Projeto: coleção de itens com um propósito/objetivo.

Itens: unidade de trabalho; o que precisa ser feito. Seriam as histórias tarefas, epicos, funcionalidades...

Componentes: (modulo, segmentação)

Painéis: quadro

Organização: Temos o kanban, que ajuda a dar visibilidade de como o trabalho está indo, dividido por etapas.

\*roadmap é diferente de cronograma

Backlog: estoque de requisições; “trabalho em atraso”

User story: descrição de uma parte da solução (quem usa, o que faz, pra que usa)

Épicos: (aulas, financeiro, mercado,

Criar projeto no Jira:

– Framework **scrum**:

Tipos de historia: épico, historia, bug, tarefa e subtarefa.

Organizar o trabalho em ciclos de desenvolvimento (sprints)

Ter relatórios de burnout de velocidade de time.

Usar template/framework.

Tipo de projeto: empresa ou equipe.

Nome do projeto: pode ser o nome do produto-> chave gerada (naõ muda).

Sprint ativo: trabalho do dia a dia do time (historias, tarefas...)

Relatórios apropriados: como grafico de burndown, burnup, relatorio de sprint, grafico de velocidade, diagrama de fluxo cumulativo...

- Framework **kanban**:

projeto contínuo; a gestão das sprints é diferente do scrum. Projeto mais simples.

Relatórios de gráfico continuo, gráfico de controle...

Itens: titulo, status, responsavel...

Componenttes: separar por módulos

Conectar projeto ao confluence pra fazer gestão da informação

Configurações se tiver direito de adm.

Jira Qualy Language?

Miro: criar telas

LOLZOLA

historia na backlog da sprint -> iniciar um quadro -> in progress

pontuação? controle de tempo?

tasks: caso de teste e execução de teste.

(fazer fluxo de trabalho) (onde? No jira mesmo? Pode ser kanban?)

(roteiro é bom pra ver o andamento -> parte com mes)

Configuração e:

- cadeia de ferramentas - build: add github

- app - extensão: add trial (ela usou zephyr scale)

voltar pro projeto - adm permitir que tds acessem em config - on/habilitar. (\*permissões no off: td permitido para todos)

voltar nos criterios de aceite e exportar os cenários pra ferramenta de teste

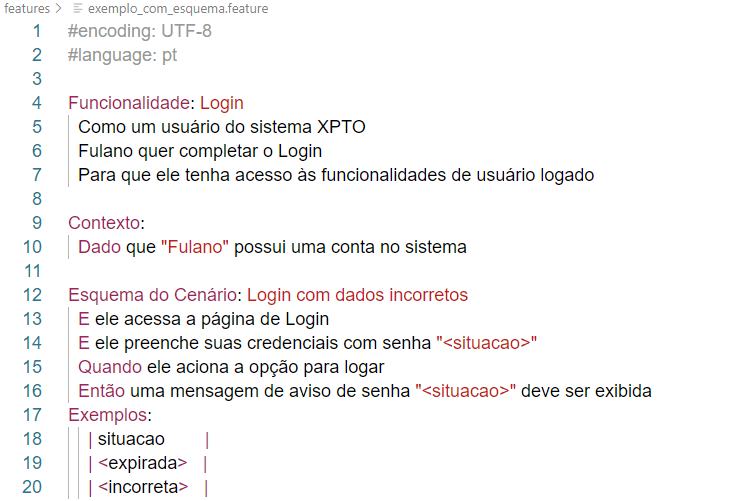
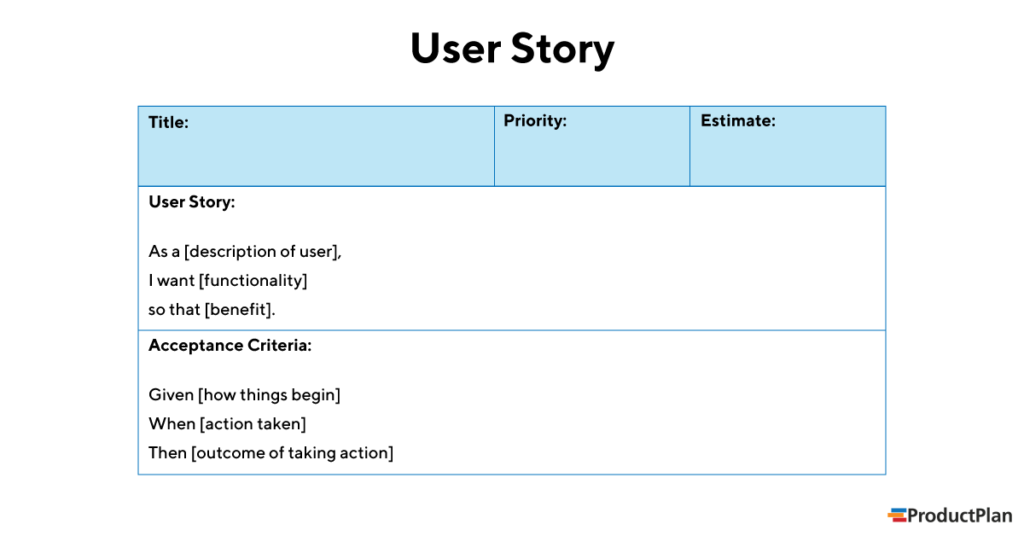
plano de teste no zephyr - testes funcionais - criar plano (nome status rótulos (sprint web), issues - criado plano associado a sprint 1 - criou pasta especifica pra caso de teste (cenário), ex: cliente sem cadastro criar conta (\*não especifica dados de entrada e saída) - add objetivo (validar info de login)-

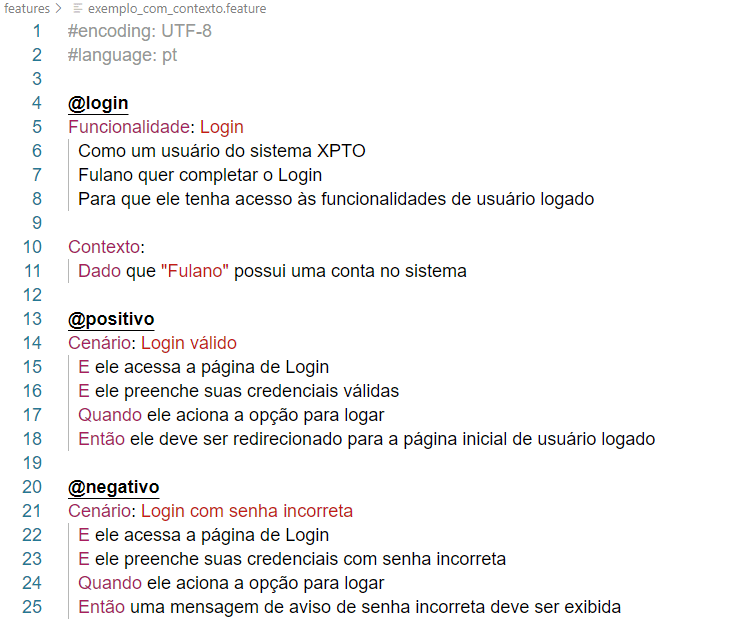
scrip de teste - exportar cenarios - usar linguagem gherkim script bdd -

usar confluence: criar caso de teste, editar (não terminou de explicar quando)

CASO: SEM CAD TENTA FAZER LOGIN

PASSOS



* 1. **INVEST:** Independente ( Independent); Negociável ( Negotiable); Valiosa (Valuable); Estimável (Estimable); Pequena (Small); Testável ( Testable).

**Critérios de aceite:** Confirmação se requisitos foram atendidos:

Em forma de sentenças como um checklist

1. ~~Entrega de valor: objetivo da funcionalidade~~
2. ~~Narrativa do usuário~~
   * 1. 3. Requisitos técnicos.
     2. a) Atores
     3. b) Interfaces/Fluxos

c)Dados

e. Regras de negócios

f. Ambiente

h. Critérios de aceite

### 3. Requisitos técnicos

No terceiro e último bloco, você vai descrever os requisitos técnicos. Para isso, é possível utilizar a técnica “7 dimensões do produto”, que percorre os seguintes aspectos:

* **1. Atores:**Administrador da Fazenda; Dono da Fazenda.
* **2. Interfaces:**

     – [Visuais de UX/UI](https://miro.com/app/board/o9J_kvjkmGg=/)

     – [Fluxos da jornada do usuário](https://miro.com/app/board/o9J_kvjkmGg=/)

     – Interface com as APIs do Climatempo.

* **3. Ação (história do usuário):**

     \*\*Eu como\*\* usuário administrador da fazenda \*\*quero\*\* visualizar a probabilidade de chuva para o dia seguinte, com umidade e volume \*\*para que\*\* eu possa planejar a irrigação da lavoura com a quantidade correta de água.

* **4. Dados:**

     – Login via Google, Facebook, Celular

     – Dados da previsão do tempo via fornecedor externo (Climatempo)

     – Deveremos taguear os eventos de uso do produto segundo o documento de mapa de eventos nesse link: <https://bit.ly/mapa_eventos_app>

* **5. Regras de negócios:**

     – Usuários com plano de assinatura suspenso não poderão ter acesso ao app.

     – Não estamos violando nenhuma regra da LGPD e os dados do Climatempo estão devidamente regulamentados. É preciso estar na tela o “por Climatempo”.

     – Níveis de permissionamento diferentes entre Administrador, Dono e demais usuários daquela conta.

* **6. Ambiente:**

     – Aplicativos na versão iOS e Android

     – Pipeline de deploy configurado

     – Precisa ser criado um ambiente de homologação para essa feature.

* **7. Qualidade (critérios de aceite):**

     – Informações carregadas na tela ficam em cache para uso offline

     – O usuário pode cadastrar notificações de aviso sobre a previsão via push

     – Ao abrir o app o usuário verá na tela o dia seguinte

     – Com scroll para cima, o usuário avança nos dias

     – Uso de dados do calendário do dispositivo para definir data de abertura no app

     – Dados da Climatempo serão armazenados na AWS; os recursos de servidores já foram alocados pelo time de DevOps; os dados dos usuários serão encriptados em curva elíptica de 384 bits para autenticação

     – Com scroll para baixo, o usuário retorna nos dias.

Além da técnica acima, você também pode aplicar a “BDD – Sintaxe Gherkin”, para criação de cenários de testes.

O template é o seguinte:

* **\*\*Feature\*\*:**Tela com probabilidade de chuva e volume para o dia seguinte
* **Contexto:** O usuário, ao abrir o aplicativo, não terá conexão com a internet via rede G ou wifi. Informações carregadas na tela ficam em cache para uso offline em uma nova abertura. Nesse caso, a interface do app avisará a falta de conexão e o horário da última atualização com a internet.
* **\*\*Scenario\*\* (cenário):**usuário sem rede G abrindo o aplicativo
* \*\*Given\*\* a falta de conexão com a internet
* \*\*When\*\* o usuário abrir o app da “Empresa XPTO”
* \*\*Then\*\* o app apresentará uma mensagem via toast de falta de conexão
* \*\*And\*\* apresentará os dados em cache da última abertura com conexão
* \*\*Then\*\* o app iniciará uma rotina de novas tentativas de conexão a cada 5 segundos
* \*\*And\*\* quando a conexão for estabelecida
* \*\*Then\*\* apresentará uma nova mensagem via toast avisando que a conexão foi restabelecida
* \*\*And\*\* os dados atualizados serão exibidos na tela.