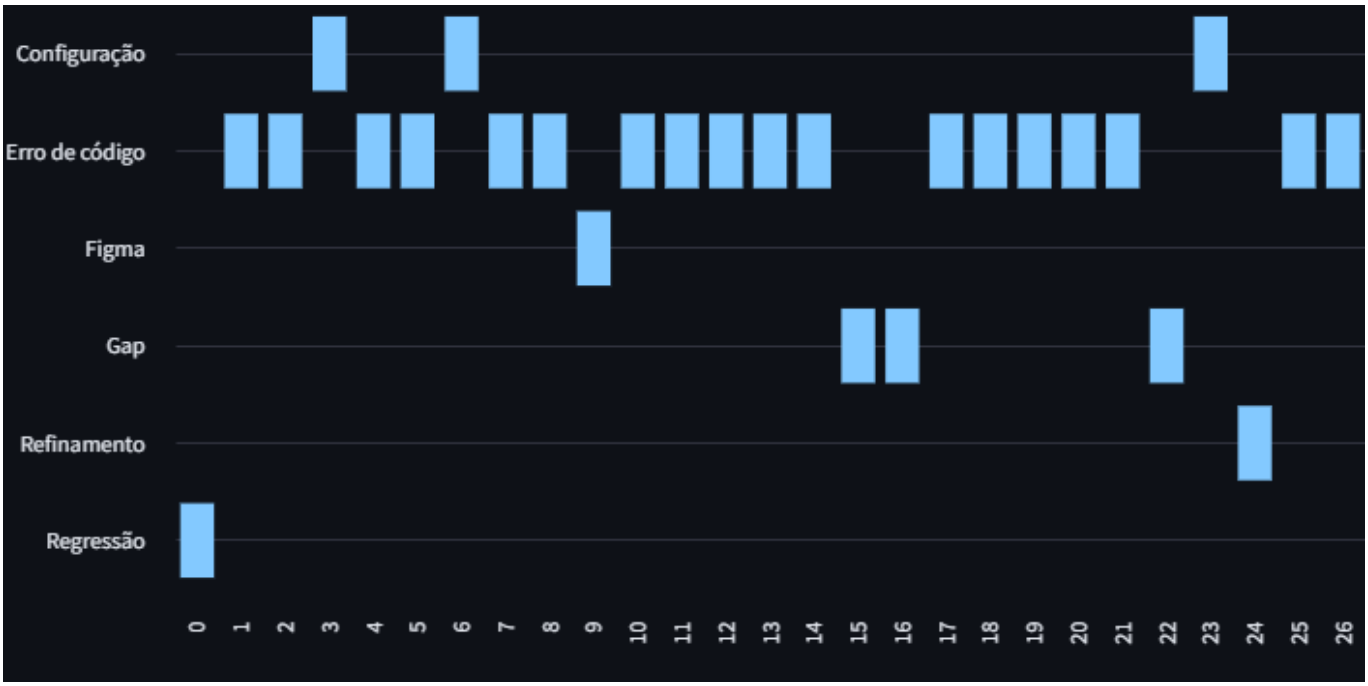
Incidência de Bugs por Categoria



Categoria com maior incidência: Erro de código

Descrição combinada: [DOC] Corrigir rotina de restauração de arquivos no bloco de documentos Ao tentar editar um vinculo, os dados exibidos ao ativar a edição não estão vindo conforme o vinculo selecionado. O sistema não enviou o email para esta notificação e cadastro de uma mudança de escopo. A notificação de cadastro de mudança de escopo não foi gerada na aplicação. Dados não estão persistindo após o salvamento do rascunho : "Quais as mudanças de escopo do projeto ?" e "Classificação do projeto" Ao ir para tela de cadastro de ME um projeto greenfield e mudar a classificação para brownfield, é necessário selecionar um complexo novamente para exibir a unidade operacional Falta exibir a mensagem informando que ja existe um parecer concluído, a mensagem esta sendo retornada, mas não exibida O texto na notificação via aplicação esta incorreto Após a conclusão ou cancelamento do ME não deveria ser possível ter mais ações relacionadas a requisição Ao tentar cadastrar uma nova comunicação em uma subatividade, esta sendo exibido o nome da atividade pai no breadcrumb Não estão sendo exibidos os arquivos anexados no momento de cadastro da ME não deveria ser possível fazer ações relacionadas ao parecer final quando uma RIPLA esta em processo de revisão Quando o parecer final trás um "estudo ambiental necessário" de uma disciplina, se o usuário cadastrar uma estrategeia de licenciamento, a listagem dos processos de licenciamentos ficam sem ação Listagem de mudanças deve ser exibida em ordem alfabética Ocultar botão de download de anexos, ele só deve ser exibido após o upload do anexo no sistema, seguindo o padrão da aplicação O campo "Qual estimativa de impacto em tempo da mudança de escopo ?" não esta atualizando o resumo do parecer, sendo necessário editar outro campo para que o resumo seja atualizado Corrigir texto da notificação enviada via E-mail Está abrindo o parecer, para edição, ao clicar no subtítulo do bloco de parecer Deveria ser apenas ao clicar do botão "Editar parecer"

Solução Proposta: ## Análise de Bugs e Soluções para um Ambiente Ágil Angular/Java

A lista de bugs apresentada indica alguns problemas recorrentes na equipe de desenvolvimento:

1. Alta incidência de bugs de Regressão:

A presença de diversos bugs de regressão sugere que os testes de regressão não estão sendo eficientes o suficiente. Isso pode ser devido a:

- Testes de regressão insuficientes: A equipe pode não estar testando o código o suficiente após cada alteração, ou pode estar testando apenas as funcionalidades mais visíveis.
- Testes de regressão incompletos: Pode haver falhas na cobertura dos testes, deixando de testar certas funcionalidades ou cenários específicos.
- Falta de integração contínua (CI): A falta de um pipeline de CI que execute testes de regressão automaticamente após cada commit pode aumentar o risco de bugs de regressão.

Soluções:

- Implementar testes de regressão automatizados e abrangentes: Utilizar ferramentas de testes como Cypress para Angular e JUnit para Java e garantir a cobertura de todas as funcionalidades e cenários importantes.
- Automatizar os testes de regressão com integração contínua: Integrar os testes de regressão ao pipeline de CI para execução automática após cada commit, garantindo que as alterações não introduzam regressões.
- Criar um sistema de testes de regressão em diferentes níveis: Testes de unidade, testes de integração e testes de ponta a ponta (end-to-end) para garantir a cobertura de todas as camadas do sistema.

---

2. Erros de código:

A grande quantidade de erros de código indica uma possível falta de atenção aos detalhes, falta de revisão do código, ou problemas com as ferramentas de desenvolvimento:

- Falta de revisão de código: A ausência de um processo de revisão de código pode permitir que erros de código passem despercebidos.
- Falta de padrões de codificação: A ausência de padrões de codificação pode levar a código inconsistente e erros difíceis de identificar.
- Ferramentas de desenvolvimento inadequadas: Ferramentas de desenvolvimento ineficazes ou mal configuradas podem aumentar o risco de erros de código.

Soluções:

- Implementar um processo rigoroso de revisão de código: Designar pares de programadores para revisar o código antes de ser integrado ao projeto.
- Definir e aplicar padrões de codificação: Utilizar padrões de codificação como Stylelint para Angular e Checkstyle para Java para garantir a consistência e legibilidade do código.
- Utilizar ferramentas de desenvolvimento eficientes: Utilizar IDEs com funcionalidades de auto-completar, detecção de erros e refatoração para minimizar a ocorrência de erros.

---

3. Erros de configuração:

Os erros de configuração indicam problemas com a configuração do ambiente de desenvolvimento ou do próprio aplicativo:

- Documentação de configuração desatualizada: A documentação do sistema pode estar desatualizada, levando a erros de configuração.
- Falta de padronização na configuração: A falta de padrões para a configuração do ambiente pode levar a configurações inconsistentes e conflitos.
- Falta de testes de configuração: A ausência de testes específicos para verificar a configuração do aplicativo pode aumentar o risco de erros.

Soluções:

- Manter a documentação de configuração atualizada e precisa: Garantir que a documentação esteja sempre sincronizada com as versões mais recentes do projeto e do ambiente.
- Criar padrões para a configuração do ambiente: Definir padrões para a configuração do ambiente de desenvolvimento, incluindo o gerenciamento de dependências, a configuração do servidor e outras configurações relevantes.
- Implementar testes de configuração automatizados: Criar testes específicos para verificar se a configuração do aplicativo está correta e se o ambiente de desenvolvimento está configurado corretamente.

---

4. Gaps e Refinamentos:

A presença de "gaps" e a necessidade de refinamento indicam falta de comunicação e planejamento:

- Falta de comunicação entre as equipes: A falta de comunicação entre a equipe de frontend (Angular) e a equipe de backend (Java) pode levar a inconsistências e "gaps" nas funcionalidades.
- Definição de requisitos imprecisa: A definição de requisitos incompleta ou imprecisa pode levar a erros de implementação e necessidade de refinamentos posteriores.
- Falta de documentação clara: A ausência de uma documentação clara e organizada pode dificultar o entendimento do projeto e levar a erros.

Soluções:

- Melhorar a comunicação entre as equipes: Implementar ferramentas de comunicação eficazes e realizar reuniões regulares para garantir que as equipes estejam alinhadas sobre os requisitos e a implementação do projeto.
- Definir requisitos precisos e completos: Utilizar técnicas de elicitação de requisitos para garantir que todos os requisitos estejam bem definidos e compreendidos por todos os membros da equipe.
- Criar uma documentação clara e organizada: Utilizar ferramentas de documentação como Swagger para a API e Readme para o projeto para manter a documentação sempre atualizada e acessível.

---

5. Erros de Figma:

A presença de erros relacionados ao Figma indica problemas na comunicação visual e na tradução do design para o código:

- Falta de especificações detalhadas: A falta de especificações detalhadas sobre os elementos visuais pode levar a erros de implementação e interpretações divergentes.

- Falta de comunicação entre designers e desenvolvedores: A falta de comunicação eficiente entre designers e desenvolvedores pode levar a problemas na tradução do design para o código.
- Falta de testes visuais: A ausência de testes visuais específicos pode levar a erros de implementação que afetam a experiência visual do usuário.

Soluções:

- Criar especificações detalhadas para o design: Definir as especificações detalhadas para cada elemento visual, incluindo medidas, cores, fontes e interações.
- Fomentar a comunicação entre designers e desenvolvedores: Realizar workshops e revisões regulares para garantir que designers e desenvolvedores estejam em sintonia com o design do projeto.
- Implementar testes visuais automatizados: Utilizar ferramentas de testes visuais como Percy para garantir a consistência do visual do aplicativo.

Considerações finais:

A implementação das soluções propostas deve ser feita de forma gradual e iterativa, adaptando-se às necessidades específicas da equipe e do projeto. O objetivo é criar um ambiente de desenvolvimento ágil e eficiente que minimize a incidência de bugs, garantindo a entrega de um produto de alta qualidade.