ENGENHARIA DE SOFTWARE

6º Semestre - Noturno

GERENCIAMENTO DE VERSÃO DE SOFTWARE

GEST-EDUCA

Trabalho apresentado ao 6º Período do curso de Engenharia de Software, da Universidade Cesumar, pelos alunos: Eduardo Richard da Silva Nascimento, RA-21161812-2; Karla Duarte Ferreira, RA-21144154-2;

MARINGÁ

2023

GERENCIAMENTO DE VERSÃO DE SOFTWARE
GEST-EDUCA

MARINGÁ
2023

SUMÁRIO

PROJETO

Projeto	**GEST-EDUCA ERP**
Gerente de Projetos	**Eduardo Richard, Karla Duarte**
Analista de Teste	**Karla Duarte**
Analista de Qualidade	**Karla Duarte**
Administrador de Banco de Dados	**Eduardo Richard**
Desenvolvedor Sênior	**Eduardo Richard**
Responsável pela documentação	**Eduardo Richard, Karla Duarte**

HISTÓRICO DE REVISÕES

Data	Versão	Descrição	Autor
15/09/2023	1.0	Elaboração do documento.	Eduardo Richard, Karla Duarte
25/09/2023 a 09/12/2023	2.0	Novidades Funcionalidades Sistema	Eduardo Richard, Karla Duarte
11/12/2023 a 25/02/2024	3.0	Novidades Funcionalidades Sistema	Eduardo Richard, Karla Duarte

1. Introdução

Este documento delineia nosso plano estratégico para as próximas versões do sistema "GEST-EDUCA" com um foco direcionado aos desenvolvedores. Nosso objetivo é fornecer um roteiro claro para o desenvolvimento e aprimoramento contínuo deste sistema educacional.

Nesta documentação, você encontrará informações detalhadas sobre as funcionalidades planejadas para cada versão, bem como a estratégia técnica que implementaremos na terceira versão para melhorar o desempenho do sistema. A ênfase principal é capacitar nossa equipe de desenvolvimento a entender e executar com precisão os próximos passos.

Exploraremos métricas de desempenho, como tempo médio de resposta do servidor, latência da rede e outras, que serão fundamentais para avaliar o sucesso de nossos esforços. Além disso, apresentaremos um plano de implementação detalhado com cronogramas específicos para cada etapa técnica.

Este documento serve como uma ferramenta de referência essencial para os desenvolvedores envolvidos na execução das próximas versões do "GEST-EDUCA". Esperamos que ele forneça a você uma visão clara de nossos objetivos técnicos e inspire a colaboração eficaz da equipe para alcançar essas metas.

2. Versionamento Semântico:

O nosso projeto adota uma abordagem de versionamento semântico rigorosa para o controle das versões do software. Essa abordagem é essencial para garantir que as versões sejam consistentes e sigam um padrão claro de incremento, facilitando a compreensão e o gerenciamento das mudanças ao longo do tempo.

2.1 Estrutura de Versão

As nossas versões de software são estruturadas em três números distintos: SUPER.MINOR.PATCH. Cada um desses números tem um significado específico e é atualizado de acordo com a natureza das alterações implementadas.

- **SUPER**: O número SUPER representa a versão principal do software. Qualquer alteração significativa que introduza incompatibilidades com versões anteriores ou que represente uma redefinição substancial do sistema resultará em um incremento no número SUPER. Isso indica uma mudança importante e impactante para os usuários.
- **MINOR**: O número MINOR é incrementado quando novas funcionalidades são adicionadas ao software de forma compatível com as versões anteriores. Isso significa que as alterações introduzidas são novas funcionalidades ou melhorias que não afetam a funcionalidade existente de maneira negativa.
- **PATCH**: O número PATCH é atualizado para correções de bugs e pequenas melhorias que não afetam as funcionalidades principais do software. Isso reflete ajustes menores destinados a aprimorar a estabilidade e a qualidade do software.

2.2 Exemplos de Atualização de Versão

Para ilustrar como nossa abordagem de versionamento semântico funciona na prática, aqui estão alguns exemplos:

- Se implementarmos uma nova funcionalidade significativa que introduza incompatibilidades com a versão anterior, faremos um incremento no número SUPER. Por exemplo, de 2.0.0 para 3.0.0.
- Se adicionarmos novas funcionalidades ou melhorias sem introduzir incompatibilidades, faremos um incremento no número MINOR. Por exemplo, de 2.1.0 para 2.2.0.
- Se realizarmos correções de bugs ou pequenas melhorias que não afetem as funcionalidades principais, faremos um incremento no número PATCH. Por exemplo, de 2.1.3 para 2.1.4.

2.3 Importância do Versionamento Semântico

A adoção do versionamento semântico em nosso projeto tem várias vantagens significativas. Isso inclui:

- Clareza: Os números de versão fornecem informações claras sobre o tipo de alterações implementadas em cada versão, facilitando a compreensão das mudanças pelos usuários e membros da equipe.
- Rastreabilidade: O versionamento semântico permite rastrear com precisão quando e como as mudanças foram introduzidas no software, o que é essencial para fins de auditoria e solução de problemas.
- Compatibilidade: Os números de versão ajudam os usuários a determinar rapidamente se uma atualização é compatível com suas necessidades e infraestrutura existente.
- Comunicação Eficaz: O uso consistente dessa abordagem de versionamento facilita a comunicação entre os membros da equipe, stakeholders e usuários finais, pois todos têm um entendimento compartilhado sobre o significado das versões.

3. Mudanças nas versões X.X

Nesta seção, descreveremos as funcionalidades planejadas e as melhorias de desempenho para as próximas versões do sistema "GEST-EDUCA".

3.1. Primeira Versão (Versão 1.0)

A primeira versão do sistema, com previsão para lançamento em 01/10/2023, conterá as seguintes funcionalidades:

1. CRUD - Docentes e Alunos

2. CRUD - Cursos
3. CRUD - Matérias
4. CRUD - Turmas
5. CRUD - Notas

3.2 Segunda Versão (Versão 2.0)

A segunda versão do sistema, com previsão para lançamento em 01/12/2023, terá as seguintes funcionalidades planejadas e documentadas:

1. Chat
2. Avaliação de desempenho dos Docentes
3. Biblioteca Virtual
4. Solicitações de Serviços

3.3 Terceira Versão (Versão 3.0)

Para a terceira versão do sistema, planejamos uma estratégia de melhoria de desempenho que permitirá dobrar o número de requisições simultâneas feitas no aplicativo. Os detalhes dessa estratégia de melhoria de desempenho serão documentados separadamente e incluirão os seguintes aspectos:

1. Descrição da estratégia de melhoria de desempenho.
2. Recursos técnicos envolvidos na implementação.
3. Metodologia de teste para validar a melhoria.
4. Cronograma previsto para a implementação.
5. Impacto esperado na aplicação após a implementação.

Além disso, planejamos incorporar as seguintes métricas para avaliar o desempenho e a qualidade do sistema.

Métricas de Desempenho:

1. Tempo médio de resposta do servidor.
2. Tempo médio de carregamento de página.
3. Latência da rede.
4. Outras métricas de desempenho críticas.

Métricas de Uso:

1. Número de usuários ativos.
2. Taxa de crescimento de usuários.
3. Frequência de uso.
4. Outras métricas relacionadas ao uso do aplicativo.

Métricas de Qualidade de Código:

1. Pontuação média de qualidade de código.
2. Número de problemas de código identificados.
3. Taxa de resolução de problemas.

Métricas de Monitoramento de Logs:

1. Registros de erros e exceções.
2. Informações detalhadas de logs de aplicativos.
3. Eventos relevantes do sistema.

3.3.1 Estratégia de Melhoria de Desempenho para Dobrar Requisições Simultâneas

Essas métricas serão usadas para avaliar o sucesso da estratégia de melhoria de desempenho e garantir que a terceira versão do sistema atenda aos padrões de desempenho e qualidade definidos.

3.3.2 Objetivo da Melhoria

O objetivo geral da melhoria de desempenho é proporcionar uma experiência de usuário mais eficiente e satisfatória, permitindo que o sistema lide com um volume maior de requisições simultâneas sem comprometer a qualidade ou a velocidade de resposta. Isso visa atender às crescentes demandas dos usuários e garantir que o sistema seja escalável para o futuro.

3.3.3 Descrição da Estratégia

1. Redução de Tempo de Resposta: Reduzir o tempo médio de resposta do servidor para proporcionar respostas mais rápidas às solicitações dos usuários.

2. Aumento da Capacidade de Carga: Aumentar a capacidade de carga do sistema, permitindo que ele suporte um maior número de usuários simultaneamente.

3. Melhoria da Estabilidade: Aumentar a estabilidade do sistema, garantindo que ele possa lidar com picos de tráfego sem falhas ou interrupções significativas.

4. Melhoria na Experiência do Usuário: Aprimorar a experiência geral do usuário, tornando o aplicativo mais ágil e responsivo.

5. Preparação para o Futuro: Preparar o sistema para lidar com um crescimento contínuo no número de usuários e demandas futuras.

3.3.4 Benefícios Esperados

Esses objetivos visam melhorar significativamente o desempenho, a escalabilidade e a qualidade do sistema, garantindo uma experiência positiva para os usuários e a preparação para desafios futuros.

Ao implementar a estratégia de melhoria de desempenho, esperamos alcançar os seguintes benefícios:

1. Melhoria Significativa no Tempo de Resposta: Antecipamos uma redução significativa no tempo médio de resposta do sistema, proporcionando uma experiência mais ágil para os usuários.

2. Aumento na Capacidade de Carga: Esperamos que o sistema seja capaz de lidar com o dobro de requisições simultâneas, garantindo que ele seja escalável para atender a picos de tráfego.

3. Estabilidade Aprimorada: Antecipamos uma maior estabilidade, com menos interrupções ou falhas mesmo em situações de carga elevada.

4. Satisfação do Usuário: Esperamos que os usuários experimentem um sistema mais responsivo e eficiente, resultando em uma experiência global mais positiva.

5. Preparação para o Futuro: Com essas melhorias, estaremos preparados para enfrentar desafios futuros à medida que o número de usuários e demandas continuarem a crescer.

3.3.5 Plano de Implementação

O plano de implementação para a estratégia de melhoria de desempenho será dividido nas seguintes etapas:

Etapa 1: Otimização de Consultas de Banco de Dados (Semana 1-2)

- Realizar análise das consultas de banco de dados atuais.
- Identificar consultas que podem ser otimizadas.
- Implementar otimizações nas consultas selecionadas.

Etapa 2: Implementação do Cache de Conteúdo (Semana 3-4)

- Projetar o sistema de cache de conteúdo.
- Implementar o cache para páginas e recursos frequentemente acessados.

Etapa 3: Paralelização de Tarefas (Semana 5-6)

- Identificar tarefas de processamento intensivo.
- Implementar a paralelização dessas tarefas.

Etapa 4: Compactação e Minificação de Recursos Estáticos (Semana 7-8)

- Identificar recursos estáticos a serem compactados e minificados.
- Implementar a compactação e minificação desses recursos.

Etapa 5: Aprimoramentos na Infraestrutura de Servidores (Semana 9-10)

- Realizar atualizações na infraestrutura de servidores.
- Configurar servidores adicionais para suportar a carga.

Esse plano será implementado ao longo de 10 semanas, com cada etapa sendo concluída em seu respectivo período. O progresso será monitorado e documentado conforme avançamos em direção à nossa meta de melhoria de desempenho.

4. Fluxo de Trabalho de Versionamento

Refere-se ao conjunto de processos, práticas e etapas que uma equipe de desenvolvimento segue para gerenciar e controlar as versões de um software desde o desenvolvimento até a implantação. Esse fluxo de trabalho define como as alterações no código-fonte são rastreadas, revisadas, testadas e finalmente implantadas como novas versões do software.

Desenvolvimento: Nesta fase, os desenvolvedores trabalham nas alterações de código-fonte, seja para adicionar novos recursos, corrigir bugs ou fazer melhorias. O desenvolvimento pode ocorrer em branches separados, dependendo da complexidade do projeto.

Revisão de Código: Antes de mesclar qualquer alteração no branch principal, as alterações são revisadas por outros membros da equipe. Isso ajuda a identificar problemas de qualidade e a garantir que as alterações estejam de acordo com os padrões do projeto.

Testes Locais: Os desenvolvedores conduzem testes locais para verificar se suas alterações não introduzem novos bugs e se os recursos funcionam conforme o esperado.

Integração Contínua: As alterações são integradas regularmente no branch principal do repositório usando práticas de integração contínua. Isso envolve a execução automática de testes automatizados para garantir que as alterações não quebrem o software existente.

Testes de Qualidade: Após a integração, as alterações são submetidas a testes de qualidade mais abrangentes, incluindo testes de unidade, testes de integração e testes de aceitação para garantir que o software funcione conforme o esperado e não cause regressões.

Aprovação: Uma vez que as alterações tenham passado pelos testes e revisões necessários, elas são aprovadas para inclusão na próxima versão.

Implementação: As alterações aprovadas são implantadas no ambiente de produção ou em um ambiente de pré-produção, dependendo do processo de implantação do projeto.

Documentação: A documentação relevante, como notas de lançamento ou atualizações de documentação do usuário, é criada ou atualizada para refletir as alterações feitas na nova versão.

Marcação de Versão: Uma nova versão é marcada, geralmente por meio do sistema de controle de versão, como o Git, para registrar as alterações feitas naquela versão específica.

Comunicação: Às partes interessadas, incluindo usuários finais, são informadas sobre a nova versão e quaisquer alterações importantes.

Monitoramento e Correção: Após a implantação, a equipe monitora o desempenho da nova versão e corrige quaisquer problemas que possam surgir.

5. Contatos

Para questões relacionadas ao controle de mudanças no projeto "Gest-Educa", entre em contato com os seguintes membros da equipe de gerenciamento de mudanças:

Gerente de Projeto:

Eduardo Richard R.A: 21161812-2

Karla Duarte Ferreira RA: 211441542