digitar seu usuário e senha corretamente o usuário irá logar na plataforma.  Ao tentar se logar e falhar 3 vezes consecutivas o usuário terá que esperar 15 minutos para tentar logar novamente.  O sistema deve aceitar usuários que já está logado em outro device se logar novamente | * Login no Sistema * Usuario Invalido * Campo Obrigatório | * O Nome e a senha devem ser obrigatórios * O sistema deve funcionar em todos os navegadores. |
| Lista de Fotos | Ao entrar na tela inicial o usuário irá ver uma lista com todas as fotos.  Deve ser exibido um ícone de coração no canto direito.  Deve o ícone de coração deve exibir uma animação e trocar para a cor vermelha.  Deve ser exibido uma caixa de texto para o usuário escrever o comentário. | * Número de fotos. * Número de curtidas e comentários correspondentes. * Animação do icone.gk |  |
| Comentario em Fotos | Ao clicar nos detalhes das fotos o usuário poderá fazer um comentário.  Deve ser exibido uma caixa de texto para o usuário escrever o comentário.  Deve ser exibido o texto “Comentar” na barra de texto para orientar o usuário.  Deve ser exibido o nome do usuário e o comentário  Deve ter limite de 140 por comentário | * Limite de 140 caracteres. * Nome do usuario. * Comentários anteriores. |  |
| Detalhes da fotos | Serão exibidos os detalhes da foto: - Foto - Curtidas - Comentarios  - Opção de remoção da foto  Usuário deve conseguir curtir uma mensagem ao clicar no ícone de “coração”  O ícone de coração deve mudar de cor  Usuário que teve sua mensagem curtida deve receber uma notificação  Deve ser exibido uma caixa de texto para o Usuário escrever o comentário. | * Número correto de curtidas e comentários. * Opção de remoção de fotos. * Possibilidade de curtir e comentar nas fotos. |  |
| Remover foto | Ao clicar no item em excluir a foto deverá ser removida  Deve aparecer uma caixa de confirmação perguntando se usuário deseja mesmo excluir o stories  O usuário deve ter a opção de cancelar exclusão  O usuário deve confirmar a exclusão. | * Exclusão da foto * Acessar a foto após a remoção. * Mensagem de confirmação em caso positivo. * Mensagem de cancelamento. |  |
| Pesquisa | Ao digitar um texto na barra de pesquisa deverá aparecer as fotos correspondentes.  Devem ser exibidas sugestões ao clicar na caixa de texto e a medida que o usuário digita.  Deve ser exibida uma mensagem caso não haja fotos correspondentes a pesquisa do usuário. | * Pesquisa encontrou os resultados esperados * Pesquisa não encontrou resultados * Sugestões de Pesquisa. |  |

# Estratégia de Teste

* **Escopo de Testes**

O plano de testes abrange todas as funcionalidades descritas na tabela acima.

Serão executados testes em todos os níveis conforme a descrição abaixo.

Testes Unitários: o código terá uma cobertura de 60% de testes unitários, que são de responsabilidade do time de firmware.

Testes de Integração: Serão executados testes de integração em todos os endpoints, e esses testes serão de responsabilidade do time de firmware e quality assurance.

Testes Manuais: Todas as funcionalidades serão testadas manualmente pelo time de qualidade seguindo a documentação de Cenários de teste e destes TestPlan.

Versão Beta: Será lançada uma versão beta para 20 usuários pré-cadastrados antes do release.

* **Ambiente e Ferramentas**

Os testes serão feitos do ambiente de homologação, e contém as mesmas configurações do ambiente de produção com uma massa de dados gerada previamente pelo time de qualidade.

As seguintes ferramentas serão utilizadas no teste:

|  |  |  |
| --- | --- | --- |
| Ferramenta | Time | Descrição |
| POSTMAN | Qualidade | Ferramenta para realização de testes de API |
| Jasmine | Desenvolvimento | Framework utilizada para testes unitarios |
| Selenium | Qualidade | Ferramenta para testes end-to-end |

# Classificação de Bugs

Os Bugs serão classificados com as seguintes severidades:

|  |  |  |
| --- | --- | --- |
| ID | Nivel de Severidade | Descrição |
| 1 | Blocker | * Bug que bloqueia o teste de uma função ou feature causa crash na aplicação. * Botão não funciona impedindo o uso completo da funcionalidade. * Bloqueia release. |
| 2 | Critical | Feature funciona pobremente   * Input esperados causam crash ou efeitos indesejados * Input incomum causa efeitos irreversíveis |
| 3 | Major | Feature não atinge certos critérios de aceitação, mas sua funcionalidade em geral não é afetada   * Mensagem de erro ou sucesso não é exibida * Input incomum causar efeitos indesejados, mas contornáveis |
| 4 | Minor | Quase nenhum impacto na funcionalidade, mas ainda é um erro válido   * Erro ortográfico * Pequenos erros de UI |

# Definição de Pronto

Será considerada pronta as funcionalidades que passarem pelas verificações e testes descritas nestes TestPlan, não apresentarem bugs com a severidade acima de Minor, e passarem por uma validação de negócio de responsabilidade do time de produto.

Teste de regressão é uma técnica de teste a ser aplicada quando surgem novas versões mais recentes do software e garante que não surjam novos defeitos em componentes já analisados;

Teste de sanidade é o subconjunto do teste de regressão e também é realizado quando não temos tempo suficiente para fazer o teste mais elaborado. Ele tem um nível superficial e verifica se as funcionalidades mais críticas do sistema estão conforme o esperado;

Testes de limite consistem em testar os valores mínimo e máximo (ou primeiro e último valores) de uma partição. Essa técnica é geralmente usada para testar requisitos que exigem um intervalo de números (incluindo datas e horas);

Teste de transição de estados é utilizado para testar a capacidade que o software tem de entrar em e sair de estados definidos através de transições válidas e inválidas;

Testes não-funcionais tem como objetivo testar aspectos do software que não são associados a funcionalidades. Ex: escalabilidade, desempenho, segurança;

Testes de performance são um conjunto de testes que visam analisar o desempenho. Entre eles, temos:

Teste de carga, que tem como objetivo verificar o desempenho de um sistema quando ele é submetido a cargas variáveis de usuários ou transações.

Teste de capacidade, parecido com o de carga, tem como objetivo identificar os limites da aplicação, ou seja, quantos usuários simultâneos ou chamadas por segundo a aplicação é capaz de suportar dentro dos parâmetros de qualidade definidos.

Teste de stress, verifica a performance de um sistema quando é submetido a cargas que estão no limite ou acima do limite especificado inicialmente.

Teste de usabilidade, tem como objetivo observar usuários reais usando o software para descobrir problemas e pontos de melhorias.