**AMF - Faculdade Antônio Meneghetti**

**Curso de Sistemas de Informação**

**Gestão de Qualidade de Software**

**Professor: Rafael**

**Documento de visão**

**Santa Maria, 13 de março de 2018**

**Lista de revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| Revisão | Data | Autor | Comentário |
| 0.1 | 13/03/2018 | Diego | Primeiro esboço do projeto |
| 0.2 | 25/03/2018 | Leonardo, Carlos | Introdução do projeto, objetivos e requisitos técnicos. |
| 0.3 | 05/04/2018 | Leonardo, Carlos | Descrição das principais estratégias |
| 0.4 | 10/04/2018 | Leonardo, Carlos | Criação da tabela e início da criação dos roteiros de testes. |
| 0.5 | 24/04/2018 | Leonardo, Carlos | Criação do diagrama de classes e métricas geradas a partir do GQM. |
|  |  |  |  |

**Tabela de aprovação**

|  |  |  |  |
| --- | --- | --- | --- |
| Versão | Comentário | Responsável | Data |
|  |  |  |  |
|  |  |  |  |

Sumário

1. Introdução

O propósito deste é a elaboração da documentação, seguindo os padrões da Gestão da Qualidade de Software (GQS), do sistema de cadastro para o calendário do aplicativo para dispositivos móveis da instituição Antônio Meneghetti Faculdade (AMF).

A qualidade de um produto hoje tem importância fundamental para disputar a concorrência com as empresas do ramo. A qualidade hoje é um pré-requisito para as empresas que querem estar sempre no mercado de forma ativa. Embora a ideia de qualidade possa parecer um tanto intuitiva e de pouca complexidade, quando estudada com mais atenção, podemos ver que é algo que exige além do que imaginado. (PRESSMAN, Roger S., 1995).

Um software sem qualidade pode ocasionar erros catastróficos e de grande custo de manutenção para as empresas de softwares. A qualidade de hoje é a grande motivadora em todas as áreas, todos querem receber e fornecer produtos com qualidade. Um software de qualidade tem impacto sobre milhares de pessoas, pois ele poderá disponibilizar serviços essenciais, irá gerar competitividade entre empresas e etc. (PRESSMAN, Roger S., 1995).

Com isso, a qualidade de um software está principalmente ligada aos requisitos solicitados pelo cliente e em conformidade as regras de desenvolvimento. Existem inúmeros conceitos que nos auxiliam a manter a qualidade do software, esses conceitos serão abordados durante o desenvolvimento desse projeto.

Escopo

O escopo principal deste documento é voltado a equipe desenvolvedora do projeto. Todos os envolvidos são responsáveis pela documentação, desenvolvimento da aplicação e relato dos resultados, monitorando o progresso do projeto e testes da qualidade.

Contexto

Com o crescimento da instituição, tanto em número de pessoas como de eventos, tornou-se fundamental ter um canal de divulgação destes eventos e, aliado a disciplina GQS, será aprimorado o desenvolvimento do aplicativo que provê o cadastro, sendo desenvolvido uma nova aplicação na plataforma *web* contemplando todas as funcionalidades já existentes e as melhorias detectadas através do estudo aprimorado na disciplina GQS.

Objetivos do projeto

O aplicativo tem por finalidade realizar o cadastro de tipos de eventos, cursos e eventos no calendário utilizado no aplicativo para dispositivos móveis da AMF.

Portanto, irá auxiliar os alunos da instituição Antônio Meneghetti Faculdade quanto a conhecimento sobre eventos a serem realizados dentro da instituição, no aplicativo, administradores poderão realizar o cadastro de eventos, assim os demais alunos com o aplicativo instalado em seu smartphone, tablet, etc, poderão ter o conhecimento desses eventos a serem realizados, assim como uma breve descrição de cada.

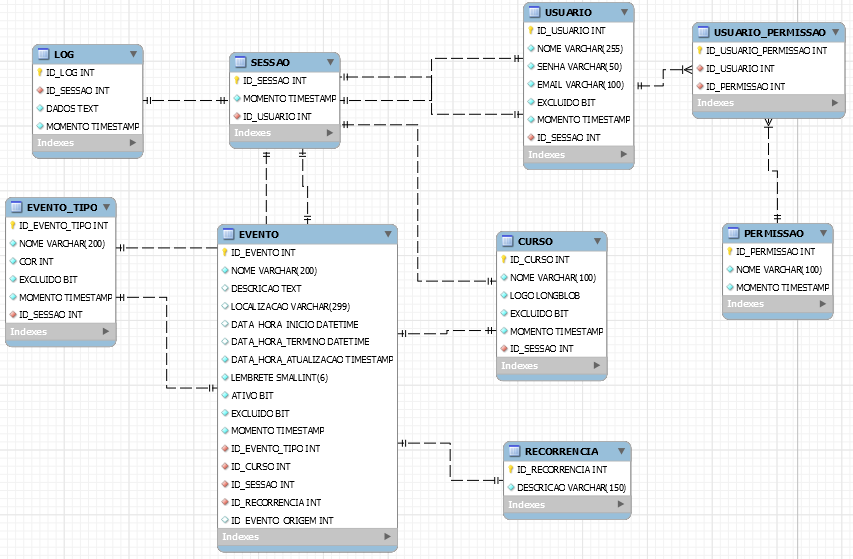
No aplicativo também existirá um calendário de notificação aos alunos, onde nele será marcado os eventos em suas respectivas datas, com isso o aluno poderá escolher eventos de sua conivência e poderá reservá-los, assim sendo notificados anteriormente sobre a realização do evento.

Objetivos da arquitetura

Vem de encontro a interface, estrutura, organização estrutural

do projeto de forma superficial.

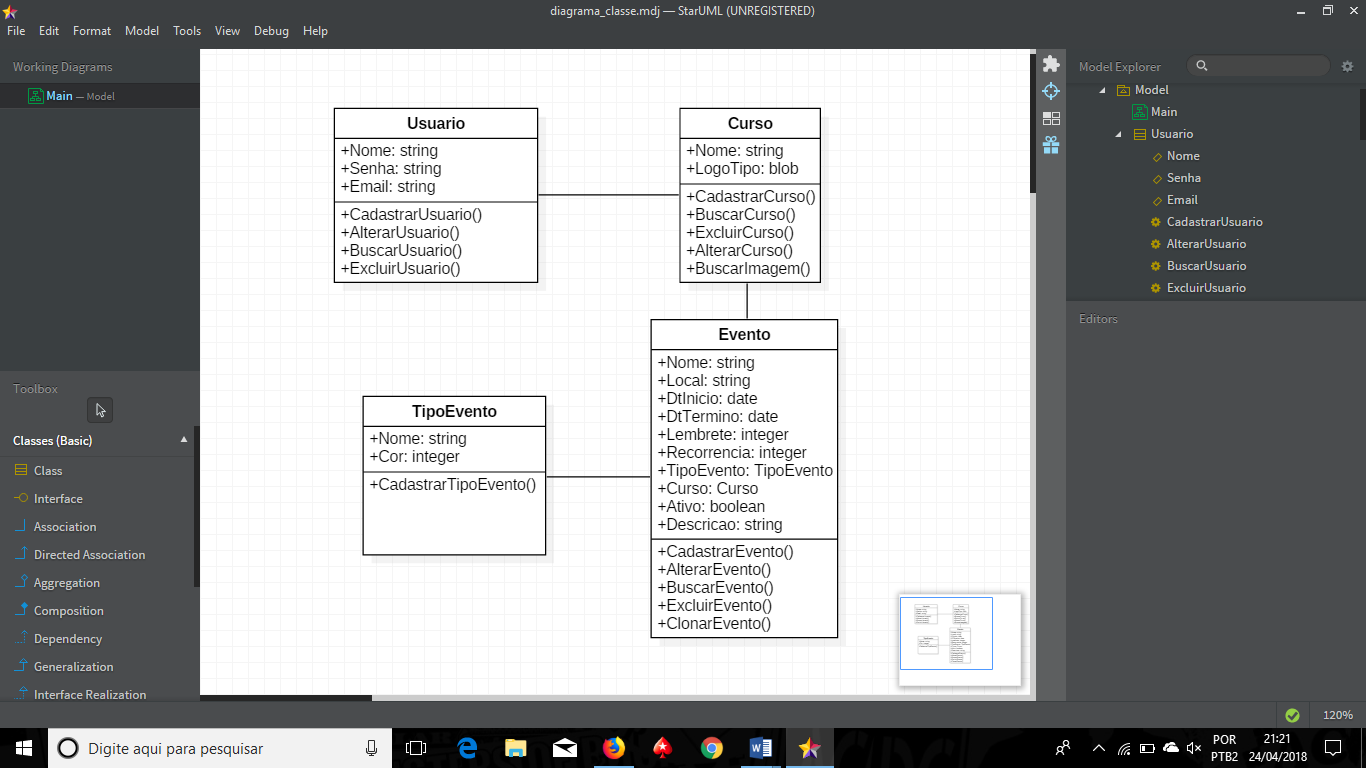
Abaixo a estrutura de nosso Banco de dados, com modelo ER:



Temos como uma representação estática utilizada para se descrever a estrutura

do sistema, apresentando suas classes, atributos e métodos, assim como suas relações.

Abaixo Diagrama de Classes:



Requisitos técnicos

Ver onde hospedar, linguagem, banco, link,

- Hospedagem Banco de dados - Andrêves

|  |  |
| --- | --- |
| Ferramenta | Ferramenta Utilizada |
| Repositório | GitHub |
| Banco de Dados | My SQL |
| Linguagens | PHP, HTML, JavaScript, CSS e JSON |
| Framework | BootStrap, JQuery |
| IDE | Visual Studio Code |

Requisitos do projeto

Vamos pensar um pouco….

2. Documentação de referência para fazer o projeto

Para a realização desse projeto utilizamos como base a norma de Qualidade de Software ISO 9126, e para realização do manual do usuário utilizamos a norma ISO/IEC 12119.

ISO 9126 - Ela estabelece uma norma para a qualidade de produto de software, definindo um conjunto de parâmetros com o objetivo de padronizar a avaliação de qualidade de Software, tendo como características principais: Funcionalidade, confiabilidade, usabilidade eficiência, manutenibilidade e portabilidade.

|  |  |  |
| --- | --- | --- |
| Característica de Qualidade | Sub característica de Qualidade | Definição |
| Funcionalidade |  | Conjunto de atributos que evidenciam a existência de um conjunto de funções e suas propriedades especificadas |
|  | Adequação | ... |
|  | Acurácia | ... |
|  | Interoperabilidade | ... |
|  | Conformidade | ... |
|  | Segurança de acesso | ... |

3. Estratégia da gestão de qualidade

Para se ter uma garantia mais ampla de qualidade em nosso software, foram implementados no projeto alguns tipos de teste de software que serão citados no decorrer do documento.

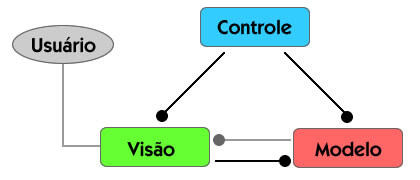
Temos Teste de software como um processo pelo qual os sistemas são executados de maneira controlada, sendo analisadas as conformidades e as funcionalidades de acordo com as especificações do projeto de desenvolvimento. O objetivo dos testes é revelar defeitos, para que estes possam ser corrigidos antes da entrega de uma versão ou da entrega final do produto (Bartié, 2002).

Para a realização desses testes, primeiramente devemos entender como está nivelada a arquitetura de nossa aplicação.

- Toda a aplicação moderna, possui diferentes camadas, nossa aplicação possui 3 camadas;

- Para se escolher a melhor estratégia de testes para a aplicação, é necessário compreender como o sistema funciona internamente e como são divididas estas camadas (*layers*).

Representação de nossas três camadas:



Usuário

Estas camadas são subdivididas em outras camadas e partes ainda menores, podemos testá-las individualmente ou em conjunto. Os tipos de testes os quais implementamos são: Teste unitário, teste de integração e teste de aceitação.

TESTE UNITÁRIO:

- Validar a menor parte de um sistema, de forma isolada, para garantir que funcione exatamente como o esperado;

- Indicado para o baixo nível;

- Funções (Validação de campos);

- A execução do teste unitário deve alertar para possíveis erros;

- Devem ser combinados com outros tipos de teste;

- Não devem ser a única estratégia de qualidade.

TESTE DE INTEGRAÇÃO:

- Validam se a comunicação entre dois componentes está funcionando de maneira esperada;

- São mais frágeis;

- Os componentes não são testados isoladamente;

- Exemplo: Comunicação da API como Banco de Dados.

TESTE DE ACEITAÇÃO:

- Normalmente envolvem todas as camadas da aplicação;

- Serve para validar um cenário de negócio;

- Testes formais que relacionam as necessidades dos usuários, requisitos e processos de negócio;

- Sistema satisfaz ou não critérios de aceitação.

- Exemplos: Login e Logout; Acessar cadastro de eventos; Adicionar localização de um evento.

|  |  |  |
| --- | --- | --- |
| **Tipo de teste** | **Descrição do teste** | **Saída esperada do teste** |
| **Unitário** | Login válido | Acesso à tela inicial |
| **Unitário** | Informações inválidas | Mensagem informativa |
| **Unitário** | Preencher o limite dos campos | Não deve dar erro, mostrar mensagem informativo |
|  |  |  |
|  |  |  |
|  |  |  |

3.1. Processo de Medição

Desenvolvimento de software requer um processo ou mecanismo de medição para feedback e avaliação. A medição ajuda no planejamento do projeto, alocação de recursos,

estimativo de custos e tempo, além de ajudar na prevenção de erros e auxílio no suporte (BASILI, R. B.; CALDEIRA, G.; ROMBACH, H.D, 2003).

Para a realização da medição de nosso software utilizamos o processo GQM (Goal Question Metric), que é um processo que visa medir objetos como: produtos, processos e recursos, ele é baseado numa estrutura, na qual se define o que vai ser medido e uma meta (objetivos). Depois são criadas questões para avaliar a produtividade e o tempo, e busca-se respondê-las através de métricas (BASILI, R. B.; CALDEIRA, G.; ROMBACH, H.D, 2003).

3.1.1. Métricas de Processo

* Medir eficência do Processo de login
  + Qual a confiabilidade no processo?
  + Quantas vezes é permitido efetuar o login com a senha incorreta ou login incorreto?
  + Quantos defeitos possui o processo de login?
  + Qual o status dos defeitos encontrados?
  + Qual o desempenho que possui o processo?
  + O código desenvolvido é claro e fácil de entender?
  + Todas as variáveis e parâmetros estão sendo usadas corretamente?
* Métricas geradas a partir dos questionamentos:
* DL = (Número de defeitos no login após implementação/ Número de defeitos encontrados a cada versão) \* 100.
* LOC = Número total de linhas de código na função.
* LCM = (Linhas de código por método/Número total de linhas de código ao final da implementação) \* 100.
* VF = (Variáveis inicias da função/Variáveis ao fim da função).
* PF = (Parâmetros inicias da função/Parâmetros ao fim da função).
* LSI = Vezes permitidas à tentativas de logar com login ou senha incorretos.
* Medir eficiência do Processo CRUD de usuários
  + Quantos usuários é possível cadastrar no sistema?
  + É possível cadastrar usuários que já existam?
  + É permitido usuários com os mesmos dados?
  + Quantos usuários podem ser alterados ao mesmo tempo?
  + Quantos usuários poderão alterar dados de outros?
  + Quais os defeitos encontrados no processo de cadastro, alteração be exclusão?
  + Quais os status de cada defeito?
  + Qual o desempenho na hora de efetuar o cadastro, alteração, exclusão e busca?
  + Quantos defeitos foram encontrados desde o início do desenvolvimento?
  + O código é fácil e simples de compreender?
  + Todas as variáveis e parâmetros estão sendo usadas corretamente?
  + Quanto tempo de desenvolvimento foi necessário para finalizar o processo de inclusão, alteração, exclusão e busca?
* Métricas geradas a partir dos questionamentos:
* DC = (Número de defeitos no cadastro após implementação/Número de defeitos encontrados a cada versão) \* 100.
* DA = (Número de defeitos na alteração após implementação/ Número de defeitos encontrados a cada versão) \* 100.
* DE = (Número de defeitos na exclusão após implementação/ Número de defeitos encontrados a cada versão) \* 100.
* DB = (Número de defeitos na busca após implementação/ Número de defeitos encontrados a cada versão) \* 100.
* VF = (Variáveis inicias da função/Variáveis ao fim da função).
* PF = (Parâmetros inicias da função/Parâmetros ao fim da função).
* TDP = Tempo total de desenvolvimento dos processos CRUD.
* DPC = Tempo total para processar um cadastro.
* DPA = Tempo total para processar uma alteração.
* DPE = Tempo total para processar uma exclusão.
* DPB = Tempo total para processar uma busca.
* LOC = Número total de linhas de código na função.

... ... ... .. .... . .. .

3.1.2. Métricas de Produto

* Revisão do Produto
* Quais as regras básicas para execução de revisões?
* Quais são as características de um documento, para obter os melhores efeitos de uma revisão?
* Quais são os critérios de aceitação para a qualidade do documento, após uma revisão?
* As revisões são um método útil e eficiente?
* Métricas geradas a partir dos questionamentos:
* NF = Número de falhas detectadas durante a revisão.
* TP = Tamanho do produto revisado.
* CP = Complexidade do produto revisado.
* TRE = Tempo de recurso estimado para revisão.
* TRR = Tempo de recurso para revisão.
* GF = Gravidade das falhas detectadas durante a revisão.

4. Documentação do projeto

Será algo constante, durante o desenvolvimento.

5. Objetivos

6. Referencial teórico

...

- BARTIÉ, Alexandre; Garantia da Qualidade de Software: Adiquirindo Maturidade. Organizacional. Rio de Janeiro: Elsevier, 2002 - 9ª Reimpressão.

- BASILI, R. B.; Caldeira, G.; ROMBACH, H.D., The Goal Question Metric Aproach, 2003.

- PRESSMAN, Roger S. Engenharia de software - Makron Books,1995.