Universidade Norte do Paraná
UNOPAR

SISTEMA DE ENSINO 100% ONLINE
SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MAYARA ALMEIDA DE SOUSA

Consultoria de Tecnologia Computacional

Rio de Janeiro,RJ
2021

MAYARA ALMEIDA DE SOUSA

Consultoria de Tecnologia Computacional

Trabalho do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas apresentado à Universidade Norte do Paraná - UNOPAR, como requisito parcial para obtenção de média semestral das disciplinas de Engenharia e Projeto de Software, Projeto Orientado a Objeto e Programação Para Web II.

Orientadores:
Gilberto Fernandes Junior;
Vanessa Matias Leite.

Nome Tutor(a) Online:
Luana Gomes de Souza

Data da Entrega: 08/05/2021

Rio de Janeiro,RJ
2021

SUMÁRIO

INTRODUÇÃO

.Para a Produção Textual Interdisciplinar (PTI) deste Quinto Semestre da graduação de Análise e Desenvolvimento de Sistemas e conforme as disciplinas de Engenharia e Projeto de Software, Projeto Orientado a Objeto e Programação Para Web II foi requerido o seguinte:

Com o foco em auxiliar grandes empresários na busca por novas soluções para suas empresas e estimular suas capacidades criativas promovendo assim o desenvolvimento de novos produtos ou serviços a StartUP de Consultoria CTP (Consultoria de Tecnologia Computacional) foi criada.

Levando sempre em consideração fatores que envolvem a sustentabilidade, inovação e um atendimento de qualidade a Start Up prestar serviços a pequenas, médias e grandes empresas e sabendo disso uma empresa de Construção Civil chamada ConstruArt solicitou os serviços da CTP e após o atendimento foi elaborado três atividades. são elas:

- Primeira Atividade - deve ser criado um relatório contendo as possíveis melhorias para os processos de software da empresa e dentro dele dois assuntos principais devem ser tratados, o primeiro é a escolha de uma metodologia ágil contendo suas características e esclarecendo o motivo da escolha. Já o segundo assunto trata-se da descrição de um modelo de maturidade que melhor atende às demandas da empresa juntamente com o nível de maturidade que se pretende alcançar e por último esclarecer o motivo da escolha ;
- Segunda Atividade - devem ser indicados quais são os tipos de linguagens orientada a objeto e os padrões de projetos mais comuns no Desenvolvimento de Sistemas seguidos da definição dos Frameworks. sua importância para os projetos de sistemas orientados a objetos e dois exemplos que correspondam aos interesses da ConstruArt ;
- Terceira Atividade - foi constatado que o padrão MVC pode ajudar a ConstruArt que está enfrentando dificuldades em seus projetos de desenvolvimento web, sendo assim foi pedido uma explicação do que é o padrão MVC, sua importância, quais são os papéis de cada camada e por último os benefícios de se utilizar este padrão, onde deve ser usado um exemplo que ilustre sua utilização.

Para que isso seja realizado os conteúdos das disciplinas norteadoras,mencionadas acima, será usado e para que se tenha um melhor entendimento e acompanhamento das tarefas requeridas pela PTI estes conteúdos se encontram organizados em etapas específicas.

DESENVOLVIMENTO

1. Metodologias Ágeis

As Metodologias Ágeis (Agile) surgiram como uma resposta ao método tradicional de criação de software, isto significa que o modo tradicional é antiquado para os tempos de hoje? De forma alguma, já que a base das metodologias ágeis vieram do modo tradicional de se desenvolver um software.

Para que se entenda de fato o que são as metodologias ágeis e a sua devida importância no processo de criação e de manutenção de sistemas será preciso mencionar dois assuntos: A Crise do Software e o Manifesto Ágil.

Crise do Software.

A Crise do Software foi um termo que surgiu nos anos 70 e expressava as dificuldades no desenvolvimento de software referentes ao rápido crescimento da demanda por software, a complexidade dos problemas a serem resolvidos e da inexistência de técnicas estabelecidas para o desenvolvimento de sistemas que funcionassem adequadamente ou que pudessem ser validadas.

No início dos anos 70, enquanto se vivia a terceira era do software, houveram muitos problemas com relação aos prazos, custos, baixa produtividade, qualidade e difícil manutenção do software.

Os problemas mais comuns enfrentados no desenvolvimento de software são:

- ❏ As Estimativas de prazo e de custo imprecisas;
- ❏ A Produtividade das pessoas da área de software não acompanha a demanda;
- ❏ Prazos Ultrapassados e Custos acima do previsto;

- A Não ênfase na facilidade de manutenção como um critério importante, gerando assim custos de manutenção elevados;
- O Não atendimento dos requisitos do usuário;
- E os altos custos de hardware e software.

Como consequência, 1/3 dos projetos passavam por cancelamentos e 2/3 extrapolavam o orçamento.

A solução encontrada para a crise do software foi a utilização de técnicas, ferramentas e processos de forma sistematizadas para produzir um software. Além de treinamentos e educação em conjunto com a mudança de paradigma sobre o que é desenvolver software e de como deveria ser feito.

Diante desse cenário focado em contornar a crise do software e dar ao desenvolvimento de sistemas um tratamento mais controlado e sistemático surgiu a Engenharia de Software.

O Manifesto Ágil

O Manifesto Ágil é um pequeno documento de texto baseado em 4 valores e 12 princípios para um desenvolvimento ágil de software e foi publicado dos dias 11 a 13 de fevereiro de 2001 em Utah.

O Manifesto ágil foi a consequência de um trabalho de 17 desenvolvedores que tinham por interesse buscar uma alternativa aos atuais processos de desenvolvimento de software assim os desenvolvedores de software envolvidos decidiram que o conteúdo da reunião deveria ser documentado e por este motivo resolveram elaborar o tal documento que foi chamado de Manifesto para o Desenvolvimento Ágil de Software ou Manifesto Ágil (Agile Manifesto).

O Objetivo principal do Manifesto Ágil é servir como um guia que aponta os rumos de um time ágil visando a potencialização de seus projetos e a escalabilidade em seus resultados , sendo assim possui os os seguintes pilares:

- Indivíduos e interação mais do que processos e ferramentas;
- Software em funcionamento mais do que uma documentação abrangente;
- Colaboração mais do que negociação de contratos;
- Responder a mudanças mais do que seguir um plano.

Com a Crise de Software deu-se origem a um modo de elaboração, produção e manutenção de software mais organizado e sistematizado conhecido como modelo tradicional de desenvolvimento de software. Já o Manifesto Ágil revelou a necessidade de tornar este modelo mais eficiente e prático a fim que no seu resultado final o software possa ter mais qualidade.

1.1 O QUE SÃO AS METODOLOGIAS ÁGEIS ?

.As Metodologias Ágeis é um conjunto de técnicas com suas especificidades que visam melhorar a qualidade na produção de softwares obtendo assim uma melhora na satisfação do cliente e tal satisfação não ocorre por acidente, afinal o cliente possui uma participação mais ativa durante todo processo de desenvolvimento se comparado com o modelo tradicional .

As Metodologias ágeis trazem como benefícios a economia de tempo, custos, eficiência e agilidade, pois permitem a eliminação de elementos desnecessários ao sistema melhorando assim a qualidade do produto final. Outro fator importante é a forma como é usado a retroalimentação no modelo tradicional dependendo da fase em que o projeto se encontra ela pode atrasar e tornar o desenvolvimento mais caro pelo fato que se tem em revisitar outras etapas contudo nas metodologias ágeis essa retroalimentação pode ser feita mais vezes identificando e solucionando alguns problemas que podem ser detectados bem no início do projeto reduzindo assim os custos.

A aplicação das metodologias ágeis também traz melhorias na rentabilidade dos negócios, pois novos investimentos podem ser feitos de maneira mais rápida. Sendo assim as principais metodologias são:

Extreme Programming ou XP

Como o principal objetivo desta ferramenta é ajudar nas relações entre funcionários e clientes, o XP se torna útil para startups ou empresas que estão passando pelo processo de consolidação.

A chave para o seu sucesso se encontra no fortalecimento das relações pessoais através do trabalho em equipe, incentivando a comunicação e eliminando o tempo de inatividade

Suas principais fases são:

- Planejamento de projetos com o cliente;

- Design do projeto;
- Codificação em pares, onde os programadores trabalham em pares com a finalidade de obterem resultados mais eficientes e de qualidade;
- Testes para verificar se os códigos, que estão sendo implementados, funcionam corretamente.

Scrum

Baseia-se em uma estrutura de desenvolvimento incremental, ou seja, após cada ciclo de desenvolvimento chamado de Sprint um pedaço do produto é liberado. ele é dividido em diferentes etapas, são elas: análise, desenvolvimento e testes.

Na fase de desenvolvimento, encontram-se as interações de processo ou Sprint, isto é, ciclos regulares que duram de 2 a 3 semanas, onde a equipe trata das funcionalidades abordadas na Sprint Backlog ocasionando em entregas regulares e parciais do produto final.

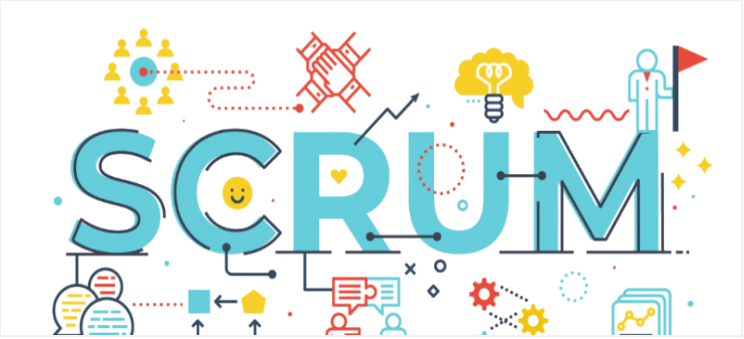
As reuniões são o pilar fundamental da metodologia e nelas são feitas a diferenciação entre o planejamento. As reuniões são divididas em: diárias, de revisão e de retrospectiva que é a mais importante, pois como ela acontece ao término de um sprint a equipe consegue refletir sobre o projeto e propor melhorias para o seu progresso.

Essa metodologia permite que projetos que são complexos e que exigem flexibilidade tenham a velocidade essencial para que os seus resultados sejam executados, sendo assim pode se dizer que o Scrum é uma metodologia voltada para a gestão dos projetos. Embora seja semelhante ao XP, o Scrum possui práticas e graus de importância diferentes.

Os principais aspectos do Scrum são:

- Inovação;
- Flexibilidade;
- Competitividade;
- Produtividade.

1.1 A METODOLOGIA USADA NA CONSTRUART.



A Metodologia escolhida para atuar na empresa de construção civil, ConstruArt é a Scrum e o motivo desta escolha é justamente por sua estrutura.

Como foi dito a metodologia Scrum é dividida em Sprints que representam os ciclos do projeto e o responsável por gerenciar e definir o que vai ser feito por cada um dos desenvolvedores durante os ciclos é conhecido como Product Owner, sua função principal é indicar quais requisitos serão tratados e definir a ordem de cada um, além do Product Owner existe o Scrum Master que atua em conjunto com ele.

Conhecido como o facilitador do projeto, o Scrum Master tem como função atuar como um moderador entre a equipe de desenvolvimento e o Product Owner evitando assim que a equipe assuma tarefas acima de suas capacidades de executá-las.

A equipe de desenvolvimento que é, geralmente, composta por seis ou dez pessoas é chamada de Scrum Team e na prática esses membros interagem com os elementos do Scrum, são eles: Product Backlog, Sprint Backlog e Sprint e tudo isso acontece da seguinte forma:

- Os Requisitos são levantados e estes precisam ser o suficiente para que projeto seja iniciado, ou seja, os requisitos não precisam ser extremamente claros e completos no início do projeto só o suficiente para que ele possa começar. Outro detalhe é que o próprio cliente é quem faz esse levantamento, sendo supervisionado pelo Product Owner;
- Após isso a Product Backlog é criada, que é uma lista com todas as funcionalidades requeridas pelo cliente para o seu produto e a partir dela as tarefas são definidas e passadas ao Scrum Team. A ordem de prioridade é definida pelo Product Owner e repassada a equipe e essa lista contendo todas as informações do cliente e da ordem de prioridade é conhecida como Sprint Backlog;

- Tanto o Product Backlog e o Sprint Backlog foram feitos dentro de ciclos chamados de Sprints que representam uma estimativa do tempo que o produto irá levar para ficar pronto;
- Por último temos as reuniões diárias que são rápidas, dinâmicas e acontecem antes e depois de cada Sprint e quando o projeto é construído de fato é feita uma última reunião, onde é realizado uma revisão de cada sprint elaborada para aquele projeto.

Ao ser aderido na ConstruArt espera-se que seus projetos obtenham as seguintes vantagens:

- Agilidade - Reduzindo o tempo de entrega do produto e obtendo uma previsão de prazos mais concretos;
- Qualidade - Como consequência o cliente será beneficiado com a entrega de um produto ou serviço de mais valor;
- Customização do Produto - Caso ocorra algum imprevisto a empresa terá mais chances de lidar bem com a situação;
- Múltiplas Entregas - Como ao final de cada Sprint é entregue um parte do produto a empresa poderá contar sempre com o suporte do cliente e isso possibilitará que múltiplas entregas sejam feitas entre outras melhorias.

2. Os Modelos de Maturidade.

2.1 O que são os Modelos de Maturidade ?

Um Modelo de Maturidade é uma forma de avaliar o quão hábil uma organização está para gerenciar os seus projetos. É como se um profundo diagnóstico dentro do negócio fosse feito e a partir dele pudessem ser pontuadas suas falhas e com isso um plano de ação para eliminar essas falhas é desenvolvido.

Como o cumprimento das tarefas, geralmente, está subordinado a diversos fatores dentro de uma empresa como a equipe, estrutura, controle de atividades, organização e etc. É bem comum que os obstáculos surjam durante a execução de um projeto e com isso cada gerente de projeto deve observar e buscar as melhores maneiras de resolver seus riscos e suas dificuldades para que o sucesso integral seja obtido.

As vantagens de se utilizar os Modelos de Maturidade consiste em:

- ❏ Desenvolvimento de Estruturas de Governança;
- ❏ Padronização e Integração de Processos;
- ❏ Utilização de Métricas de Desempenho;
- ❏ Controle e Melhoria Contínua de Processos;
- ❏ Priorização de Projetos e Alinhamento com a Estratégia Organizacional;
- ❏ Utilização de Critérios para Continuação ou Término de Projetos entre outras.

Quanto a escolha dos modelos maturidades cabe a cada empresa definir o que melhor se encaixa com os seus objetivos, neste caso para a empresa ConstruArt é proposto dois modelos, são eles:

O Modelo CMMI foi projetado para descrever os níveis de melhorias de processos por meio de estágios que oferecem uma ordem, recomendada,de aplicação dessas melhorias.

Estes estágios concentram sobre si as práticas que as empresas devem adotar para alcançar as melhorias desejadas dos seus processos e essas práticas se diferenciam umas das outras conforme o nível de maturidade que se pretende atingir.

Antes de começar a utilizar um modelo CMMI, a empresa deve mapear os seus próprios processos e medir com as áreas de processos do CMMI para que assim seja possível identificar o nível de conformidade da organização com o modelo, pois através desta avaliação a empresa poderá compreender quais práticas que ela adota se encontram em conformidade o CMMI e quais devem ser mudadas uma vez que é preferível que cada área de processo do CMMI se relacione com mais de um processo da organização.

Os Níveis de maturidade do CMMI são:

- ❏ Inicial;

- ❏ Gerenciado;
- ❏ Definido;
- ❏ Gerenciado Quantitativamente;
- ❏ Otimizado.

O OPM3 é um padrão mundialmente reconhecido voltado às melhores práticas para avaliar e desenvolver capacidades organizacionais na Gestão de Projetos, Programas e Portfólios. Neste método utiliza-se um questionário que gera quatro gráficos uma listas de quais são as melhores práticas utilizadas pela organização e com esse resultado o OPM3 permite que a situação atual da gestão de projetos da empresa seja visualizada levando em consideração uma série de atributos.

São eles a maturidade em Projetos, Programas e o Grau de Utilização das Melhores Práticas que é subdividida em quatro outros atributos: Padrões, Métricas, Controles e Melhorias. Tendo um bom entendimento da diferença do valor atual e do desejado permite que seja feito o desenvolvimento de um plano de longo prazo e assim uma estimativa do esforço e tempo necessário para a implementação das melhorias pode ser feito.

O OPM3 é dividido em cinco níveis de maturidade que são classificados como:

- ❏ Inexistente;
- ❏ Consistente;
- ❏ Integrado;
- ❏ Completo;
- ❏ Otimizado.

2.2 O Nível de Maturidade Escolhido e o Modelo Adotado.

Com os resultados que serão alcançados pela adoção das técnicas presentes nos modelos de maturidade uma empresa consegue gerenciar os seus próximos passos, sendo assim foram feitas duas propostas a empresa ConstruArt e após uma análise o modelo escolhido foi o CMMI.

Com a adoção do CMMI além de ter uma base tecnológica mais forte a ConstruArt obterá as seguintes vantagens:

- Maior Controle Produtivo;
- Melhora na Interação entre as Equipes ;
- Agilidade no Atendimento das Demandas ;
- Diminui os Riscos Durante o Planejamento ;
- O Aumento na Satisfação do Cliente.

Quanto aos níveis de maturidade, o CMMI é dividido em cinco níveis e cada um abrange as seguintes atividades.

Nível de Maturidade 1: Inicial.

Os processos são informais e caóticos isto significa que a organização normalmente não possui um ambiente estável e o seu sucesso depende da competência e do heroísmo das pessoas e não do uso de processos comprovados.

Apesar deste ambiente informal e caótico, as organizações de nível 1 de maturidade produzem muitas vezes produtos e serviços que funcionam só que frequentemente os orçamentos e o cronograma de seus projetos são excedidos.

Nível de Maturidade 2: Gerenciado.

Os projetos da organização asseguraram que os requisitos são gerenciados e que os processos são planejados, executados, medidos e controlados.

Nesse nível é preciso trabalhar pontos como a Gestão de Requisitos, Acordo com os Fornecedores, Acordo de Configuração, Planejamento e Monitoramento de Projetos, Medição e Análise.

Nível de Maturidade 3: Definido.

Os processos são bem caracterizados e possuem um bom entendimento, tais processos estão descritos em padrões, procedimentos, ferramentas e métodos.

Neste nível os requisitos a serem trabalhados são: Solução Técnica, Integração do Produto, Verificação, Validação, Definição de Processos Organizacionais, Gestão de Riscos, Análise de Decisões e Resolução.

Nível de Maturidade 4: Gerenciado Quantitativamente

A partir deste nível os subprocessos são selecionados e controlados utilizando-se de estatísticas e outras técnicas quantitativas contribuindo assim de forma significativa para o desempenho geral dos processos.

Quanto aos níveis 4 e 5, respectivamente, são trabalhados pontos como Performance de Processos Organizacionais, Gestão Quantitativa de Projetos, Inovação, Análise de Causas e Resolução.

Nível de Maturidade 5: Otimizado

Os processos são melhorados de forma contínua com base em um entendimento quantitativo das causas comuns de variações que são inerentes aos processos.

Após uma análise inicial dos processos da empresa ConstruArt foi constatado que os seus processos se encontram compatíveis com o do nível 1 de maturidade de forma que os produtos são entregues, mas como os riscos e o orçamento não são bem planejados ocorrem atrasos e custos adicionais que prejudicam o crescimento da empresa e como uma forma de solucionar este problema chegou - se à conclusão de que o nível de maturidade desejável para a empresa é o Nível 4, sendo assim os níveis 2 e 3, respectivamente, devem ser implementados. Pois desta forma espera-se trazer uma melhora na organização, qualidade, controle de custos e flexibilidade da empresa.

3. Linguagens Orientada a Objetos e Padrões de Projetos.

Em Engenharia de Software os padrões de projeto ou design pattern são definidos como uma solução geral para um determinado problema que ocorre com certa frequência em um projeto de software, sendo assim um padrão de projeto não corresponde a um projeto finalizado que pode ser diretamente transformado em um código fonte ou de máquina, pois ele é uma descrição ou um modelo de como resolver um problema e por isso pode ser usado em muitas situações diferentes.

As características, obrigatórias, que devem ser atendidas por um padrão de projeto é composto basicamente por quatro elementos e são eles:

- Nome do padrão;
- Problema a ser resolvido;
- Solução dada pelo padrão;
- Consequências.

Estas características também estabelecem um vocabulário comum ao projeto e facilitam a comunicação, a documentação e o aprendizado dos sistemas de software.

Quanto aos padrões principais temos a classificação dos padrões GOF (Gang of Four) que são divididos em 24 tipos e em função desta grande quantidade de padrões eles foram classificados com base nas suas finalidades, sendo assim eles são divididos em padrões de criação, estrutura e comportamento.

- Padrões de Criação - São aqueles que abstraem ou adiam o processo de criação dos objetos e tornam-se mais importantes à medida que os sistemas evoluem no sentido de dependerem mais da composição dos objetos do que da herança de classes.

  O desenvolvimento baseado na composição de objetos possibilita que os objetos sejam compostos sem a necessidade de que o seu interior seja exposto como acontece na herança de classe, o que possibilita a definição dos comportamentos de forma dinâmica e a ênfase desloca-se da codificação para a definição de um conjunto menor de comportamentos.
- Padrões Estruturais - Os padrões estruturais se preocupam com a forma como as classes e objetos são compostos para darem origem a estruturas maiores.

Enquanto que a composição de classes utiliza- se da herança para compor interfaces ou implementações, a composição de objetos é usada para descrever formas de se organizá-los no projeto com a finalidade de que se obtenham novas funcionalidades.

A capacidade de mudar a composição em tempo de execução obtida pela composição de objetos provê uma flexibilidade que não é possível ao utilizar a composição estática (herança de classes).

- Padrões Comportamentais - Os padrões comportamentais não descrevem apenas os padrões de objetos ou de classes, mas também os padrões de comunicação entre os objetos e se concentram nos algoritmos e na atribuição de responsabilidades entre os objetos.

  Enquanto os padrões comportamentais de classes utilizam a herança como forma de distribuir o comportamento entre as classes, os padrões de comportamento de objeto utilizam- se da composição de objetos em contrapartida a herança.

Os Benefícios ao usar os Padrões de Projetos.

Como eles são modelos que já foram utilizados e testados anteriormente, os padrões de projeto podem representar um bom ganho de produtividade para os desenvolvedores.

Seu uso também contribui para a organização e a manutenção de projetos, pois esses padrões se baseiam no baixo acoplamento entre as classes e na padronização do código. Além disso, a padronização dos seus termos faz com que as discussões técnicas sejam facilitadas, pois é mais fácil falar o nome de um design patterns em vez de ter que explicar todo o seu comportamento.

Os Principais Padrões GOF referentes a sua classificação:

Padrões de Criação:

- Abstract Factory;
- Object pool;
- Builder;
- Factory Method;
- Prototype;
- Singleton.

Padrões Estruturais:

- ❏ Private class data;
- ❏ Adapter;
- ❏ Bridge;
- ❏ Composite;
- ❏ Decorator;
- ❏ Façade ou Facade;
- ❏ Business Delegate;
- ❏ Flyweight;
- ❏ Proxy.

Padrões Comportamentais:

- ❏ Chain of Responsibility;
- ❏ Command;
- ❏ Interpreter;
- ❏ Iterator;
- ❏ Mediator;
- ❏ Memento;
- ❏ Observer;
- ❏ State;
- ❏ Strategy;
- ❏ Template Method;
- ❏ Visitor.

Apesar dos Padrões do GoF serem os mais conhecidos e usados eles não são os únicos que existem, pois uma série de outros padrões são catalogados o que acontece é que devido a evolução das linguagens de programação e a busca por novos padrões que atendam melhor a um determinado cenário é bastante comum o desuso de outros padrões.

Alguns dos Padrões de Projeto GOF:

Abstract Factory.

A intenção deste padrão é de fornecer uma interface para a criação de famílias de objetos relacionados ou dependentes sem que suas classes concretas sejam especificadas.

Se houvesse a necessidade de em uma aplicação que fosse implementada uma operação que oferece suporte a outras plataformas com características distintas uma maneira de constituí-la, seria definir uma família de componentes para cada plataforma em conjunto com uma fábrica que os instâncias que se alteraria de acordo com a plataforma alvo em que a aplicação estivesse usando. Ao ser executada desta forma a aplicação conheceria apenas uma interface e a implementação concreta seria conhecida somente em tempo de execução ou de compilação.

O Uso do padrão Abstract Factory deve estar restrito às seguintes situações:

- Um sistema deve ser independente com relação a forma de como os seus produtos são: criados, compostos ou representados;
- Um sistema deve ser configurado como um produto de uma família de múltiplos produtos;
- Se uma família de objetos for projetada para ser usada em conjunto deve-se garantir esta restrição.

Benefícios e Desvantagens:

- Isola as classes concretas;
- Facilita a troca de famílias de produtos;
- Promove a harmonia entre produtos;
- Difícil de suportar novos tipos de produtos.

Factory Method

Tem como objetivo definir uma interface para criar objetos e deixar que as subclasses decidam qual classe deve ser instanciada.

Este padrão é utilizado tanto por projetistas de software quanto por aqueles que trabalham no desenvolvimento de frameworks, pois ambos podem ter a necessidade de encapsular a criação de uma classe isolando-a do conhecimento da classe concreta da aplicação cliente através de uma interface.

Dessa forma as aplicações cliente implementam as funcionalidades esperadas pelo framework adicionando a lógica de negócio da aplicação, sem que o framework tenha o conhecimento de qual lógica foi usada pela aplicação para complementá-lo e de como ela foi feita.

A Utilização do Padrão Factory Method é necessário quando:

- ❏ Uma classe não pode antecipar a classe ou o tipo de objetos que se deve criar;
- ❏ Uma classe deixa explícito que as suas subclasses tem o conhecimento dos objetos que elas criam;
- ❏ Se tem uma classe que delega responsabilidade para uma dentre várias subclasses auxiliares, e se quer obter o conhecimento de qual subclasse auxiliar é a delegada.

Singleton

O Seu objetivo é garantir que um objeto terá apenas uma única instância, ou seja, garantir que uma classe irá gerar apenas um objeto e que este estará disponível de forma única para todo o escopo de uma aplicação.

Algumas aplicações possuem a necessidade de controlar o número de instâncias criadas de algumas classes e isto se dá pela necessidade da própria lógica, por motivos de performance ou pela economia de recursos.

O Padrão Singleton deve ser usado quando se tem a necessidade de:

- ❏ Manter num sistema, seja ele distribuído ou não, apenas uma instância de objeto e que o ponto de acesso para este objeto seja bem conhecido;
- ❏ Quando uma única instância tiver de ser extensível através de subclasses, possibilitando aos clientes usarem a instância estendida sem alterar o seu código (visões polimórficas).

Os Benefícios que o Padrão Singleton apresenta são:

- ❏ Acesso controlado a uma instância única;
- ❏ Espaço de nomes reduzido;
- ❏ Permite um refinamento de operações e de representações;
- ❏ Permite um número variável de instâncias, pois o padrão possibilita que seja adotada a estratégia de criar mais de uma instância da classe, de forma controlada;
- ❏ Mais flexível do que as operações de classe.

As Principais Linguagens de Programação Orientada a Objetos:

- ❏ Java;
- ❏ C++;
- ❏ C# ;
- ❏ Python;
- ❏ JavaScript;
- ❏ PHP;
- ❏ Ruby;
- ❏ Objective C;
- ❏ TypeScript.

3.1 Descrição das Linguagens Orientada a Objetos.

O Java é a linguagem base para o desenvolvimento de aplicações mobile, tendo o seu uso voltado para sistemas operacionais, dispositivos móveis, mainframes entre outros e como a sua sintaxe é similar com a de linguagens de programação mais antigas como, por exemplo C e C + + o Java proporciona um fácil entendimento para aqueles programadores que já usaram tais linguagens.

O Java também é uma das principais representantes das linguagens orientadas a objetos e tem como uma de suas características a Portabilidade, na qual a compilação do código fonte cria um código executável que, por sua vez, será interpretado por uma máquina virtual e essa máquina funciona como um intermediário entre o código e a plataforma em que

esse código será executado e isso faz com que a aplicação seja executada em diferentes plataformas proveniente da Filosofia WORA - Write Once, Run Anywhere (Escreva uma vez, execute em qualquer lugar).

A Linguagem de Programação C + + é baseada na linguagem C e foi criada na década de 80 por Bjarne Stroustrup, inicialmente o seu nome seria “C com classes", no entanto três anos mais tarde passou a ser chamada de C + +.

Stroustrup implementou diversas melhorias em sua linguagem , incluindo alguns recursos de orientação a objetos que a deixavam mais poderosa e capaz de resolver problemas ainda mais complexos e até hoje ela continua em constante evolução.

Tendo o seu uso divididos nos mais diversos tipos de aplicações como, por exemplo jogos, editores de texto, editores de imagem e entre outros a linguagem C++ possui como principais características o fato de ser multi-paradigma , possuir compatibilidade com a linguagem C, portabilidade e garantir uma boa performance.

É uma linguagem orientada a objetos, cuja sintaxe foi baseada nas precursoras C + +, Java e Object Pascal foi desenvolvida pela Microsoft e lançada em julho de 2002.

A linguagem C# é um dos recursos da plataforma .NET, e foi criada com o objetivo de melhorar a comunicação entre as diferentes tecnologias utilizadas pela empresa.

Ela pode ser utilizada na criação de diversos tipos de aplicações, pois como é focada em soluções de alto nível o seu sucesso está relacionado fortemente a sua constante evolução em conjunto com o leque de recursos que a tecnologia oferece aumentando assim a

produtividade no processo de desenvolvimento.

Principais características:

- ❏ Suporte à Orientação a Objetos;
- ❏ Uso do Conceito de Máquina Virtual (assim como Java);
- ❏ Portabilidade;
- ❏ Sintaxe Simples e de Fácil Compreensão.

Criada no início dos anos 90, Python é uma linguagem de programação desenvolvida e distribuída pela Python Software Foundation.

Utilizada para diversos fins na programação, o Python é considerado uma linguagem de altíssimo nível e suporta diferentes paradigmas de programação contando assim com recursos poderosos.

Uma das principais características dessa linguagem é a legibilidade do código, pois como a linguagem possui uma sintaxe moderna e clara favorece a escrita de códigos mais organizados, fáceis de se compreender e manter de forma que não se é perdido a produtividade.

Outras propriedades do Python :

- ❏ Suporte a Múltiplos Paradigmas de Programação;
- ❏ Desenvolvimento Comunitário.

Muita gente confunde JavaScript com Java, essa confusão se dá não só pela semelhança dos nomes, mas também pelo fato das suas sintaxes serem parecidas, mas isso se dá pelo fato de que ambas foram baseadas na linguagem C.

O JavaScript é uma linguagem voltada para os navegadores com o objetivo de proporcionar uma maior interatividade às páginas web e como é Suportada, hoje, por todos os navegadores essa linguagem é a mais popular do mercado. Com ela é possível trabalhar com os elementos no Client-Side (lado do cliente), alterar a estrutura de um documento HTML, modificar estilos CSS, realizar ações conforme a interação do usuário com a sua aplicação, realizar validações de formulários e etc.

Em outras palavras, o JavaScript é o responsável por trazer vida a uma página web no lado Client-Side contudo atualmente graças às constantes evoluções das linguagens de programação, a tecnologia do JavaScript, em alguns casos, também pode ser utilizada no lado Server-Side (lado do servidor) e em aplicações mobile tornando o Javascript uma das linguagens mais versáteis existentes.

O PHP é uma linguagem de programação de livre distribuição, utilizada em todo o mundo para criação de sistemas web dinâmicos.

Os códigos PHP são interpretados no servidor, logo trata-se de uma linguagem server-side (lado do servidor) e sempre que o navegador faz o seu requerimento, o interpretador processa o código da página e gera um HTML, que será enviado como resposta ao cliente.

Como o PHP possui o código aberto ele é liberado para a comunidade de programadores que trabalham na sua evolução e podem realizar consultas dos problemas resolvidos anteriormente.

As Principais Características do PHP São:

- Suporte tanto para programação estruturada, quanto para programação orientada a objetos.
- Fácil aprendizado, sendo necessário saber o HTML antes;
- Boa performance;
- Portabilidade.

Criado em 1995 pelo programador japonês Yukihiro Matsumoto (mais conhecido como Matz), inspirada em linguagens como Perl, LISP e SmallTalk, o Ruby é uma linguagem orientada a objetos e possui uma sintaxe simples.

A proposta de Matz era desenvolver uma linguagem que fosse legível, de fácil entendimento e agradável essa linguagem de programação é usada principalmente no desenvolvimento de aplicações web e tem como suas características principais: a Linguagem interpretada, ser Multiplataforma, Produtividade e o Código aberto (open source) que é mantido por uma comunidade ativa de desenvolvedores em todo o mundo.

Trata-se de uma linguagem de programação orientada a objetos multiplataforma baseada em SmallTalk e C criada pelos cientistas da computação Brad Cox e Tom Love no

início dos anos 80 e a ideia era criar uma linguagem de programação que primasse pela reutilização de código.

O Objective-C que hoje pertence à Apple é utilizado no desenvolvimento de aplicações para o sistema iOS e apesar da companhia ser a criadora da linguagem Swift que é baseada em Objective-C, a intenção, segundo a própria Apple, não é que uma linguagem venha a substituir a outra, mas sim fazer com que ambas sejam capazes de coexistir, sendo assim, aplicativos desenvolvidos em Swift podem ter partes do código escritas em Objective-C e vice-versa.

Apesar dessa filosofia apresentada pela Apple, o Objective-C encontra-se em declínio, devido a ascensão da linguagem Swift, porém a linguagem ainda apresenta boas colocações nos rankings de popularidade mundial.

Criada pela Microsoft, o TypeScript está provando ser uma escolha comum entre os desenvolvedores ASP.NET.

O TypeScript não necessariamente se trata de uma linguagem completamente nova, mas sim um superset (ou superconjunto) do JavaScript se com ele dispomos de recursos que suportam melhor o uso da Programação Orientada a Objetos.

A programação orientada a objetos (POO) sempre foi um problema ao ser aplicada em JavaScript, devido a sua sintaxe não permitir escrever classes, por exemplo, de forma tão clara, além da fraca tipagem de dados, sendo assim o TypeScript apresenta uma maneira de corrigir ou contornar esses problemas, adicionando funcionalidades que quando compiladas resultarão novamente em um código JavaScript.

4.Frameworks.

4.1. O que é um Framework ?

Os Frameworks são templates que possuem diversas funções e podem ser usados pelo desenvolvedor e eles auxiliam em um gerenciamento mais ágil dos projetos. Eles também são

os responsáveis por “tomar conta” da solução criada, pois os mesmos são responsáveis por gerar todo o fluxo de controle da aplicação fazendo com que não se tenha a necessidade de ficar reescrevendo os códigos e com isso o foco possa ser direcionado para a resolução dos problemas e nos esforços para alcançar os objetivos requeridos.

As Bibliotecas de Classes nada mais são do que uma implementação em que as funções podem ser importadas para serem usadas em diversos projetos e a grande diferença entre elas e os frameworks é a integração de suas funções, já que nas bibliotecas de classes as funções operam de forma, relativamente, independente, porém em um framework há relações já embutidas, ou seja, existentes de dependência entre os componentes. Também é importante considerar a possibilidade dos frameworks existentes não suprirem todas as necessidades das aplicações ou em certos casos apresentarem uma complexidade muito maior do que as soluções necessitam, sendo assim alguns devem ser tomados ao se utilizar esta tecnologia, são eles:

- ❑ Escolher um profissional responsável por editar o código desenvolvido;
- ❑ Definir quantas pessoas utilizarão o código e uma equipe responsável na detecção de bugs;
- ❑ Como as correção desses Bugs será feita e quanto isso vai custar;
- ❑ Programar as atualizações das aplicações que utilizam esse framework e definir o intervalo de tempo entre elas;
- ❑ Montar um cronograma para realizar a aplicação de funcionalidades;
- ❑ Definir como os frameworks serão documentados.

Os Principais Modelos de Frameworks.

Zend - É um dos frameworks mais atualizados e consistentes do mercado, mas não é um modelo simples, de forma que tende a ser mais indicado para os projetos mais robustos.

Laravel - Lançado em 2011, ele tem se tornado extremamente popular ao longo dos últimos anos, sendo considerado um dos frameworks PHP mais utilizados no mercado.
Ele se adapta aos mais diferentes tipos de projetos, dos mais simples aos mais robustos, uma vez que existem diversos tutoriais e vídeos que ajudam e orientam no aprendizado e na utilização desse framework.

Symfony - Esse modelo foi lançado em 2005 e foi arquitetado justamente para trabalhar de forma colaborativa com outras metodologias ágeis de desenvolvimento de soluções.
Fazendo uso da arquitetura MVC, o Symfony foca essencialmente em regras de negócio da aplicação e ,normalmente, é indicado para trabalhos de grande escala e de maior robustez.

Phalcon - Criado em 2012 ,o Phalcon tem crescido de forma exponencial na área de desenvolvimento e isso traz uma vantagem importante a sua comunidade, que auxilia na descoberta de erros e bugs disponíveis no código do framework e assim também ajuda a tirar dúvidas dos demais programadores.
Outro ponto importante é a facilidade em se ter versões traduzidas para os diversos idiomas logo depois de serem lançadas, já que é a própria comunidade que realiza esse trabalho.
Uma das suas principais vantagens é o fato de ser escrito em linguagem C característica que o diferencia dos demais frameworks que em sua maioria utiliza-se da linguagem PHP.

CakePHP - Lançado em 2005, o seu objetivo é simplificar o processo de desenvolvimento para aqueles que utilizam a linguagem PHP e com isso ele facilita o trabalho dos iniciantes e dos mais avançados.
Ele também contém mecanismos para trabalhar com JavaScript, Ajax, entre outros, se tornando bastante útil.
O fato dele trabalhar em cima da arquitetura MVC traz a possibilidade de se criar projetos de diversos portes

Bootstrap - Framework de código aberto, o Bootstrap permite que desenvolvedores que utilizam HTML, CSS e JavaScript e que trabalham , principalmente, com o design de uma aplicação, desenvolvam um aspecto visual mais bonito, intuitivo e também padronizado.
Uma das suas principais vantagens está ligada à responsividade, afinal é ele que dá às telas e aos elementos que as formam, a capacidade de se adequar ao tamanho do dispositivo do usuário, ou seja, o layout da aplicação se adapta a qualquer dispositivo, seja ele um celular, tablet ou um monitor.

Ionic - O Ionic é um completo (SDK — Software Development Kits) que em portugues significa Kit de Desenvolvimento de Software e também possui o seu código aberto.

Ele é muito utilizado para o desenvolvimento de aplicativos móveis híbridos devido a sua estrutura fornece várias ferramentas e recursos de desenvolvimento baseadas em tecnologias da Web, tais como a linguagem de marcação de hipertexto (HTML—HyperText Markup Language), CSS e o JavaScript.
O seu diferencial é a ferramenta de construção de interface que é feita no modo de arrastar e soltar, tornando assim o projeto muito mais intuitivo.

Angular - O Angular é um framework criado especialmente para auxiliar na interação do front com o back end.
Ele é muito utilizado em projetos de página única para possibilitar a comunicação entre o computador local e o servidor. Este recurso permite que muitas interações e tarefas sejam realizadas diretamente na máquina do usuário e isso poupa o processamento e consequentemente desafoga o link de internet.

Spring Boot - O Spring Boot é a evolução do Spring, o que deixa a sua estrutura um pouco complexa já que para definir um sistema, em vez de ser escrito diversos mini arquivos XML, o desenvolvedor precisa atuar direto nas anotações realizadas dentro do código-fonte.
As principais vantagens desse framework envolve o fato de que a estrutura já define uma série de convenções de desenvolvimento como por exemplo, como os objetos são nomeados e como os mesmos são organizados na arquitetura. Esse sistema também busca disponibilizar aos usuários informações armazenadas no seu banco de dados ou no back-end, para facilitar a criação de aplicações voltadas aos dispositivos móveis e os computadores.

Cordova - É um framework com o objetivo de simplificar e padronizar o desenvolvimento de aplicações híbridas para mobile.
Em geral, sua principal função é traduzir para sistemas operacionais como iOS e Android, linguagens como o HTML5 para que a aplicação possa funcionar da mesma forma independente do dispositivo que seja usado.
Também é possível encontrar bibliotecas de códigos prontos no site do Cordova, o que é outro ponto positivo, pois, apesar das particularidades de cada marca e modelo de dispositivo, existem funcionalidades e padrões comuns, o que torna o acesso em diversos aparelhos padronizado.

React - O React é uma biblioteca de JavaScript muito utilizada pelos desenvolvedores para criar interfaces de usuário. Ele permite criar aplicações de grande porte para diversas finalidades e oferece flexibilidade para fazer alterações ao longo do tempo de maneira simplificada. O objetivo do React envolve especialmente entregar velocidade, simplicidade e escalabilidade à produção de aplicações.

Pure - Como ele trabalha com as linguagens HTML e CSS faz com que as suas aplicações se tornem mais leves de fácil acesso aos desenvolvedores.
O Pure também possui características de responsividade que permite a sua customização conforme o projeto exigir e conta como característica principal o desenvolvimento de forma modular, que permite ao projeto em desenvolvimento a importação somente da parte do código necessário para aquele momento.

Material Design - O Material Design é uma estrutura de código aberto lançada pelo Google a partir das funcionalidades do Bootstrap.
O objetivo desse Framework é desenvolver um padrão visual do Google e desse jeito fazer com que as páginas consigam obter uma melhor compreensão para os mecanismos do buscador e se tornem mais limpas, sendo assim o desenvolvedor que está habituado a trabalhar com as ferramentas produzidas pela empresa terá uma facilidade em trabalhar com esse framework.

Flutter - O Flutter é um framework de UI (User Interface) desenvolvido também pelo Google voltado para dispositivos móveis que trabalham com os códigos open-source (Código aberto) preexistentes. Este Framework tem como objetivo criar interfaces em multiplataformas nativas para os sistemas operacionais iOS e Android.
No entanto, para se criar aplicações nele é necessário conhecer um pouco mais sobre a linguagem de programação Dart já que a ideia desse framework é entregar soluções Ahead of Time - AOT (antes do tempo).
Como este modelo utiliza-se do pacote Skia para a renderização de imagens 2D o carregamento de aplicativos, jogos e animações ocorre de maneira mais rápida e fluida e com isso a usabilidade e a experiência final do usuário é melhorada.

4.2 Sua importância para os projetos de sistemas orientados a objetos.

Devido a sua capacidade de reutilização de códigos já testados e com sua eficácia comprovada o benefício principal de se utilizar os Frameworks está ligado a capacidade que eles têm de economizar tempo no desenvolvimento de softwares outro fator é o suporte e apoio trazido aos projetos que utilizam os conceitos da orientação a objeto, como a abstração, o polimorfismo e a herança já que a maioria das linguagens de programação utilizadas, atualmente, por eles suportam o uso do paradigma orientado a objeto.

No entanto o processo de construção de um framework é muito mais complexo do que de uma aplicação tradicional, devido à flexibilidade inerente e a capacidade de variação de um framework, sendo assim o caminho até a sua reutilização de fato pode ser árduo, mas os benefícios e vantagens trazidas por ele justificam e alguns desses benefícios são:

- Menos Bugs - Como os códigos de um Framework já passaram por diversos testes, geralmente, eles já estão sem bugs graves, além disso as comunidades ajudam a reportar e corrigir estes erros.

  Ao aplicar uma estrutura como esta no projeto ocorre uma menor preocupação com erros de implementação e naturalmente isto representa um aumento na qualidade dos códigos do sistema;
- Facilidade de Aprendizado - Como a maioria dos frameworks tem um registro extenso de documentação ocorre a facilidade de aprendizado por parte dos desenvolvedores e esse material ajuda também no conhecimento das funções e na melhor forma de usá-las;
- Padronização de Código - Para que haja uma compatibilidade, o desenvolvedor deve seguir o mesmo padrão de codificação usado pelo framework este ato contribui para que o código se torne mais legível ajudando também na sua manutenção e garantido que todos os desenvolvedores que estejam envolvidos no projeto utilizarão o mesmo padrão;
- Redução de Custos - Uma das principais vantagens de se adotar os frameworks em projetos de desenvolvimento de software é a redução significativa de custos em relação ao tempo de produção.

  Como as bases são providas pela ferramenta, a equipe só precisará se concentrar na camada de negócio, o que facilita o desenvolvimento de software e diminui o tempo

das entregas;

- ❏ Maior Consistência das Aplicações - Um dos problemas comuns da produção de software é a falta de consistência de algumas partes do projeto em relação a outras.
  Com o padrão de uso exigido pelos frameworks se tem a garantia de que a aplicação terá menos falhas e isto permite que os desenvolvedores se concentrem naquilo que realmente importa e desse modo as regras de negócio, os requisitos apresentados pelos clientes e a satisfação deles com relação ao sistema passam a ser prioridade;
- ❏ Incentivo ao Conhecimento - Quanto mais se trabalha com os frameworks mais conhecimento é adquirido sobre o seu funcionamento e como estas estruturas estão em constante evolução, o aprendizado se torna algo natural e constante.

4.3 Exemplos Existentes e do interesse da ConstruArt.

Ao utilizar os frameworks a empresa de construção civil , ConstruArt pode melhorar suas aplicações independente do foco delas serem Front-End ou Back-End de modo que a consistência, maleabilidade e a facilidade de uso possam ser geridas de maneira quase que automatizada resultando em melhorias na qualidade dos seus produtos ou serviços e para que isso ocorra a empresa pode utilizar os seguintes Frameworks:

JavaScript - Front-End

- ❏ Angular;
- ❏ Vue;
- ❏ Ember.js;
- ❏ React (que é considerado um framework dentro de um ecossistema de React).

JavaScript - Back-end

- ❏ Express;
- ❏ AdonisJs.

CSS

- ❏ Bootstrap;
- ❏ Materialize.

PHP - Back-End

- ❏ Laravel;
- ❏ CodeIgniter;
- ❏ Symfony;
- ❏ CakePHP.

Java

- ❏ Spring;
- ❏ Python;
- ❏ Django;
- ❏ Ruby;
- ❏ RubyOnRails.

5. Arquitetura de Software.

A Arquitetura de software consiste na definição dos componentes que compõem um software, suas propriedades externas e na forma como este software se relaciona com outros softwares e como parte da arquitetura de software se tem o projeto de arquitetura de software que consiste na compreensão de como um sistema deve ser organizado e como a sua estrutura geral deve ser.

Outro aliado importante para a construção de uma boa arquitetura é a documentação, pois com ela a comunicação entre os stakeholders é facilitada e as decisões iniciais acerca do projeto de alto-nível são registradas, outra coisa que a documentação possibilita é o reuso dos padrões entre projetos.

5.1 Arquitetura Padrão ConstruArt.

Após uma análise na empresa ConstruArt chegou-se à conclusão de dois dois tipos de arquiteturas que podem ser usadas pela empresa são elas: Modelo em três Camadas e Modelo Orientado a Serviço.

Modelo em Três Camadas.

Esta estruturação objetiva facilitar a alocação das funcionalidades aos componentes e tem como base a mesma usada no Modelo de N Camadas (Duas Camadas), porém mais uma camada é acionada.

Cada camada possui uma função bem definida, são elas:

- Camada de Apresentação - Nela se encontram as telas ou formulários que interagem diretamente com o usuário e através dela requisições como cadastros, consultas e etc. São realizadas;
- Camada de Negócio ou Lógica do Negócio - Nela se encontram as regras de negócio e funções, não existindo nesta camada uma interface que se conecte diretamente com os usuários;
- Camada de Dados - É a responsável pela persistência dos dados é nela que os dados da camada de negócio são recebidos e salvos em um banco de dados.

O uso deste modelo oferece suporte à flexibilidade e a portabilidade, o que resulta em facilidade de manutenção , pois cada camada deve ter um certo grau de independência uma da outra, dessa forma as atualizações ou correções feitas em uma camada não afetará a outra.

No entanto sua desvantagem é que quanto mais camadas um sistema possuir mais lento será o seu desempenho uma tática para tentar compensar isto tem sido a redução do nível de acoplamento entre os componentes já que assim eles terão uma menor necessidade de comunicação entre si o que resultará num desempenho melhor.

Modelo Orientado a Serviço (SOA)

Esta arquitetura tem como objetivos principais integrar as aplicações, disponibilizar maior flexibilidade para mudanças e suportar serviços independentes de plataforma ou protocolos.

Um dos componentes mais importante em SOA é o Barramento de Serviços , ele não implementa a arquitetura, mas oferece as funcionalidades para implementá-la. Este barramento provê uma camada de abstração acima de um sistema de mensageria que permite a integração entre os aplicativos.

A vantagem desta estrutura em uma organização é justamente a flexibilidade que os

web services trazem, sendo considerados importantes para que sejam mantidas e cumpridas as regras de negócio e solucionar problemas. Em um ambiente, completamente, heterogêneo em que todo mundo precisa conversar entre si, por exemplo os Web Services podem trazer uma homogeneidade necessária ao negócio, pois qualquer linguagem é capaz de consumi-los.

Mais uma vantagem que SOA oferece é a garantia da extensibilidade afinal, se um dia a regra de negócio de validação precisar ser alterada ou até mesmo uma nova etapa do negócio precisar ser acrescentada, basta modificar o web service e todos os softwares que o consomem serão automaticamente "modificados".

Outra intenção ao se utilizar a arquitetura SOA é uma completa integração entre o setor de negócios e o setor de tecnologia, pois como as empresas precisam responder de forma efetiva e rápida ao mercado, as aplicações têm que ter flexibilidade em executar as mudanças de forma rápida.

Serviços.

Na prestação de algum serviços, existe um fornecedor que fornece algum tipo de serviço e o consumidor que consome o serviço fornecido na SOA não é diferente os serviços expõem seus membros através de um contrato de serviços no qual são definidas quais as operações que serão disponibilizadas e a interface técnica ,que corresponde às ligações entre um serviço e as aplicações que irão consumi-lo, expondo assim somente as operações desejadas pelo cliente ou consumidor.

Algumas Características dos Serviços:

- São reutilizáveis;
- Compartilham um contrato formal;
- Possuem um baixo acoplamento;
- Abstraem a lógica;
- São capazes de se comportarem;
- São autônomos;
- Evitam alocação de recursos por longos períodos;
- São capazes de serem descobertos.

Vantagens de se usar a Arquitetura Orientada a Serviço.

- ❏ Reutilização: O serviço pode ser reutilizado para outras aplicações;
- ❏ Produtividade: Com o reuso, a equipe de desenvolvimento pode reutilizar serviços em outros projetos e diminuir o tempo de desenvolvimento;
- ❏ Flexibilidade: Isolando a estrutura de um serviço as mudanças são feitas com maior facilidade;
- ❏ Manutenção: Com o baixo acoplamento, a manutenção dos serviços é facilitada;
- ❏ Alinhamento com o Negócio: A área de negócio consegue visualizar os processos alinhados com a tecnologia;
- ❏ Interoperabilidade: Disponibilizar os serviços independentemente da plataforma e tecnologia;
- ❏ Integração: A integração com outros serviços, aplicativos e sistemas legados.
- ❏ Governança: Gerenciamento nos processamentos do negócio;
- ❏ Padronizado: Baseado no uso de padrões;
- ❏ Abstração: Os Serviço são totalmente abstraídos das suas implementações.

Desvantagem:

- ❏ Complexidade: Uma grande quantidade de serviços precisa ser gerenciada;
- ❏ Performance: A performance depende tanto do servidor onde o serviço está publicado, como também da rede;
- ❏ Robustez: Caso uma exceção aconteça não tem como reverter o processo;
- ❏ Disponibilidade: Uma queda na rede ou no servidor deixa todos os serviços indisponíveis;
- ❏ Testabilidade: O debug no serviço é um problema para os desenvolvedores;
- ❏ Segurança: Os serviços estão disponíveis na rede e qualquer aplicativo pode consumir este serviço, sendo assim os dados que são trafegados pela rede podem ser interceptados.

6. Padrão MVC.

6.1 O que é o MVC ?

O MVC (Modelo, Visão e Controle) é um padrão de projeto de software ou de arquitetura de software formulado na década de 1970 com o foco no reuso de código e na separação dos conceitos em três camadas, interconectadas, onde a apresentação dos dados e a interação com os usuários (front-end) são separados dos métodos que interagem com o banco

de dados (back-end).

Mesmo tendo sido desenvolvida ,inicialmente, para computação pessoal, o MVC foi drasticamente se adaptando como uma arquitetura para as aplicações web de modo que muitos frameworks de aplicação comercial ou não foram desenvolvidos tendo este modelo como base, porém os mesmos frameworks variam em suas interpretações, principalmente no modo em que as responsabilidades MVC são separadas entre o cliente e servidor.

Com o aumento da complexidade inerente aos sistemas ou sites desenvolvidos, o MVC permite que o conceito chamado de Dividir para conquistar seja realizado. Este conceito consiste em pegar um grande problema e o dividir em problemas menores diminuindo assim a sua complexidade, dessa forma qualquer tipo de alteração feita em uma das camadas não irá interferir nas demais o que facilita a atualização de layouts, das regras de negócio e a adição de novos recursos.

6.2 Qual é a sua importância ?

O MVC ajuda a deixar o código mais manutenível, ou seja, a sua manutenção é facilitada, já que as responsabilidades são devidamente separadas o que também traz uma facilidade para a compreensão do código e para sua reutilização. Além disso, se tem um código mais testável, pois como ele é mais granular a realização de testes é feita de uma maneira muito mais rápida e eficiente sendo possível investigar a origem e causa do problema de acordo com as camadas e com suas respectivas funções.

Como no MVC a lógica de apresentação está separada em sua própria camada (Lógica e Física) a separação em camadas lógicas torna o sistema mais flexível e as suas partes podem ser alteradas de forma independente.

Quanto às funcionalidades da camada de negócio elas podem ser divididas em classes e estas classes podem ser agrupadas em pacotes ou componentes reduzindo assim as dependências entre elas. Além disso, os pacotes podem ser reutilizados por diferentes partes do aplicativo ou por aplicativos diferentes.

A modelagem orientada a objetos ajuda a promover a modularidade , pois os objetos encapsulam seus dados (propriedades ou estados) e oferecem funcionalidades através dos seus métodos e se forem projetados de forma adequada os objetos e as dependências entre si podem vir a ser reduzidas ficando assim fracamente acoplados o que torna mais fácil a sua manutenção e a sua evolução.

6.3 Quais são os papéis de suas camadas ?

View: Camada de Apresentação ou Visualização.

É por onde os dados solicitados do Model (Modelos) são exibidos e pode ser qualquer saída de representação dos dados, como uma tabela ou um diagrama, sendo possível se ter várias view do mesmo dado, como um gráfico de barras para gerenciamento e uma visão tabular para contadores.

O View é a camada de interface com o usuário utilizada para receber a entrada de dados e apresentar visualmente o resultado, sendo assim ele não se dedica em saber como o conhecimento foi obtido ou de onde ele foi retirado.

Model: Camada de Modelo ou da Lógica da Aplicação.

A Camada Model é a ponte entre as camadas View e Controller e consiste na parte lógica da aplicação, sendo responsável por gerenciar o comportamento dos dados através de regras de negócios, lógica e funções Portanto pode se dizer que o Model é o coração da execução.

O Model sabe o que o aplicativo quer fazer e a partir dos comandos da camada controller em um ou mais elementos de dados ele é capaz de analisar a condição do aplicativo e executar as instruções para mudá-las Já que ele tem acesso a toda e qualquer informação vinda de um banco de dados.

Ele modela os dados e os comportamentos por trás do processo de negócios se preocupando apenas com o armazenamento, manipulação e com a geração dos dados.

Controller: Camada de Controle ou Intermediária.

Sendo o componente final da tríade o controller faz a mediação da entrada e saída, comandando o view e o model para serem alterados de forma apropriada conforme o usuário solicitou através do mouse e teclado , sendo assim ele é o responsável por controlar todo o fluxo de informação que passa por um site ou sistema.

6.4 Os benefícios de se utilizar o Padrão MVC.

As Vantagens do Padrão MVC

- Como o Padrão MVC gerencia múltiplos views ao utilizar o mesmo Model é fácil

manter, testar e atualizar sistemas compostos;

- ❏ É muito simples adicionar novos clientes apenas incluindo seus views e controller;
- ❏ Torna a aplicação escalável;
- ❏ É possível ter desenvolvimento em paralelo para o model, view e controller, pois são independentes;
- ❏ Facilita o reuso do código;
- ❏ Melhor nível de sustentabilidade, pois facilita a manutenção da aplicação;
- ❏ Transformação da interface fácil sem que se tenha a necessidade de modificar a camada de negócio;
- ❏ Melhor desempenho e produtividade devido a estrutura de pacotes modulares;
- ❏ A arquitetura modular, que permite aos desenvolvedores e designers desenvolverem em paralelo;
- ❏ Partes da aplicação podem ser alteradas sem afetar outras.

As Desvantagens do Padrão MVC

- ❏ Necessita de um tempo maior para explorar e modelar o sistema;
- ❏ Requer mão-de-obra especializada;
- ❏ Não é aconselhável para pequenas aplicações;
- ❏ Conforme o tamanho e a complexidade do projeto crescem, a quantidade de arquivos e pastas também é expandida, sendo assim os interesses da interface do usuário (UI), que é constituída das camadas MVC, se localizam em várias pastas que não são formadas em grupos por ordem alfabética, o que pode causar erros se não forem manuseadas corretamente.

6.5 Sua utilização através de um exemplo..

Página Web, onde o usuário realizará o cadastro dos clientes.

Neste caso, provavelmente, deve-se ter uma classe chamada Cliente.php, onde as informações que se deseja guardar serão mostradas aos usuários, sendo assim esta classe será o Model do sistema.

Nesta camada também podem ser acoplados os aspectos de manipulação de bancos de dados através dos métodos inserir, alterar, excluir e listar os clientes a partir de uma tabela em um banco de dados.

A página HTML será o View, que mostrará o formulário para o cadastro de novos usuários entre outras funcionalidades implementadas no sistema, entre outras palavras é o meio pelo qual o usuário, que no caso é um funcionário da ConstruArt, irá interagir com o sistema em si.

O Controller fará então a mediação entre o Model e o View. Ele é necessário, pois as estruturas presentes no View não devem acessar diretamente o Model uma vez que isso pode criar um acoplamento entre as estruturas de apresentação e definição de negócio o que não é interessante, portanto uma Classe dentro do projeto PHP poderá ser então um controller, realizando a ligação entre as camadas Model e Views.

CONCLUSÃO

Ao Concluir as tarefas desta PTI (Produção Textual Interdisciplinar) referentes às disciplinas de: Engenharia e Projeto de Software, Projeto Orientado a Objeto e Programação Para Web II foi permitido o aprimoramento e a aplicação dos conhecimentos obtidos neste Quinto Semestre da graduação de Análise e Desenvolvimento de Sistemas a fim de que a visão, que até então era teórica, fosse mudada para uma concepção prática e desta forma durante a realização de cada tarefa pode se observar os seguintes resultados:

- Na Primeira atividade foi estabelecido a importância de se escolher uma metodologia ágil consistente com os negócios e os custos de colocá-la em prática assim como sua origem e características em conjunto foi escolhido um modelo de maturidade e a escolha do nível desejável para empresa ConstruArt, onde pode ser estudado sobre os impactos que a escolha destes modelos trazem aos negócios e como consequência ajudam no crescimento das empresas tornando-as mais competitivas;

- Na Segunda atividade foi visto os tipos de linguagens orientadas a objetos usados no mercado atual assim como suas características e aplicações. O mesmo foi feito com relação aos padrões de projetos, onde foi explicado os seus benefícios e os padrões mais comuns e a partir destes conhecimentos é possível fazer escolhas mais consistentes e análises mais precisas já que a união dos padrões de projeto com a implantação de uma linguagem de programação certa permitem a predição de resultados gerando assim uma maior consistência de dados para a empresa.

Nesta tarefa também foi estabelecido o que são os Frameworks e sua importância e com isso foi notado as diferenças entre eles e os padrões de projetos e como ambos são usados dentro de um projeto de software por último se tem a arquitetura de software, onde foi solicitado a apresentação de uma arquitetura padrão que se encaixasse no desenvolvimento do Sistema da ConstruArt, sendo assim foram deixado dois exemplos e após uma análise o melhor seja escolhido, com isso é mostrado como que a escolha de uma arquitetura de software contribui para um sistema mais organizado e como consequência facilita a sua manutenção;

- Na Terceira atividade foi mostrado um pouco sobre o que é o padrão MVC, sua importância e um exemplo do seu funcionamento, onde o papel de cada camada pode ser esclarecido assim como os benefícios ao adotar este padrão em algum projeto.

Ao conectar as atividades ou tarefas trazidas por esta PTI com todas as unidades vistas até agora no curso pode-se concluir que todo conhecimento obtido é de extrema importância e se torna indispensável para o desenvolvimento prático e eficaz da profissão.

REFERÊNCIAS

TreinaWeb - Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-mvc/> Acesso em 15 Abril. 2021.

Wikipédia - Disponível em: <https://pt.wikipedia.org/wiki/MVC> Acesso em 14 Abril. 2021.

Oficina Net - Disponível em: <https://www.oficinadanet.com.br/artigo/desenvolvimento/o_que_e_model-view-controller_mvc> Acesso em 14 Abril. 2021.

Devmedia - Disponível em: <https://www.devmedia.com.br/vantagens-e-desvantagens-de-soa/27437> Acesso em Abril. 2021.

Wikipédia - Disponível em: <https://pt.wikipedia.org/wiki/Arquitetura_de_software> Acesso em 06 Abril. 2021.

Hub - Disponível em: <https://decodehub.buzz/quais-principais-linguagens-e-frameworks-para-o-desenvolvimento-web/> Acesso em 02 Abril. 2021.

Pixlr - Disponível em: <https://pixlr.com/br/x/#home> Acesso em 03 Março. 2021.

Devmedia - Disponível em: <https://www.devmedia.com.br/frameworks-e-padroes-de-projeto/1111> Acesso em 31 Março. 2021.

Gaea - Disponível em: <https://gaea.com.br/entenda-o-que-e-framework/> Acesso em 29 Março. 2021.

CronApp - Disponível em: <https://blog.cronapp.io/frameworks-para-desenvolvimento-de-softwares/> Acesso em 29 Março. 2021.

HostGator - Disponível em: <https://www.hostgator.com.br/blog/frameworks-na-programacao/> Acesso em 29 Março. 2021.

Becode - Disponível em: <https://becode.com.br/principais-linguagens-de-programacao/> Acesso em 25 Março. 2021.

Devmedia - Disponível em: <https://www.devmedia.com.br/top-10-linguagens-de-programacao-mais-usadas-no-mercado/39635> Acesso em 25 Março. 2021.

Cp4us - Disponível em: <https://www.gp4us.com.br/modelos-de-maturidade/> Acesso em 2 Março. 2021.

Academy - Disponível em: <https://www.pmgacademy.com/cobit-modelos-maturidade/> Acesso em 21 Março. 2021.

Devmedia - Disponível em: <https://www.devmedia.com.br/cmmi-capability-maturity-model-integration/3530> Acesso em 21 Março. 2021.

Project.Builder - Disponível em: <https://www.projectbuilder.com.br/blog/quais-sao-os-principais-tipos-de-metodos-ageis/> Acesso em 18 Março. 2021.

Geek.Hunter - Disponível em: <https://blog.geekhunter.com.br/manifesto-agil/> Acesso em 16 Março. 2021.

Ciencia.Computaçao - Disponível em: <https://cienciacomputacao.com.br/tecnologia/o-que-foi-a-crise-do-software-e-o-inicio-da-engenharia-de-software/> Acesso em 16 Março. 2021.

IEBS - Disponível em: <https://www.iebschool.com/pt-br/blog/software-de-gestao/as-metodologias-ageis-mais-utilizadas-e-suas-vantagens-dentro-da-empresa/> Acesso em 16 Março. 2021.

Google Drive - Disponível em: <https://drive.google.com/drive/u/0/?tab=wo> Acesso em 02 Março. 2021.

Google - Disponível em: <https://www.google.com.br/> Acesso em 02 Março. 2021.