UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - CSHNB
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Anderson Arraes de Moraes Monte

PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE INFORMAÇÕES ESCOLARES

Picos, PI
2014

Anderson Arraes de Moraes Monte

PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE INFORMAÇÕES ESCOLARES

Trabalho de conclusão de curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal do Piauí como parte dos requisitos para obtenção do título de Bacharel em Sistemas de Informação, sob orientação do Prof. Esp. Francisco das Chagas Imperes Filho.

Picos, PI

2014

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

M775p Monte, Anderson Arraes de Moraes.
Protótipo de sistema de gerenciamento de informações escolares / Anderson Arraes de Moraes Monte. - 2014.
CD-ROM : il. ; 4 ¾ pol. (69 f.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí. Picos-PI, 2014.
Orientador(A): Prof. Esp. Francisco das Chagas Imperes Filho

1. Sistema Gerenciais. 2. Educação. 3. Rede Particular de Ensino. 4. Desenvolvimento Ágil. I. Título.

CDD 005.75

Anderson Arraes de Moraes Monte

PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE INFORMAÇÕES ESCOLARES

UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - CSHNB
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Trabalho apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal do Piauí como requisito para obtenção do título de Bacharel em Sistemas de Informação.

Data de Aprovação: 06 de janeiro de 2015.

Prof. Esp. Francisco das Chagas Imperes Filho
Universidade Federal do Piauí – UFPI

Prof. Esp. Dennis Sávio Martins da Silva
Universidade Federal do Piauí – UFPI

Prof. Esp. Allan Jheyson Ramos Gonçalves
Universidade Federal do Piauí - UFPI

AGRADECIMENTOS

Agradeço, a meu Pai, Antonio Arrais do Monte, e minha Mãe, Clara Lúcia de Moraes Monte, por sempre me apoiarem, compreenderem e darem base para seguir sempre em frente, a toda minha família e todos aqueles que me ajudaram direta ou indiretamente, em especial ao professor Francisco das Chagas Imperes Filho, que me orientou durante todo processo, a Mariana Rodrigues da Silva pela paciência durante esse período e a equipe da <body> Desenvolvimento pela força e ajuda para concluir este trabalho.

“O único lugar aonde o sucesso vem antes do trabalho é no dicionário.”

Albert Einstein

RESUMO

A necessidade de informatização de rotinas organizacionais está cada vez mais evidente, onde até mesmo instituições de menor porte se veem com a necessidade de buscar alternativas para adquirir sistemas que lhes deem suporte administrativo, auxiliando desta forma, no processo de tomada de decisão. Dentro deste contexto, surgiu a motivação para o desenvolvimento deste trabalho, que traz a pesquisa e o processo de implementação de um sistema para o gerenciamento de informações escolares, com foco no ensino fundamental, principalmente da rede particular de ensino. Este projeto descreve como se deu os levantamentos de requisitos, seguindo os principios básicos da Engenharia de *Software* (ES), seu desenvolvimento com a adoção da metodologia de desenvolvimento ágil *Scrum* e utilização de tecnologias como *HTML5*, *CSS3*, *Javascript*, *PHP* e Sistema Gerenciador de Banco de Dados (SGBD) *MySQL*, e ainda sobre os testes e manutenção que foram realizados em uma escola do município de Picos, no Piauí.

Palavras-chave: Sistemas Gerenciais. Educação. Rede Particular de Ensino. Desenvolvimento Ágil.

ABSTRACT

The need of computerization of organizational routines is increasingly evident, where even smaller institutions find themselves with the need to seek alternatives to acquire systems that give them administrative support, aiding thus, in the decision-making process. Within this context, came the motivation for development this work which brings search and the implementation process of a system for managing school information, focusing in elementary school, mainly the private schools. This project describes how was requirements gathering, following the basic principles the Software Engineering (SE), your development with the adoption the agile development methodology Scrum and using technologies such as HTML5, CSS3, JavaScript, PHP and Database Management System (DBMS) MySQL, and still on tests and maintenance were performed in a school the municipality of Picos, Piauí.

Keywords: Management Systems, Education, Private Network Education, Agile Development.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente a informática está presente em todas as áreas. No entanto, as empresas de menor porte sofrem para acompanhar essa evolução, principalmente no âmbito do gerenciamento de informações e consequentemente no suporte à tomada de decisão por parte dos gestores. Esse fator está intrinsecamente relacionado à grande parcela dos colégios que atuam no ensino fundamental da rede particular de ensino do município de Picos-PI, parcela esta, foco do presente trabalho de pesquisa.

1.1 Considerações Iniciais e Motivação

Muitos autores afirmam que estamos vivenciando a era da Tecnologia da Informação (TI), em razão dos sistemas estarem todos integrados, possibilitando a otimização dos processos e a diminuição da redundância de dados. Fatores estes que tornam possível melhorar o desempenho das empresas, pois os processos tornam-se mais estruturados, e minimiza o retrabalho (REZENDE, 2002 apud PIVA, 2010).

Este retrabalho foi bastante notado em nossa linha de pesquisa ao visualizar ações que todos os anos se repetem, podendo ser automatizadas, ao menos, parcialmente, e as TIs são alternativas que buscam formas de acelerar esses processos e evitar a duplicidade de dados, sendo a automatização uma possibilidade viável.

Com esse panorana nos surgiu a motivação para pesquisar mais sobre o assunto e produzir um sistema em que se encontrasse as principais funcionalidades atendidas por instituições de ensino, principalmente aquelas de menor porte, pois assim nos proporcionaria uma maior visão de toda a instituição.

1.2 Objetivo

O principal objetivo do projeto foi desenvolver um sistema *web* de gerenciamento de informações escolares que atendesse as necessidades básicas de um colégio do ensino fundamental, como: o controle de matriculas; cadastro de

professores e alunos; gerenciamento de notas e faltas; dentre outros, de modo que atingisse qualidade nos serviços oferecidos de maneira intuitiva, uma vez que Pressman (2011) diz que quando um *software* atende às exigências do usuário operando sem problemas por um longo tempo, torna-se bem-sucedido. E, por consequencia, deve ser fácil de modificar e, mais ainda de utilizar, sendo realmente capaz de transformar rotinas para melhor.

No entanto, para isso ocorrer é necessário toda uma metodologia, em especial a Engenharia de *Software* (ES), para que os objetivos sejam alcançados com qualidade, desempenho e produtividade.

1.3 Procedimentos Metodológicos

Em busca de melhores resultados foi realizado um levantamento bibliográfico sobre temas relacionados e então, destinou-se a por em prática os conhecimentos adquiridos principalmente nas disciplinas de planejamento, engenharia e qualidade de *software*, para desenvolver uma alternativa para instituições de ensino.

A inicialização dos trabalhos se deram por meio de observação das atividades corriqueiras do estabelecimento de ensino e alguma entrevistas para conhecer o ambiente e os funcionários que fazem parte dele para a partir dai poder iniciar um planejamento adequado para todo processo de desenvolvimento.

Foram buscados sistemas existentes e analisados para se ter uma melhor compreensão de como se dá os processos automatizados, a acessibilidade de cada funcao e principalmente, como as empresas disponibilizam esses sistemas para o publico.

1.4 Organização do Trabalho

O presente projeto está divido em mais 3 capítulos de conteúdo, sendo o Capítulo 2 o Referencial Teórico que traz uma berve descrição dos assuntos a serem abordados dentro de seus 4 subtópicos: 2.1 Engenharia de *Software*; 2.2 Banco de Dados; 2.3 Programação *Web* e 2.4 Trabalhos Relacionados.

Sistema de Gerenciamento Escolar é o Capítulo 3 deste trabalho é dedicado

ao projeto de criação do Protótipo do Sistema de Gerenciamento de Informações Escolares, levando em consideração o que foi mencionado pelos autores anteriormente citados e colocando em prática o que foi visto durante o curso. Ele também possui subtópico, sendo ele: 3.1 Desenvolvimento da Aplicação;

Ainda contamos com o Capítulo 4 Resultados, que discorre um pouco sobre a finalização do Projeto. O Capítulo seguinte destina-se à Conclusão do Trabalho, trazendo um pouco sobre os conhecimentos adiquiridos durante o projeto. Por fim, além dos capítulos mencionados, está disposto no trabalho as seções para Referências Bibliográficas e Apêndices.

2 REFERENCIAL TEÓRICO

Este capítulo nos apresenta alguns autores e o que seus trabalhos nos falam a respeito do tema abordado, nos concedendo embasamento teórico e fontes confiáveis. O capítulo nos traz ainda uma breve apresentação das tecnologias e ferramentas que foram utilizadas no projeto, além de alguns exemplos e comentários a respeito de outras aplicações de gerenciamento escolar.

2.1 Engenharia de *Software*

Filho (2001) explica que o *software* é um produto, e todo produto possui um ciclo de vida que se inicia na sua concepção, logo que o programador ou responsável pelo *software* colhe suas necessidades e passa por um desenvolvimento, que é a parte de codificação dos módulos e entregas ao cliente, seguindo logo após à parte de operação que é quando ele começa e permanece executando dentro de um plano de negócio e, por fim, é retirado de operação, quando sua utilização não é mais adequada para o empreendimento e o *software* chega ao fim de sua vida útil.

Ainda em relação ao ciclo de vida de um sistema, pode-se ter uma melhor compreensão através da Figura 1 que mostra um esquema simplificado do ciclo de vida de um *software*.

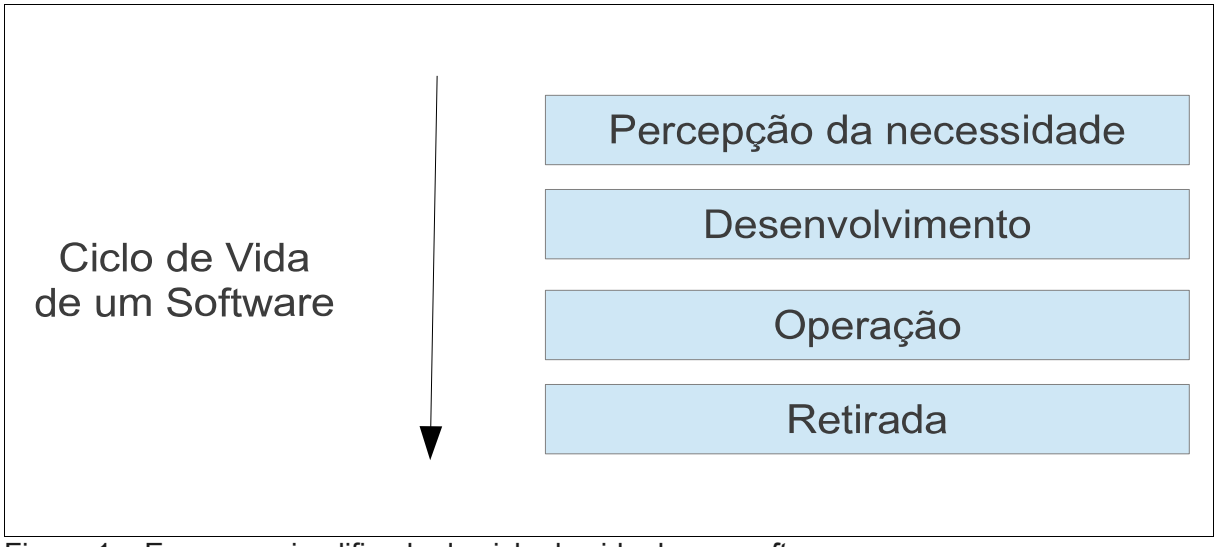


Figura 1 – Esquema simplificado do ciclo de vida de um *software*
Adaptado de: Tabela 2, Filho (2001).

Para entender o que vem a ser o desenvolvimento de um *software*, primeiro é preciso entender o que é *software.* Pressman (2011) afirma que *software* de computador é o produto desenvolvido por profissionais de *software* que abrange programas executáveis em um computador de qualquer porte ou arquitetura, conteúdos e informações abrangendo praticamente qualquer tipo de mídia eletrônica, sendo este pensamento confirmado por Da Rocha (2004) citada por Piva (2010), onde declara que desenvolvimento de sistemas significa o ato de elaborar e implementar um programa, entendendo as necessidades dos usuários e transformando-as em um produto denominado *software*.

Contudo, elaborar e implementar um *software*, exige muitos passos e não se inicia diretamente na codificação. Um *software* é parte de um processo, e esse processo precisa ser previamente planejado e idealizado, principalmente para resolver situações do dia a dia. Os processos de desenvolvimento são essenciais para o bom andamento do projeto de um *software*, assim como a compreensão do sistema como um todo, e “quanto mais aumenta a complexidade dos sistemas, mais difícil se torna a sua visibilidade e compreensão, portanto, sem um processo bem definido o projeto tem grande chance de insucesso” (PIVA, 2010, p.22).

No passado não existia muita preocupação sobre o processo de desenvolvimento, os *softwares* eram solicitados e desenvolvidos apenas com requisitos passados pelo cliente e muitas vezes não geravam um produto de qualidade.

> Quando um *software* não é especificado de forma correta, podem ocorrer vários problemas decorrentes desta falta de especificação, por exemplo: atraso na entrega, mudanças muito impactantes, um *software* que não atende as necessidades do cliente e um produto de baixa qualidade, e decorrente de todos estes erros o custo acaba se tornando muito alto fazendo com que o cliente ou a empresa de desenvolvimento tenha um grande prejuízo, ou porque o *software* não atende as reais necessidades ou porque o *software* demandou muito mais tempo do que o esperado (MANFIO, 2012, p.12).

Conforme já explanado, antes do surgimento da Engenharia de *Software*, todos os *softwares* eram desenvolvidos apenas com base em alguns requisitos determinados pelo cliente. Assim que estes requisitos eram identificados o desenvolvimento começava, porém muitos clientes não tem conhecimento em desenvolvimento de *softwares*, tornando várias vezes a interpretação desses requisitos, pelo desenvolvedor, algo muito diferente do que o cliente realmente

necessita, resultando em um *software* que não atende as necessidades do cliente (MANFIO, 2012).

O que pode ser notado ao examinar a Figura 2, que mostra a representação da discordancia de entendimento entre as partes participantes de um processo de criação de um *software*, que tem por consequência falhas devastadoras, gerando prejuizos e insatisfação do cliente.

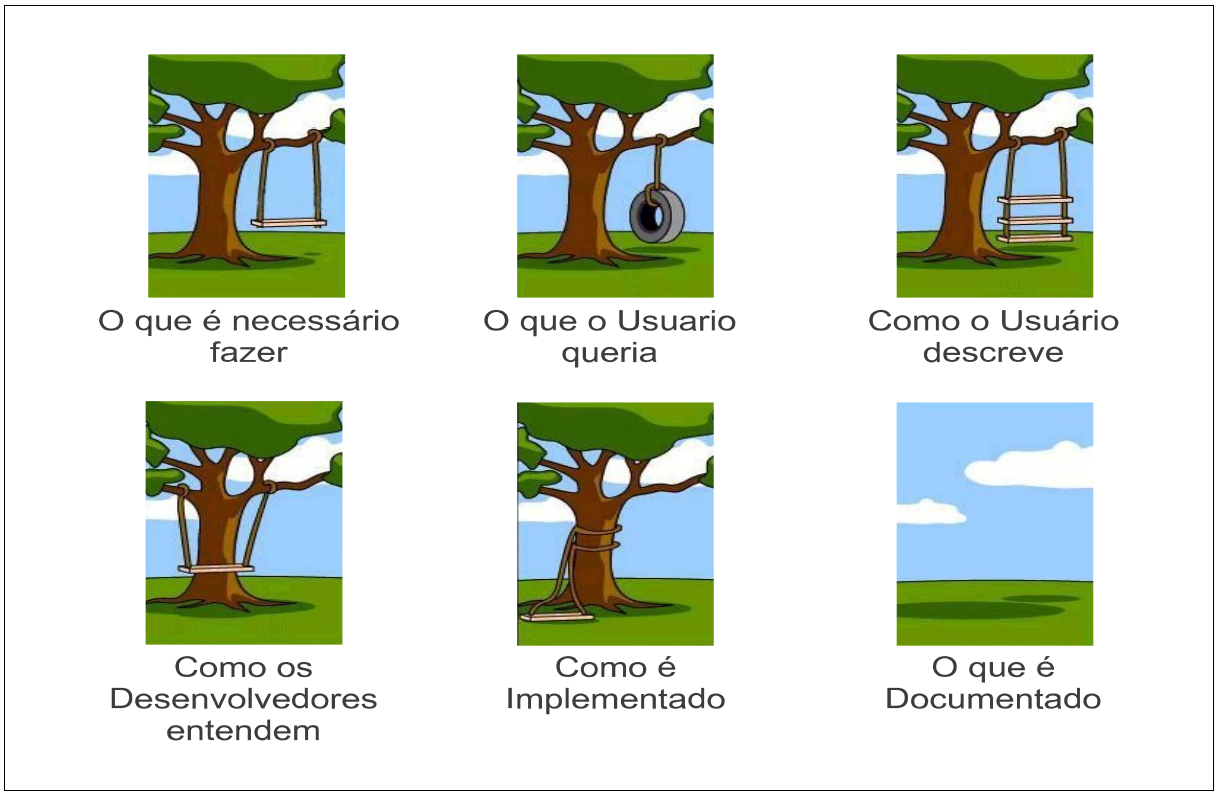


Figura 2 – Divergência de entendimento entre cliente e desenvolvedores
Adaptado de: http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/intro/processo.htm

Como é notado, cada etapa do processo tem um entendimento totalmente diferente um do outro, acarretando em falhas que são resultantes de má gestão do *software* e de seus requisitos.

Boa parte destas falhas poderiam ter sido evitadas se houvesse um boa comunicação entre a equipe de desenvolvimento e se desde o inicio tivesses conseguido entender qual era realmente a expectativa do usuário/cliente e para evitar mais problemas como estes, buscou-se cada vez mais organização para melhorar a qualidade dos *softwares* produzidos. De acordo com Guerra (2009) a qualidade de *software* constitui uma área cuja demanda está crescendo significativamente, pois os usuários exigem cada vez mais eficiência, eficácia, dentre

outras características de qualidade importantes para um produto tão especial como o *software*.

Para chegar a estes resultados, normalmente um produto de *software* é feito dentro de um projeto, o que Filho (2001) e Sommerville (2007) chamam em suas representações de processo, e ainda que existam muitos processos de *software* diferentes, algumas atividades fundamentais são comuns a todos eles como a especificação de *software*, projeto e implementação, validação e evolução.

Não só é importante o processo de *software* estar bem definido, como também precisa ter subdivisões que permitam avaliar o progresso do projeto e corrigir seus rumos quando acontecerem problemas. Estas subdivisões são chamadas de fases, atividades, iterações ou módulos. Cada subdivisão geralmente termina com um marco, isto é, pontos que representam estados significativos do projeto. Em alguns casos esses marcos são utilizados apenas para acompanhamento interno da equipe de desenvolvimento, outras vezes, são esses marcos de indicam quando um módulo está pronto e pode ser entregue ao cliente.

Paralelamente à demanda do mercado, existe um movimento nacional e internacional, no sentido de estabelecer normas na área de Engenharia de *Software*, como é o caso da ISO (*International Organization for Standardization* – Organização Internacional para Padronização), no Subcomitê de Engenharia de *Software*, e da Associação Brasileira de Normas Técnicas – ABNT.

Estas normas ou padronizações que estas instituições vem produzindo permitem aos desenvolvedores de *softwares* trabalharem em equipes e até prestar serviços de manutenção junto a sistemas desenvolvidos por outras equipes de programação, já que conseguem identificar as funcionalidades de modo claro e objetivo seguindo as documentações do *software* e modelos de Engenharia de *Software*.

Além dessa dificuldade da equipe de desenvolvimento de entender o que o cliente precisa durante o levantamento de requisitos, ainda existe um outro problema, que é comum no desenvolvimento de *software:* a instabilidade dos requisitos, que acontece quando clientes e usuários trazem novos requisitos, ou alterações de requisitos, quando o desenvolvimento já está em fase adiantada. Isto normalmente eleva os custos do projeto e muitas vezes significa desfazer algumas rotinas e perder trabalhos que já tinham sido feitos.

Filho (2001) afirma que essa instabilidade de requisitos é tão prejudicial e

danosa na engenharia de *software* quanto em qualquer outra engenharia e informa que a gestão de requisitos é a responsável por manter sob controle o conjunto de requisitos de um produto afim de reduzir a instabilidade destes, mesmo diante de algumas inevitáveis alterações.

Essa instabilidade também eleva os prazos durante o desenvolvimento, tornando ainda mais difícil a vida de quem planeja o desenvolvimento de um *software*, onde criar cronogramas e segui-los corretamente é quase impossível. E, para um produto ser viável tem de ser produzido dentro de certos parâmetros de prazo e custo, formando um triângulo crítico dentro da engenharia de *software* (ver Figura 3).

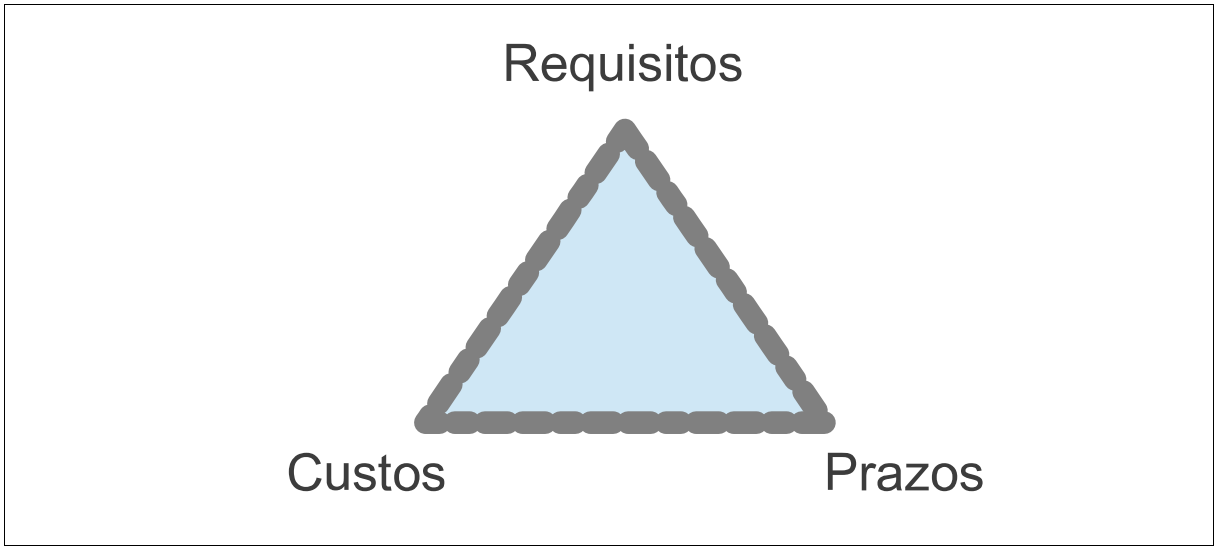


Figura 3 – Triangulo crítico da engenharia de *software*
Adaptado de: Figura 3, Filho (2001).

Na Figura 3 é possível notar que os requisitos, os custos e os prazos estão todos interligados. Quando aumenta-se os requisitos, ou aumentarão também os custos ou os prazos. Em certos projetos até ambos podem aumentar, e quando diminui-se os requisitos, poderá haver uma baixa também nos custos ou prazos.

Para suprir essas adversidades que eram bem frequentes, surgiram os métodos de desenvolvimento ágeis, que gerenciam melhor as mudanças de requisitos evitando que os custos e prazos sejam modificados ou sofram muitas alterações.

2.1.1 Modelos de Engenharia de Software

Os primeiros paradigmas da ES que surgiram Então, para evitar prejuízos durante e após o desenvolvimento de uma aplicação, surgiram alguns paradigmas de ES capazes de minimizar ou resolver estes inconvenientes, tais como: Ciclo de Vida Clássico ou Modelo Cascata, Modelo Incremental, Prototipação, Modelo Espiral, Modelo Baseado em Componentes, Métodos Formais, Engenharia de *Software* Orientada a Agentes, Engenharia de *Software* Orientada a Aspectos, Metodologias de Desenvolvimento Ágil, entre outros.

Os chamados modelos de processos tradicionais, foram os primeiros modelos desenvolvido, onde podemos destacar o Modelo Cascata, também chamado de ciclo de vida clássico segue uma aborgagem sequencial e sistemática para o desenvolvimento de *software*, cujo pode ser entendido melhor ao observar a Figura 4.

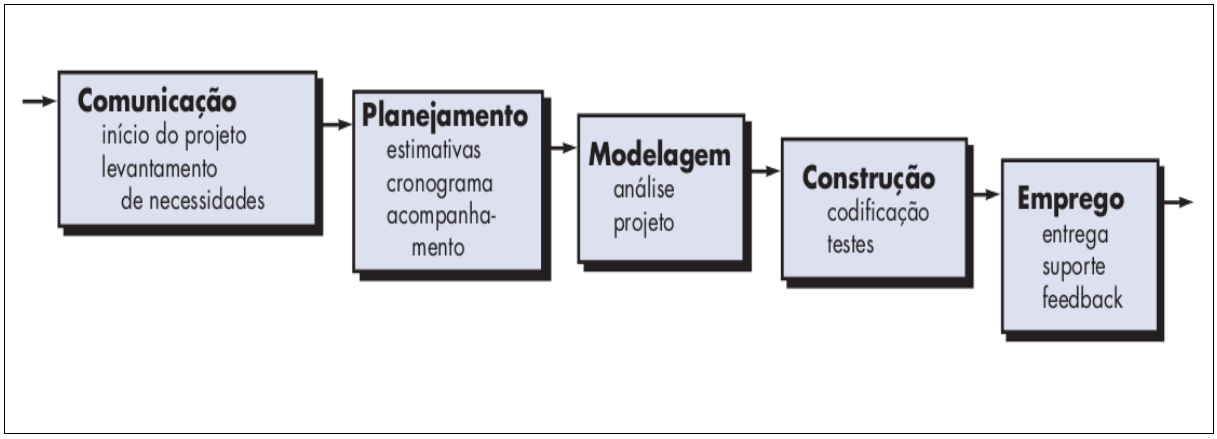


Figura 4 – Ciclo de vida clássico ou modelo cascata
Extraído de: Pressman 2011 - Figura 2.3.

O modelo Cascata começa com o levantamento de necessidades por parte do cliente, avançando pelas fases de planejamento, modelagem, construção, emprego e culminando no suporte contínuo do *software* concluído.

Em outras situações, que também possuem requisitos iniciais razoavelmente bem definidos, entretanto, devido ao escopo geral do trabalho de desenvolvimento, o uso de um processo puramente linear não é utilizado, pode-se optar por um modelo de processo incremental, que combina elementos dos fluxos de processos lineares e paralelos, aplicando sequências lineares, de forma escalonada, à medida que o tempo vai avançando e cada sequência gera um incremento, que pode ser entregue como podemos visualizar na Figura 5.

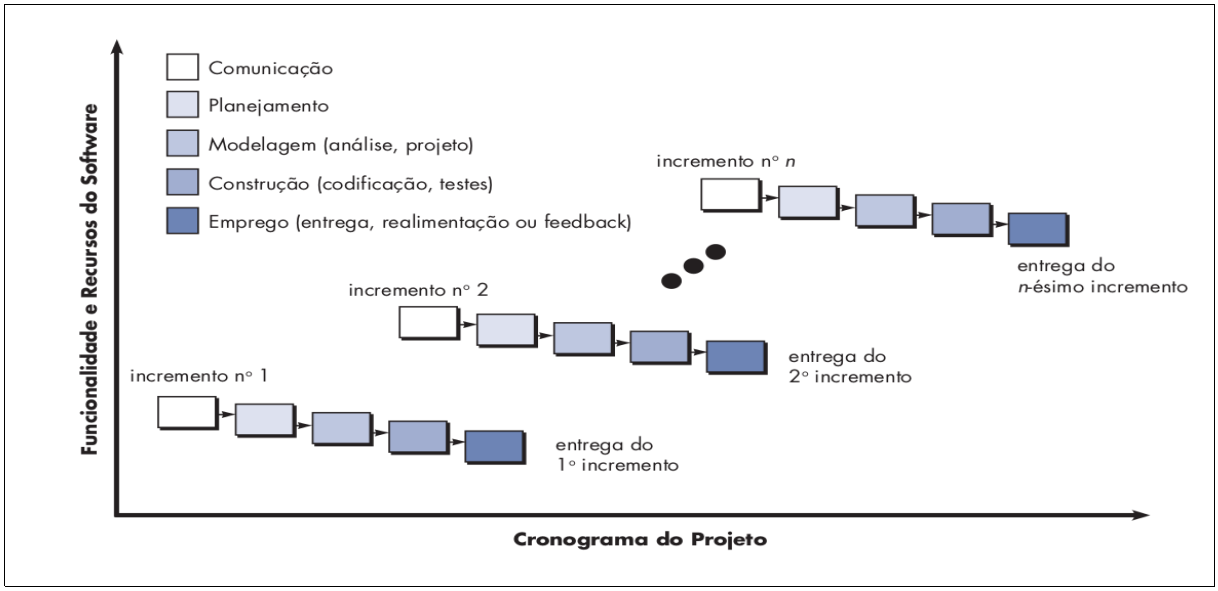


Figura 5 – Modelo Incremental
Extraído de: Pressman 2011 - Figura 2.5.

Já muitos casos os *softwares* precisam evoluir ao longo do tempo, assim quando o sistema é mais complexo, e o desenvolvimento do projeto avança, as necessidades de negócio e de produto podem mudar, tornando inadequado seguir um planejamento em linha reta de um produto final surgindo assim o modelo de prototipação (ver Figura 6) (PRESSMAN, 2011).

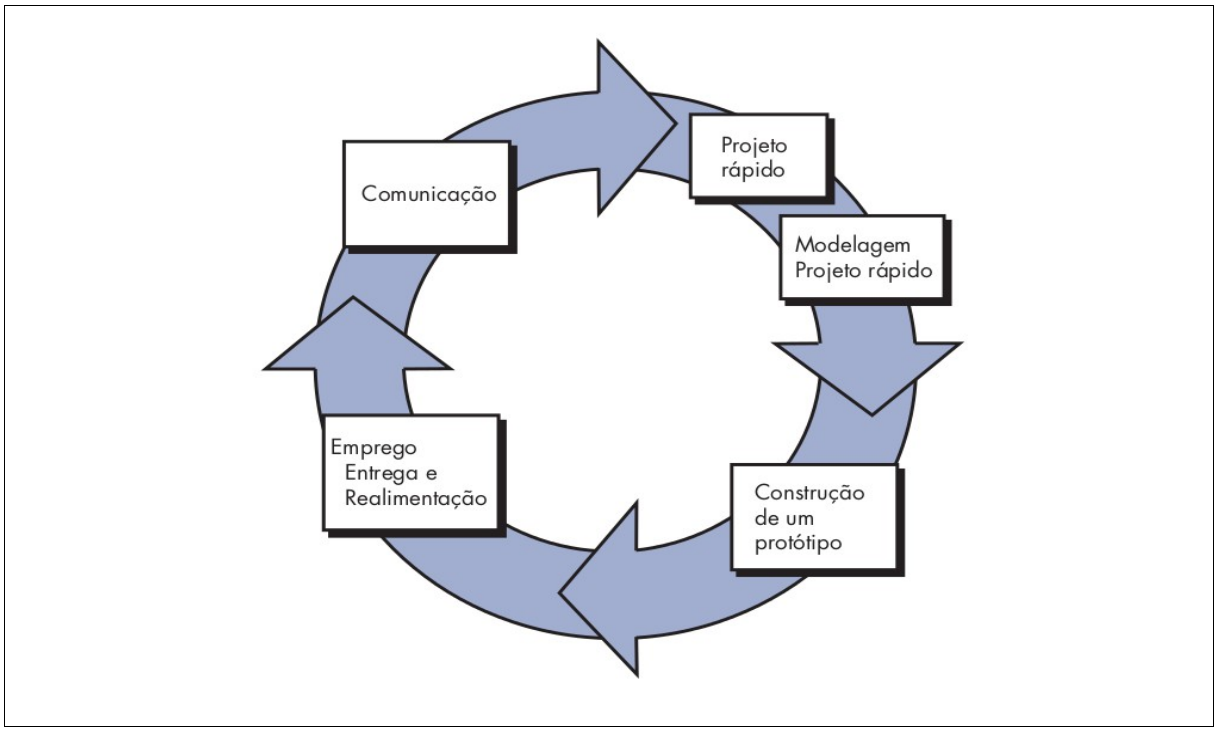


Figura 6 – Paradigma da Prototipação
Extraído de: Pressman 2011 - Figura 2.6.

Esses modelos são uma representação abstrata do processo de um *software* e podem ser utilizados para explicar diferentes abordagens no processo de desenvolvimento, além de poderem ser amplamente adaptados para criar processos mais específicos. (SOMMERVILLE, 2007).

Já os métodos de desenvolvimento ágil, surgiram para sanar fraquezas reais e perceptíveis da Engenharia de *Software* convencional. Pressman (2011) afirma que esses métodos oferecem vários benefícios importantes, no entanto, não são para todos os projetos, produtos, pessoas e situações e não devem, nem podem ser aplicados como uma filosofia geral para todos os trabalhos de produção de um *software*.

Um dos fatores que tornaram os métodos de desenvolvimento ágil tão utilizados foi a dinamicidade do mundo moderno onde em muitas situações, não se consegue definir completamente os requisitos antes que se inicie um projeto, e Pressman (2011) discorre que é preciso ser ágil o suficiente para dar uma resposta ao ambiente bastante mutável, e onde essas mudanças geralmente são caras, aiinda mais se essas mudanças forem sem controle e mal gerenciadas, o que proporciona uma grande busca por métodos que possam contribuir para evitar que isso ocorra.

2.1.2 Métodos de Desenvolvimento Ágeis

As abordagens ágeis têm como característica mais convincente, a habilidade de reduzir os custos de mudanças ao longo de todo o processo de desenvolvimento de *softwares*, sendo que a maioria deles se utiliza de ciclos curtos, que são chamados de iterações e normalmente têm duração de poucas semanas.

Para isso, as metodologias de desenvolvimento ágil valorizam alguns princípios, que podem ser encontrados em um manifesto[1] criado por desenvolvedores para difundir a cultura de uma programação produtiva, como mostra a Figura 7.

Seguindo o proposto pelo manifesto, tem-se muito mais chances de ter um cliente satisfeito quando ele é parte importante no processo. Uma vez que acompanhando e visualizando a evolução do sistema o cliente pode descobrir novas prioridades e desde cedo e iniciar as mudanças que são necessárias.

1 Manifesto Agil - http://agilemanifesto.org/iso/ptbr/

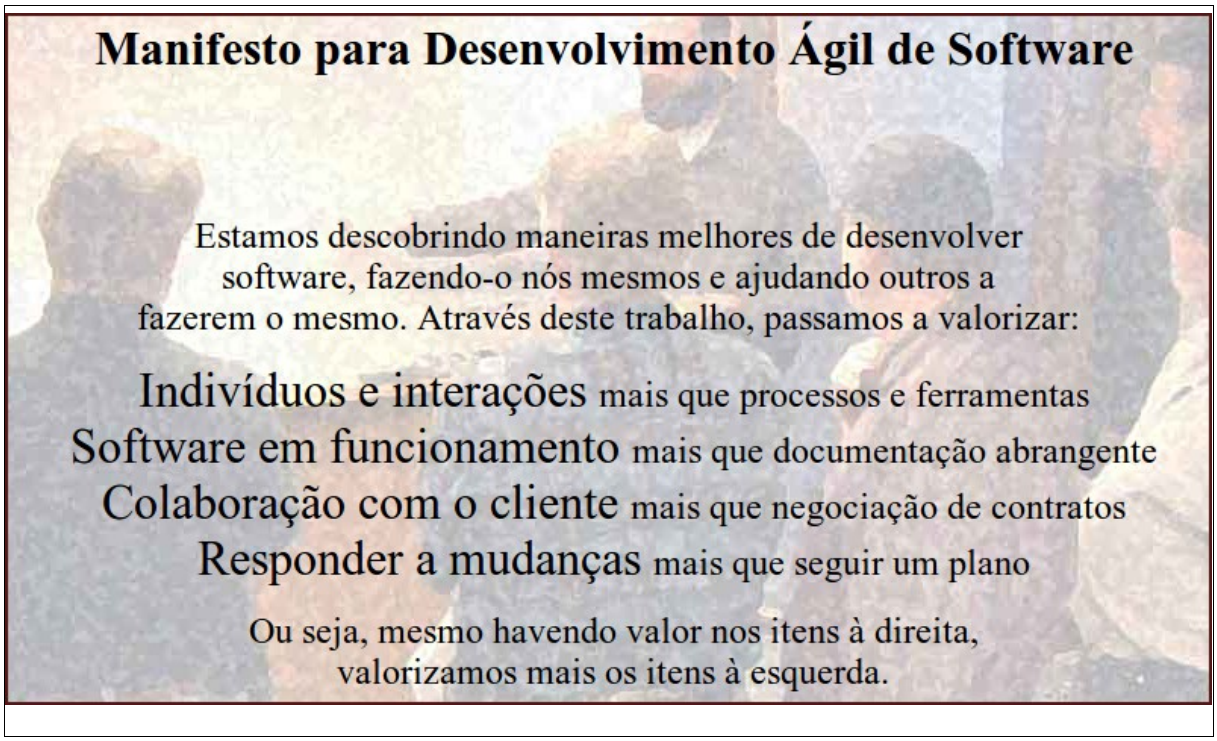


Figura 7 – Manifesto para desenvolvimento ágil
Extraído de: <http://agilemanifesto.org/iso/ptbr/>

Outro fator importante que pode-se citar sobre esses métodos de desenvolvimento é que as equipes de trabalho podem ajustá-los às suas necessidades buscando sempre a melhor forma de trabalhar, como está explicitado no site, “em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento” (agilemanifesto, 2001-2014).

Dentre as principais metodologias e descrições de processos, métodos e notações de modelagens, pode-se enfatizar as duas mais utilizadas, a *Extreme Programming* (ou simplesmente *XP*) e a *Scrum*.

De acordo com Pressman (2011), a *XP* é a abordagem mais amplamente utilizada para desenvolvimento de *software* ágil e emprega como paradigma uma abordagem orientada a objetos.

Existe um trabalho inaugural escrito por Kent Beck, apontando cinco valores que estabelecem as bases para todo trabalho realizado como parte da Programação Extrema: comunicação, simplicidade, *feedback*, coragem e/ou disciplina e o respeito, sendo esses valores essenciais para a *XP* e as equipes que os seguirem estarão cada vez mais ligadas entre si e sentirão mais respeito para com esse método (PRESSMAN, 2011).

É de suma importância ainda saber que a *XP* envolve um conjunto de quatro

atividades metodológicas, conforme realçado na Figura 8: planejamento, projeto, codificação e testes.

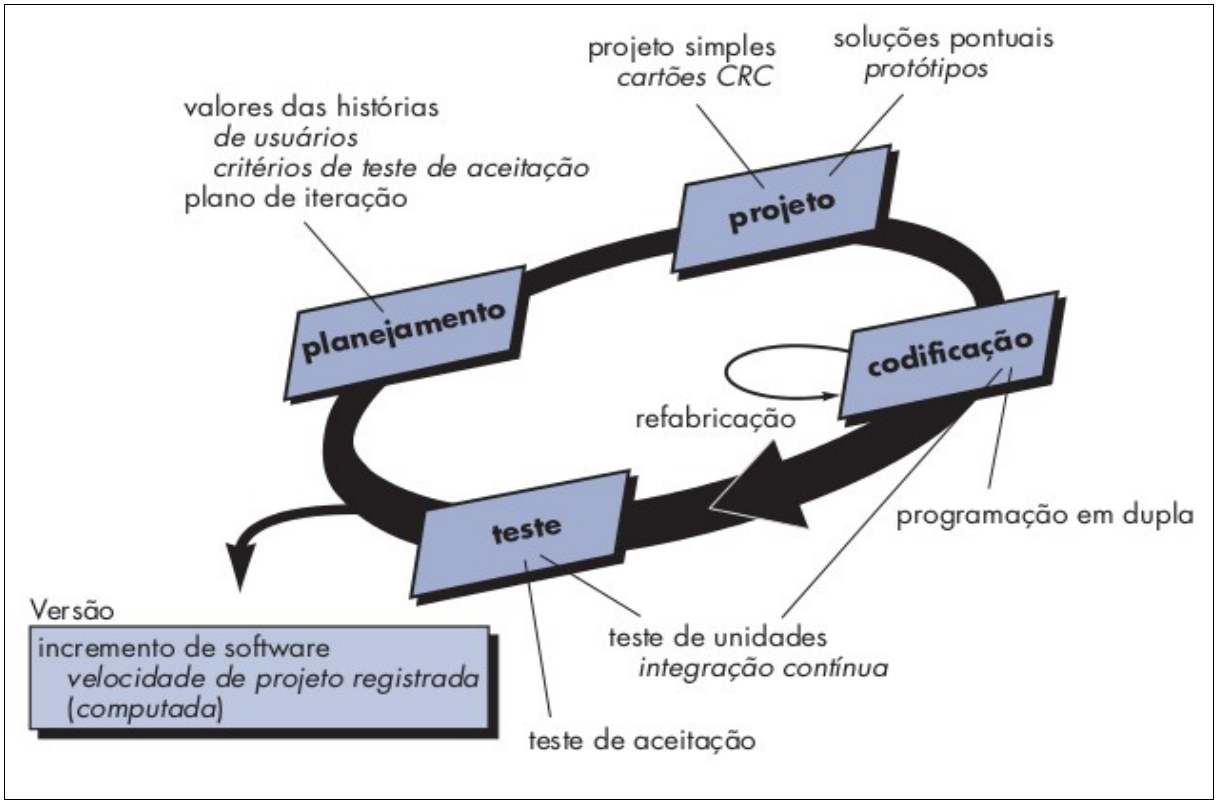


Figura 8 – Processo da programação extrema (*XP*)
Extraído de: Pressman 2011 - Figura 3.2.

O planejamento inicia-se ouvindo o cliente, uma atividade de levantamento de requisitos que possibilita os membros da equipe *XP* a entender o ambiente a ser trabalhado e gerando uma percepção ampla sobre o *software* solicitado.

As histórias apuradas são analisadas e solicitado do cliente uma seleção de prioridades, depois monta-se um projeto a partir das histórias selecionadas e é calculado a velocidade do projeto referente ao número de histórias do cliente que serão implementadas.

O cliente tem liberdade de acrescentar histórias, mudar o valor de uma existente, dividir algumas ou até mesmo eliminá-las conforme o trabalho de desenvolvimento prossegue.

Logo depois da definição do projeto seria a codificação, mas a equipe *XP* trabalha testes para verificar se as histórias realmente atendem todos os requisitos do sistema a ser desenvolvido, gerando um incremento de *software*, que será codificado.

> Depois de desenvolvidas as histórias e o trabalho preliminar de elaboração do projeto ter sido feito, a equipe não passa para a codificação, mas sim, desenvolve uma série de testes de unidades que exercitarão cada uma das histórias a ser inclusas na versão corrente (incremento de *software*). […] Estando o código completo, este pode ser testado em unidade imediatamente, e, dessa forma, prover, instantaneamente, *feedback* para os desenvolvedores (PRESSMAN, 2011, p. 90).

Outra metodologias de desenvolvimento ágil, a *Scrum*, que é um método de desenvolvimento ágil de *software* concebido por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 1990, mas só a partir de 2000 tornou-se mais popular, desbancando métodos tradicionais e tornando-se a forma mais comum de se trabalhar em projetos de desenvolvimento de *software* (PRESSMAN, 2011), (SABBAGH, 2013).

Sabbagh (2013) diz que *Scrum* é um *framework* simples e pequeno, funcionando bem em cada contexto se utilizado em conjunto com outras técnicas e práticas, podendo permitir a redução dos riscos de insucesso, e a entrega de valor mais rápido e desde cedo, pois sempre tem algo a ser apresentado, além de lidar com as inevitáveis mudanças de escopo, transformando-as em vantagem competitiva. Seu uso pode também aumentar a qualidade do produto entregue e melhorar a produtividade das equipes.

Diferentemente das metodologias tradicionais onde frequentemente acontece apenas uma ou poucas entregas e praticamente já no final do projeto de desenvolvimento, o *Scrum* possibilita que se entregue, desde cedo, partes do produto funcionando. E cada uma dessas entregas proporcionam um retorno ao investimento do cliente, permitindo também um *feedback* rápido sobre o produto possibilitando as mudanças ou adições que se fizerem necessárias (SABBAGH, 2013).

Pressman (2011) informa que os princípios do *Scrum* são consistentes com o manifesto ágil e são usados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades estruturais: requisitos, análise, projeto, evolução e entrega.

O *Scrum* possui uma estrutura que facilita a entrega de módulos utilizáveis os quais são chamados de *sprints* de *backlog*, conforme nos revela a Figura 9, que exibe os passos que deve existir no desenvolvimento de um *software* utilizando essa metodologia.

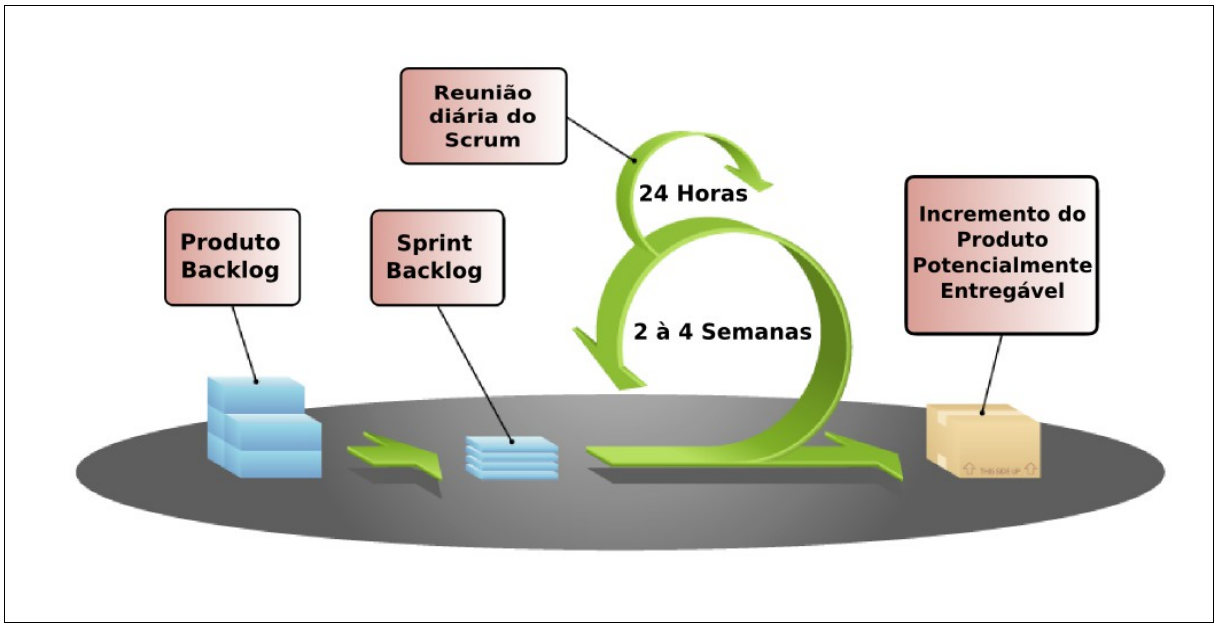


Figura 9 – Ciclo de fases do *SCRUM*
Adaptado de: <http://www.profissionaisti.com.br/2011/09/metodologia-agil-o-*scrum*-de-forma-simples/>

Como é possível notar, no *Scrum* cada iteração é chamada de *sprint* e geralmente cada *sprint* tem um período que pode variar de poucos dias a algumas semanas, sendo que as pessoas envolvidas no processo de desenvolvimento são divididas em três papéis principais: o *Scrum Master*, o *Product Owner* (dono do produto) e a equipe (GOMES, 2013).

O *Scrum Master* é também chamado de facilitador, sendo o responsável por manter o processo em funcionamento, assegurando que todas as regras estão sendo aplicadas. É também função do *Scrum Master* remover os impedimentos da equipe, isto é, resolver qualquer problema que possa atrapalhar o progresso do desenvolvimento, garantindo assim que o objetivo da iteração seja atingido.

Já o *Product Owner* é o maior interessado no projeto, é ele quem solicitou o sistema e vai definir as funcionalidades a serem desenvolvidas e suas prioridades, mantendo uma lista ordenada, incompleta e dinâmica de itens que representam o chamado de *product backlog,* que ele acredita que será produzido ao longo do projeto (SABBAGH, 2013).

Por fim, Gomes (2013) nos fala da Equipe, que é composta por um número que varia de cinco a nove pessoas, sendo dos mais variados perfis, como testadores, desenvolvedores, designers, analistas de negócio etc. Seu principal objetivo é implementar as funcionalidades que foram selecionadas para serem

desenvolvidas na iteração e entregá-las em funcionamento ao final do período.

Normalmente os grupos de projetos com *Scrum* são pequenos e os membros interagem e comunicam-se durante todo expediente de trabalho. A integração da equipe é essencial para o sucesso do projeto. O time tem autonomia para decidir o quanto é possível de se fazer em um ciclo. Neste caso, tem-se uma meta específica e a ajusta de forma que todos possam se comprometer com ela, onde cada membro da equipe é igualmente responsável por atingir essa meta (SABBAGH, 2013).

Tudo isso possibilita a redução de riscos, sendo que, a partir de ciclos de entrega, o cliente tem um acompanhamento mais detalhado do projeto e visualiza as mudanças fundamentais mais rapidamente, sem necessidade de desfazer e refazer funcionalidades, pois sabe-se que dificilmente uma pessoa consegue colher todos os requisitos imprescindíveis para elaboração de um sistema completo de uma só vez.

2.2 Banco de Dados

Continuando então o processo de criação do *software,* é de ampla importância a modelagem de uma base de dados antes de ir para a parte de implementação do banco de dados (Linguagem SQL), dado que, uma base de dados bem modelada poupa muitos problemas futuros, já que Ramakrishnan (2011) afirma que um bom gerenciamento pode até não garantir o sucesso do projeto, mas o mau gerenciamento geralmente resulta em falhas.

E devemos ter em mente que o banco de dados é um ponto crucial em qualquer aplicação que tenha que guardar dados e depois recuperá-los de alguma forma e segundo Angelotti (2010) banco de dados é de grande importância no mundo da informática, uma vez que a informação é um bem precioso e deve ser armazenada de forma coerente e adequada, já que, atualmente, por menor e mais simples que seja um Sistema de Informação, ele precisará ter a capacidade de armazenar e recuperar dados rapidamente.

Ramakrishnan (2011) complementa as informações supracitadas afirmando que a quantidade de informação que nos são disponíveis está literalmente explodindo, e o valor dos dados como um ativo organizacional é amplamente reconhecido.

Mas antes de prosseguirmos devemos saber que na informática em si dados são tudo que podemos inferir ou coletar sobre uma situação específica e podem ser úteis ou não. Quando esses dados são úteis formam a informação, porém em banco de dados essas duas palavras podem ser consideradas sinônimas, já que desejamos guardar em nossa base de dados apenas informações úteis.

E para ter um banco de dados estruturado e habilitado para manter os dados consistentes é necessário a modelagem dos dados que virá após todo levantamento de requisitos e com o entendimento adequado de quais são as necessidades do usuário. Da Silva (2001) aponta essa modelagem ou modelo, como referente a uma interpretação simplificada da realidade.

A Modelagem pode se dar através de um Modelo de Entidade e Relacionamento (ER) também conhecida como Modelagem Conceitual do Banco de Dados. E para esse propósito, a UML[2] fornece diversos construtores que correspondem aos construtores do ER. (RAMAKRISHNAN, 2011).

Piva (2010) diz que com o projeto conceitual montado, precisamos definir o Sistema Gerenciador de Banco de Dados (SGBD), pois é importante verificar quais são os tipos de dados que este aceita, bem como seus tamanhos, pois tais características variam muito.

Um SGBD é um *software* projetado para auxiliar a manutenção e utilização de vastos conjuntos de dados que trazem para quem o utiliza a independência, acesso eficiente, integridade e segurança, administração acesso concorrente e recuperação de falhas, dentre outras vantagens.

Depois de definido o SGBD podemos passar à etapa seguinte: fazer o projeto lógico, de forma a nos conduzir a um Modelo Relacional (MR), utilizado em bancos de dados relacionais como o MySQL.

2.2.1 Modelagem do Banco de Dados

Angelotti (2010) nos informa que o MR pode ser descritivo. Onde se coloca o nome da tabela, e dentro de parênteses seus atributos, identificando o atributo

2 UML (Unified Modelling Language) - é uma linguagem diagramática, utilizável para especificação, visualização e documentação de sistemas de *software* surgindo em 1997 na sequência de um esforço de unificação de três das principais linguagens de modelação orientadas por objectos (OMT, Booch e OOSE). Extraído de: DA SILVA, 2001.

chave ou chave primária através de sublinhado, como exemplificado na Figura 10.

```
tbTurma(codigo_turma, nome_turma)

tbPessoa(RG_pessoa, nome_pessoa, telefone_pessoa)

tbProfessor(matricula_professor,RG_pessoa,CPF_professor,escolaridade_professor)
RG_pessoa referencia tbPessoa
tbAluno(matricula_aluno, RG_pessoa, data_nascimento_aluno, codigo_turma)
codigo_turma referencia tbTurma
RG_pessoa referencia tbPessoa
```

Figura 10 – Modelo Relacional - forma descritiva
Extraído de: Angelotti 2010 - Figura 5.1.

A autora referenciada ainda diz que o Modelo Relacional também pode ser diagramado, e que esta é a forma mais comum de se encontrar.

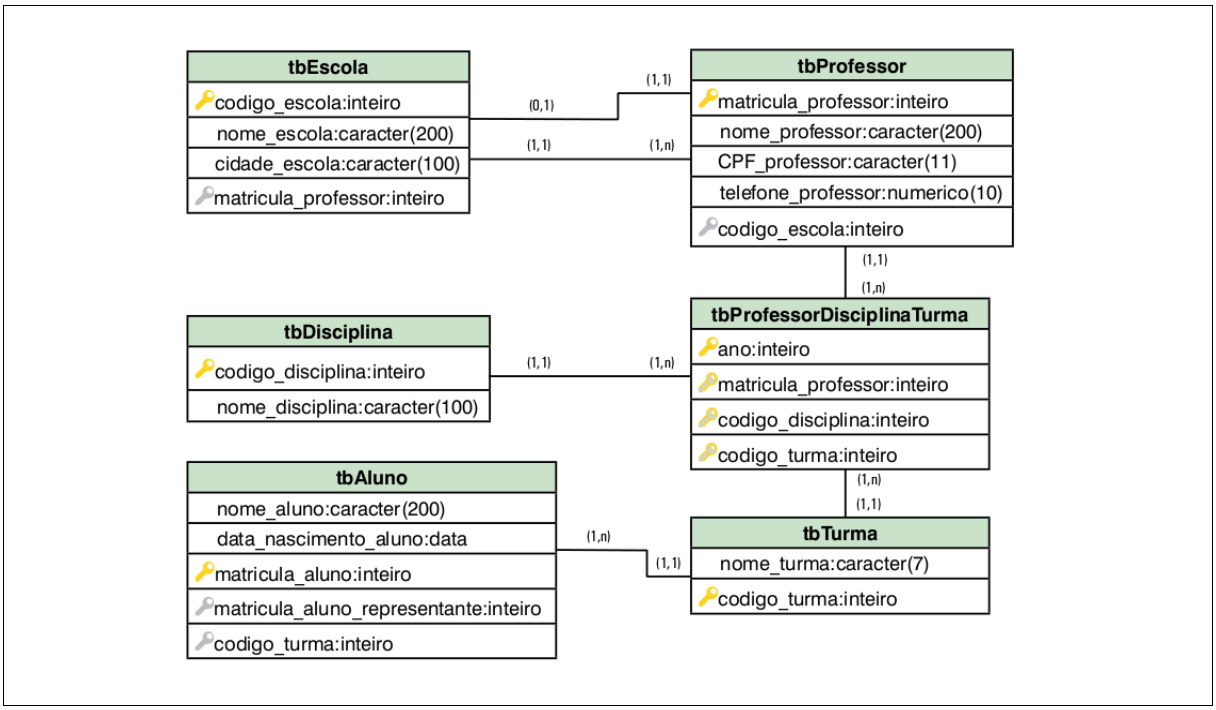


Figura 11 – Modelo Relacional - forma diagramada utilizando o BrModelo
Extraído de: Angelotti 2010 - Figura 5.2.

Para fazer a diagramação existem vários programas que auxiliam essa etapa, agilizando e simplificando todo o processo, além do mais, caso o desenvolvedor tenha descrito todo MR, a diagramação também será bastante facilitada (ver Figura 11).

A Figura 11 foi produzida utilizando o BrModelo, as chaves primárias aqui

são identificadas por uma chave dourada antes do nome do atributo, as chaves estrangeiras são representadas por chaves prateadas e as chaves estrangeiras que fazem parte da chave primária são exibidas com uma chave prateada com contornos dourados.

Outros programas podem trazer simbologias diferentes, mas todos trabalham dentro de um padrão que é de fácil entendimento para pessoas que possuem algum entendimento de banco de dados e modelagem relacional, conforme apresenta a Figura 12.

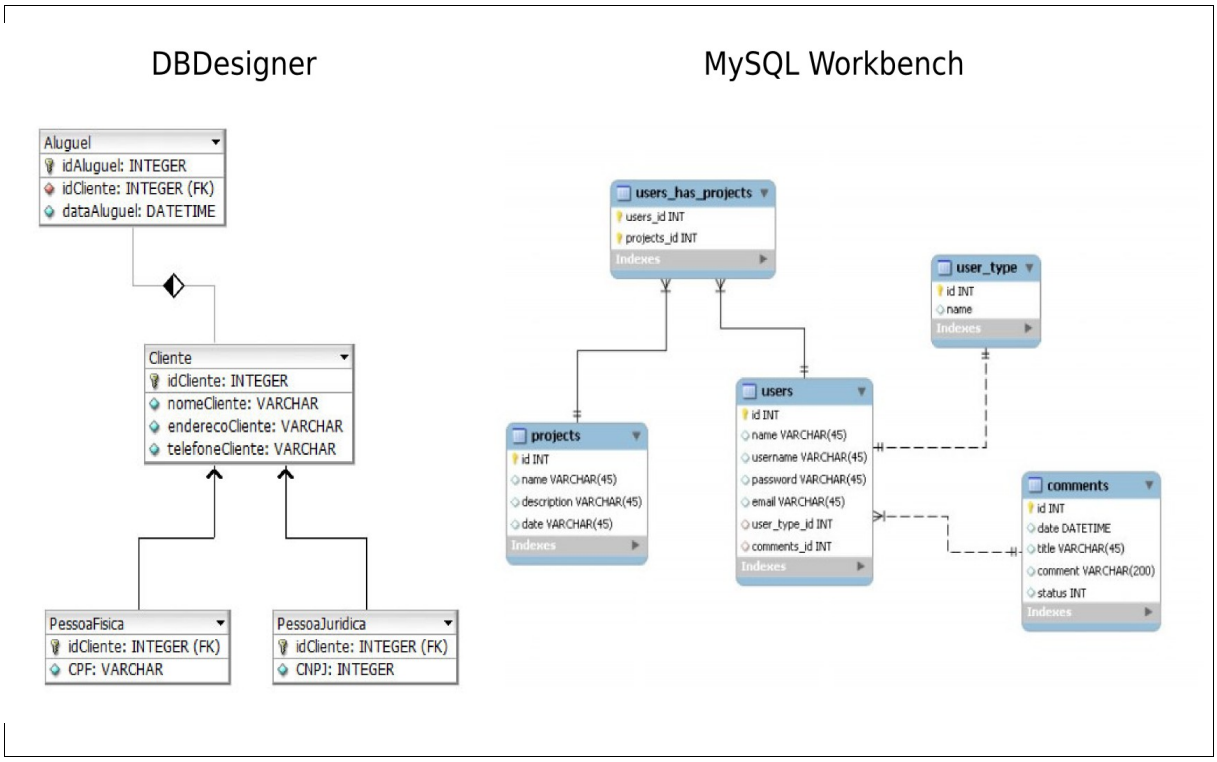


Figura 12 – Exemplo de MR utilizando outros programas de modelagem
Fonte: o Autor.

Alguns desses programas, como *MySQL Workbench*[3], trazem a funcionalidade de exportar o modelo relacional construído para linguagem SQL[4]

3 MySQL Workbench - é uma ferramenta visual unificada para arquitetos de banco de dados, desenvolvedores e DBAs desenvolvido pela Oracle. Que fornece modelagem de dados, desenvolvimento de SQL e ferramentas de administração abrangentes e muito mais. Fonte:<http://www.mysql.com/products/workbench/>

4 Linguagem SQL (Structured Query Language ou Linguagem de Consulta Estruturada) - é uma linguagem para banco de dados relacional. Ela foi desenvolvida na década de 70 e tem sido aprimorada e padronizada desde então. Extraído de: Angelotti, 2010.

fazendo com que o processo torne-se ainda mais ágil, além de criar o banco de dados propriamente dito e dar suporte a seu gerenciamento.

É de grande importância quando se trabalha com banco de dados, fazer um Dicionário de Dados para a base, isso fará com que o programador tenha um conhecimento mais detalhado de todos os campos do banco e encontre maior facilidade quando precisar fazer alguma alteração ou correção no banco. A forma de como criar um dicionário de dados é seguindo o modelo apresentado na Figura 13.

Nome	Descrição	Tipo	Tamanho	Nulo	Regra (check)	Chave	Default	Unique
cod_pessoa	Armazena o código do voluntário	Inteiro	__	Não	__	PK	__	Não
nome_pessoa	Armazena o nome do voluntário	Caracter	50	Não	__	__	__	Não
fone_pessoa	Armazena o telefone do voluntário	Caracter	10	Sim	__	__	__	Não

Tabela tbPessoa

Figura 13 – Exemplo de criação de dicionário de dados
Extraído de: *Angelotti, 2010.*

Vale lembrar que SQL aceita a criação de índices, que é definido sobre um atributo para melhorar no desempenho das consultas, confirmando a afirmação de Angelotti (2010), que diz que, a criação de índices em uma tabela é o principal método de otimização que os SGBD's oferecem.

2.2.2 SQL e MySQL

O SQL é uma linguagem que teve seu início em junho de 1970 proposto por Edgar F. Codd e hoje é considerada o fundamento e um padrão para qualquer Sistema de Gerência de Banco de Dados Relacionais (SGBDR), tendo um objetivo bem definido: a manipulação dos dados. Sua primeira implementação comercial foi realizada pela *Relational Software*, que hoje é a *Oracle Corporation* (DAMAS, 2007).

Mesmo a linguagem SQL sendo um padrão, nem todos os SGBDs trabalham iguais, existem diferenças nas sistaxes de alguns comandos da linguagem, como é o

caso do LIKE, comando utilizado para buscar principalmente palavras que possuem o conjunto ou caractere comparado. Em alguns SGBDs, como no MySQL a comparação se dava com o simbolo de porcentagem (%) dentro de aspas simples para determinar um conjunto qualquer de caracteres após o comando LIKE.

EX.: ...LIKE 't%' (neste exemplo seria retornado todas as palavras que iniciassem com a consoante 't'.

Já o Access, SGBD da Microsoft, o caractere '%' é substituido pelo '*', onde a mesma consulta se daria da seguinte maneira: ...LIKE 't*'. Essas pequenas diferenças devem ser levadas em consideração, pois caso queira mudar o SGBD no meio do desenvolvimento pode ser que tenham que verificar todo banco de dados novamente e refazer algumas consultas para garantir a funcionalidade correta da aplicação.

Depois de escolhido o gerenciador que se vai trabalhar, deve-se cria o banco de dados, o próprio SGBD pode facilitar essa atividade, já que muitos dão suporte visual para criar o banco e partir para a implementação física através de botões. Logo após, basta criar as tabelas seguindo o discionário de dados ou o modelo relacional.

Segundo Ramarkrishnan (2000), o MySQL é um dos melhores SGBD (Sistema Gerenciador de Bancos de Dados) relacional de codigo aberto, sendo o mais popular para aplicações *Web*.

2.3 Programação *Web*

O termo programação *web* é utilizado para a ação de desenvolver sites ou aplicações *web*, que são sistemas acessados através de navegadores (browsers), como existem muito navegadores diferentes, o programador *web* deve tomar cuidados para que sua aplicação funcione bem na maioria deles.

Rodrigues (2010) diz que “Uma página da *web* é um arquivo onde você pode inserir textos, imagens, tabelas e incorporar planilhas, apresentações, vídeos e muito mais”. E que desde meados da década de 1990 o desenvolvimento *web* vem se expandindo no mundo, pois a *Internet* trouxe novos hábitos de consumo e criou inovadoras formas de negócios.

Um problema encontrado na hora de criar um site ou sistema *web* é o de compatibilidade causado pela grande variedade de dispositivos que podem ser

utilizados para acessar os *sites* e as aplicações *web*. Atualmente, as pessoas acessam os sites e as aplicações *web* através de computadores tradicionais, *tablets*, celulares, televisores, entre outros. Esses dispositivos possuem telas de tamanhos diferentes. Dessa forma, os desenvolvedores *web* devem considerar essas diferenças na criação das páginas *web*.

Então, em outubro de 1994, um consórcio regulador foi criado para manter a padronização dos navegadores e também para os programadores, o *World Wide Web Consortium* ou simplesmente *W3C.*

O *W3C* é o responsável pelo que se refere aos padrões para o desenvolvimento de páginas *web*, incluindo *HTML5*, *CSS*, *SVG*, *Ajax*, e outras tecnologias para Aplicações *Web* ou "*WebApps*" (*Web* Design e Aplicações *Web*. Disponível em: <http://www.*w3c*.br/Padroes/*Web*DesignAplicacoes> Acesso em: 13 dezembro 2014).

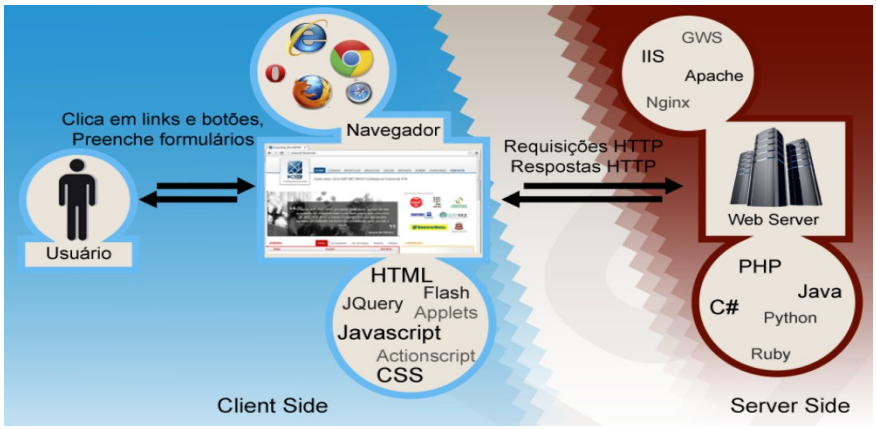


Figura 14 – Diagrama geral da arquitetura *web*
Extraído de: K19, Figura 1.18.

É importante também saber que existem dois tipos pricipais de programadores *web*, os que dedicam-se a programação do lado do usuário, criando as páginas que são visíveis nos navegadores e aqueles que dedicam-se a programação do lado do servidor, codificando as funções que farão a aplicação funcionar (ver Figura 14).

Podemos notar que o diagrama da Figura 14 foi dividido em duas partes: *client side* e *server side*. Outra denominação possível que é muito utilizada hoje em dia para *client side* é *front-end* e para *server side* é *back-end*.

2.3.1 W3C

O *World Wide Web Consortium* (*W3C*[5]) é um consórcio internacional que possui uma equipe trabalhando em tempo integral, auxiliada pelos escritórios locais, como o *W3C* Brasil, para juntamente com organizações filiadas e o plúblico em geral para recomendar, criar e manter padrões tecnológicos baseados na *web* que fossem interoperáveis, abertos e acessíveis.

O *W3C* foi fundado pelo inventor da *web Tim Berners-Le*e que o lidera juntamente com seu diretor executivo, Dr. *Jeffrey Jaffe*, que supervisionam o desenvolvimento continuado da *web* para que atinja todo seu potencial.

O Consórcio desenvolve especificações técnicas e orientações através de um processo projetado para maximizar o consenso sobre as recomendações, garantindo qualidades técnicas e editoriais, além de transparentemente alcançar apoio da comunidade de desenvolvedores, do consórcio e do público em geral.

Seu escritório no Brasil iniciou suas operações em 1º de novembro de 2007 disseminando a cultura de adoção de padrões, traduzindo para o Português os textos produzidos pelo *W3C* que forem de interesse da região e disponibilizando apostilas contendo os padrões recomentados, auxiliando assim os desenvolvedores a construirem páginas *web* assessíveis para todos.

2.3.2 HTML5, CSS3 e Javascript

O *HTML*[6] é uma linguagem de marcação que era bastante utilizada para fazer páginas *web* completas, hoje, com sua evolução, divide funções juntamente com o *CSS* e JavaScript para criar páginas mais dinâmicas.

> Desde o começo, o *HTML* foi criado para ser uma linguagem independente de plataformas, entre 1993 e 1995, o *HTML* ganhou as versões *HTML*+, *HTML*2.0 e *HTML*3.0, que enriqueceram ainda mais as possibilidades da linguagem e em 1997, o grupo de trabalho do *W3C* responsável por manter o padrão do código, e trabalhou na versão 3.2 da linguagem, fazendo com que ela fosse tratada como prática comum (FERREIRA: 2014, p.8).

5 *W3C* (Consórcio *World Wide Web*) - é uma comunidade internacional que desenvolve padrões com o objetivo de garantir o crescimento da *web*. Missão do *W3C*: conduzir a *Web* ao seu potencial máximo. Extraído de: <http://www.*w3c*.br/Home/*Web*Home>.

6 *HTML* (*Hypertext Markup Language*) - é uma Linguagem de Marcação, ou seja, uma série de códigos que definem o formato ou o significado do texto. Essa linguagem de marcação é a principal ferramenta para se fazer uma página na *Internet*. Extraído de: Rodrigues, 2010.

O *W3C* foi tentando evoluir a linguagem e criou uma outra versão chamada *XHTML*, e enquanto focava suas atenções para a criação da segunda versão do XHTML, um grupo chamado *Web Hypertext Application Technology Working Group* ou WHATWG[7] trabalhava em uma versão do HTML que trazia mais flexibilidade para a produção de *websites* e sistemas baseados na *web*. Então estas organizações se juntaram para escrever o que seria chamado hoje de HTML5.

Ferreira (2014) ainda nos diz que um dos principais objetivos do HTML5 é facilitar a manipulação do elemento possibilitando o desenvolvedor a modificar as características dos objetos de forma não intrusiva e de maneira que seja transparente para o usuário final.

Esses elementos que são manipulados pelo HTML, hoje HTML5, são chamados de *tags*, cujos são palavras reservadas da linguagem que são definidas entre parênteses angulares (< e >). Os elementos podem possuir atributos e conteúdo. Os atributos são formados por nome e valor. Normalmente, os valores dos atributos são definidos dentro de aspas dupla e o conteúdo dos elementos é um texto ou outros elementos (ver Figura 15).

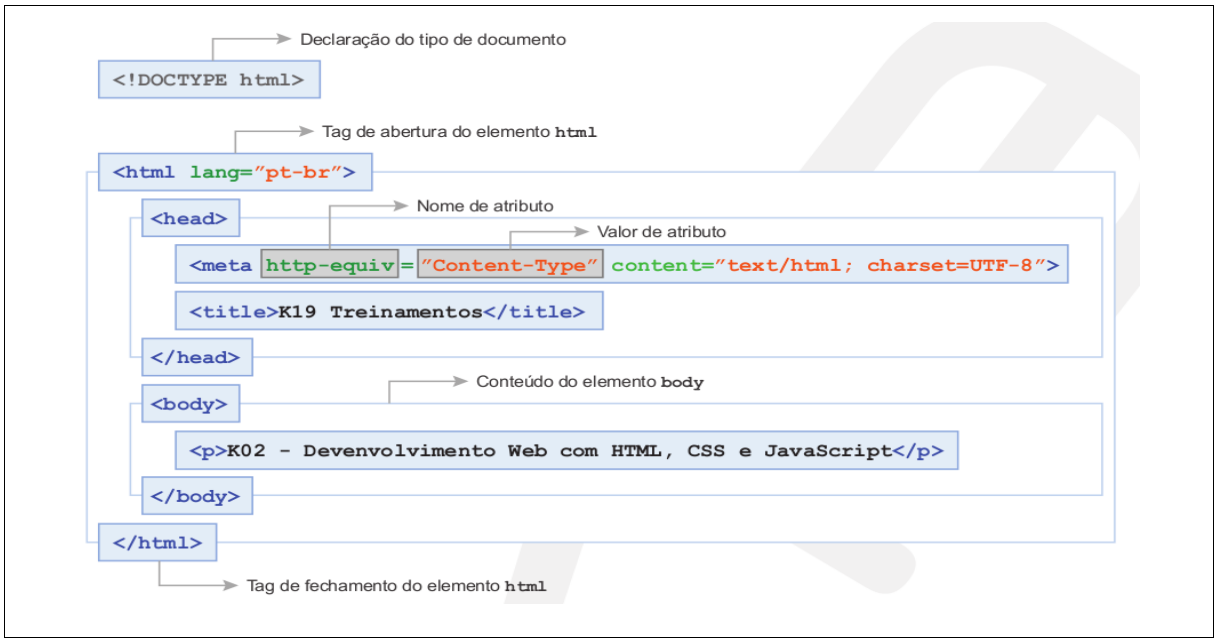


Figura 15: Estrutura de uma página *HTML*
Extraído de: K19, Figura 2.1.

7 WHATWG (*Web* Hypertext Application Technology Working Group) - é uma crescente comunidade de pessoas interessadas em evoluir a *Web*. Ele se concentra principalmente no desenvolvimento de *HTML* e APIs necessárias para aplicativos da *Web*. Extraído de: <https://wiki.whatwg.org/wiki/FAQ#What_is_the_WHATWG.3F>.

Os desenvolvedores podem se utilizar do *HTML5* juntamente com *CSS3* e o *Javascript* para fazerem seu trabalho da melhor maneira possível. Pois essas tecnologias foram desenvolvidas para que seus elementos sejam manipulados de forma que as páginas *web* ou aplicações *web* continuem leves e funcionais.

As páginas *web* expandiram consideravelmente sua usabilidade com o HTML5, principalmente na semântica, que estão mais definidas com as novas *tags* e novos valores para o atributo *type* de *tags* da linguagem. O que podemos notar quando nos deparamos com as *tags nav* que representa uma seção da página que contém *links* para outras partes do *website, a header* que é um grupo de introdução ou elementos de navegação e os *types tel* representando telefones, *search* para um campo de buscas, *url* para endereço de páginas *web* (FERREIRA, 2014).

Como dito anteriormente, esses campos não mudam o visual da aplicação desenvolvida, mas facilitam na questão de buscadores de conteúdo e leitores de tela entenderem os campos, deixando o trabalho destes mais efetivo. Existem ainda outros elementos que aumentam a interatividade dos navegadores, como os vídeos, áudios e outros elementos que podem ser inseridos através do HTML5.

2.3.3 PHP

Quando a gente pensa em *Web* lembra logo de HTML, *CSS* e JavaScript. ssas linguagens são fundamentais no lado dos navegadores. Mas elas são apenas metade da história. Muita coisa precisa acontecer do outro lado da conexão, nos servidores. E é aí que entra o PHP[8], com o poder de transformar páginas estáticas em páginas dinâmicas e sistemas complexos.

> PHP é uma ferramenta que possibilita o pré-processamento de páginas HTML. Dessa forma, PHP consegue alterar o conteúdo de uma página, antes de enviá-la para o navegador. Além disso, PHP também permite capturar entradas de dados do usuário, como formulários e outras formas de interação (BENTO: 2012, p3).

O PHP é uma linguagem de criação de scripts do lado do servidor que foi projetada especificamente para a *web* e uma de suas grandes qualidades é que ele

8 PHP – é uma linguagem de criação de scripts do lado servidor que foi projetada especificamente para *web*. Originalmente significava Personal Home Page, mas foi alterado para PHP Hypertext Preprocessor. Extraído de: Welling, 2005.

funciona na maioria dos sistemas operacionais, principalmente quando utilizado junto com o MySQL, além de ser uma linguagem muito eficiente e de código-fonte aberto e gratuito.

2.4 Trabalhos Relacionados

Existem vários sistemas de gestão escolar espalhados pela *internet*, disponiveis para utilização em escolas de diversos setores e atendendo diferentes níveis de ensino, como o Gennera[9] que adapta seu sistema para utilização em Educação Infantil, Educação Básica, Pré-Vestibular, EJA, Graduação, Pós-Graduação, Ensino Técnico e Profissionalizante, Ensino a Distância e Cursos Livres (ver Figura 16).

Figura 16 – Alguns Sistemas de Gestão Escolar
Fonte: o Autor.

A maioria dos sistemas não disponibilizam testes gratuitos, informando

9 Gennera - http://www.gennera.com.br/gestao-educacional.html

apenas suas principais funcionalidades, e os que disponibilizam, solicitam dados da instituição de ensino, como CNPJ e telefone fixo como dados obrigatórios. As formas como apresentam seus sistemas para os clientes também são variadas, uns aceitando que o cliente faça testes por alguns dias, outros como o Escolar Plus[10] disponibilizam um manual ensinando como instalar e utilizar sua versão *desktop* e um *login* e senha para quem desejar visualizar telas do sistema *online* como se fosse um Pai/Aluno (ver Figura 17).

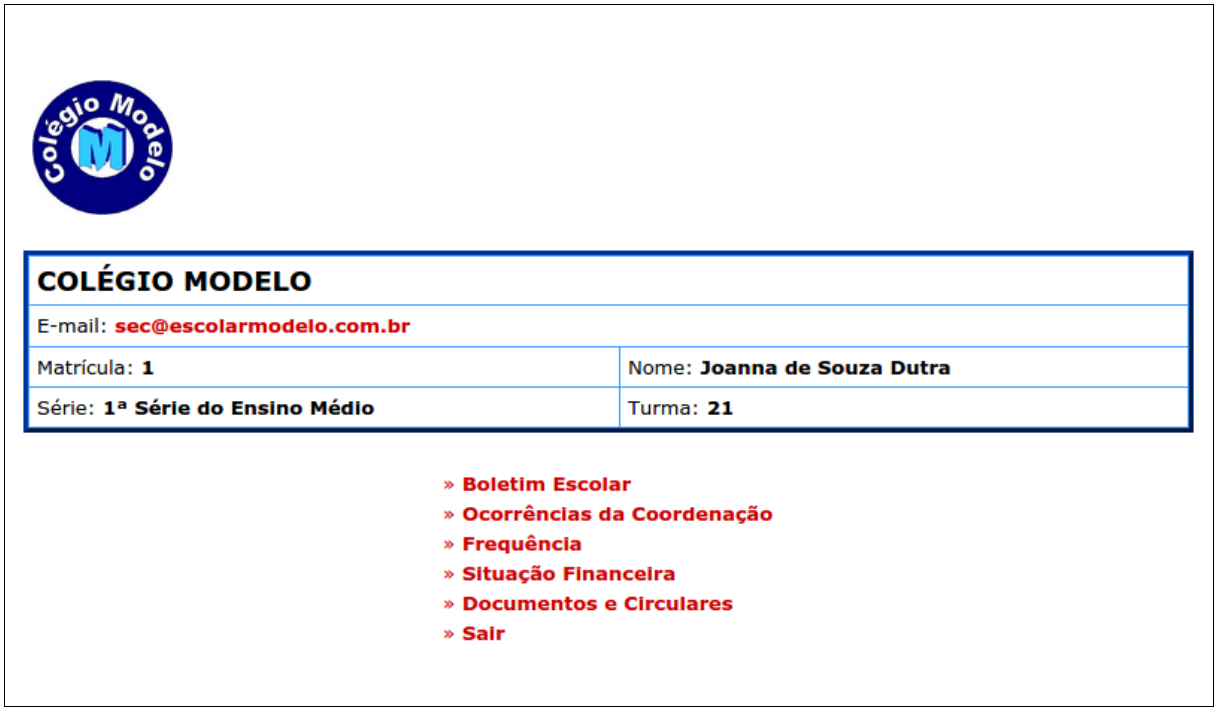


Figura 17 – Visão Pai/Aluno do Sistema Escolar Plus
Fonte:<http://www.escolarplus.com.br/online/principal.asp>

O DKSOFT[11], outro sistema que possui como principal versão a *desktop*, é um dos que mais se diferencia entre os demais analisados, já que ele tem como público-alvo apenas cursos livres profissionalizantes e cursos interativos de informática, e disponibiliza o seu sistema para *download*, sendo apenas alguns módulos *online*, servindo para integrar dados quando a empresa possuir filiais, e ainda, frequência, notas, parcelas quitadas e abertas ficam disponíveis para consulta, podendo a escola disponibilizar essas informações para os pais de seus alunos através de um site próprio, pois a DKSOFT não disponibiliza site institucional.

10 Escolar Plus - http://www.escolarplus.com.br/online/

11 DKSOFT - http://www.dksoft.com.br/dk*web*.html

Os sistemas SIA - Sistema Escolar[12] da Secretaria de Educação do Governo do Estado da Bahia e Secretaria Escolar Digital[13] da Secretaria de Educação do Governo do Estado de São Paulo, são diferenciados dos demais, sendo disponibilizados apenas para escolas da rede estadual de ensino dos referidos estados. Apenas o sistema Secretaria Escolar Digital oferece informações sobre suas funcionalidades no seu site.

No geral, as listas de funcionalidades são bem parecidas de um com o outro, geralmente divididas em módulos como financeiro, secretaria, portal Pais/Alunos, portal Professores. Nos módulos financeiros dos sistemas são oferecidas funcionalidades como contas a pagar e a receber, controle de mensalidades, gestão de contratos, emissão de boletos/carnês/recibos (com opção dos pais imprimirem segunda via) e alguns relatórios financeiros. Para os pais geralmente são disponibilizados o acompanhamento de notas, faltas e boletins, sendo esse o módulo que mais possui funcionalidades diferentes de um sistema para outro, onde um dá opção de acesso a material de aula e apoio, acompanhamento de calendário de aulas, outros disponibilizam relação de ocorrências da coordenação.

Tivemos acesso a uma versão de testes apenas do iScholar, que na versão de alunos e pais dão acesso a notas, faltas, ocorrências e à sua situação financeira com a escola de suas próprias casas. Seu Gestor Escolar é um *software* totalmente *online*, além disso, disponibilizam para os clientes um canal exclusivo onde podem registrar solicitações, acessar o *chat online* e tirar dúvidas.

A versão que nos foi disponibilizada é referente a área administrativa, que o responsável por uma escola teria acesso. Todas as suas funcionalidades estão dispostas no menu principal conforme mostra a Figura 18.

Seu menu é bastante completo, com várias funcionalidades bem distribuidas, sendo a principal forma de organização o objetivo principal de cada função, onde as funcionalidades que tratam de dados administrativos ficam dispostos em um menu, as funcionalidades que tratam dos dados da coordenação em outro, e assim por diante.

Nenhum outro sistema de gestão escolar nos disponibilizou acesso gratuitamente, sendo assim, não tivemos a oportunidade de testar tudo que estes gerenciadores podem oferecer. Foi procurado dentre os sistemas acima

12 SIA – Sistema Escolar - http://www.sec.ba.gov.br/siig/sistemaescolar/home.asp

13 Secretaria Escolar Digital - https://sed.educacao.sp.gov.br/Logon

mencionados e/ou ilustrados, algum que fornecesse informações visuais através de gráficos e só foi visto informações sobre isso nos módulos referente a finanças, sendo que, não foi possível visualizar os sistemas integralmente.

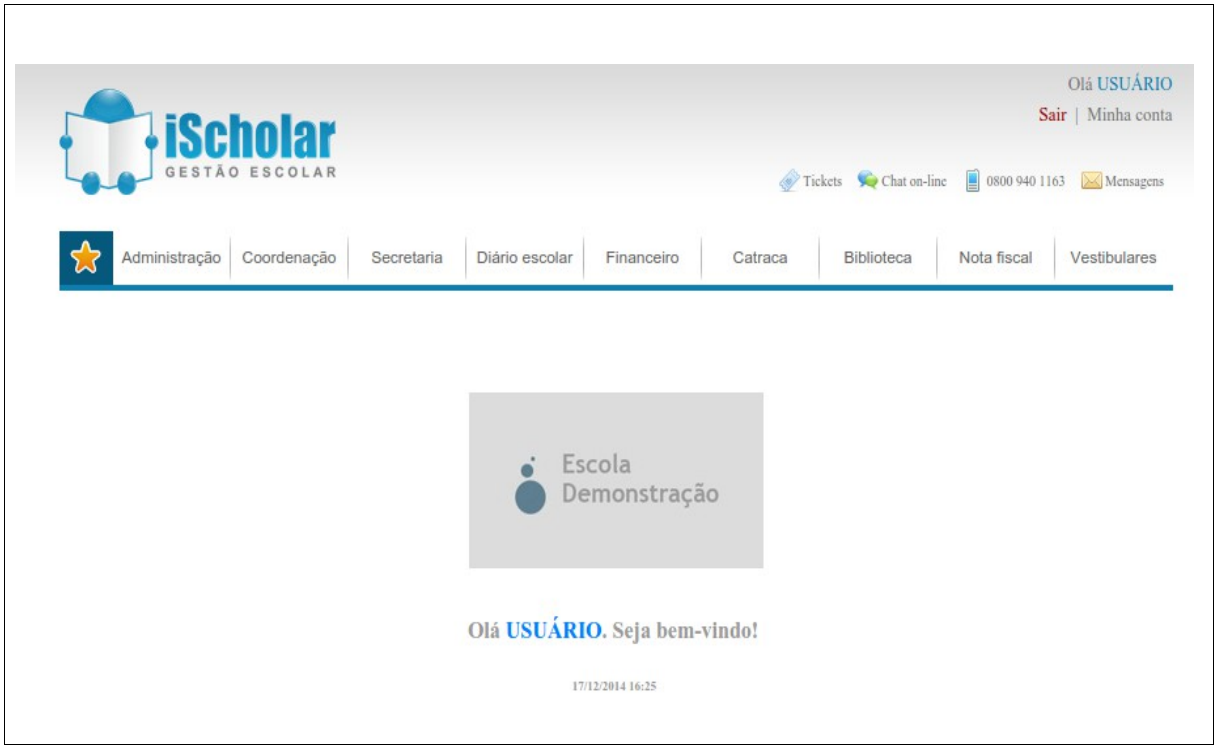


Figura 18 – Interface do usuário do sistema iScholar
Fonte:<http://experimente.ischolar.com.br>

3 SISTEMA DE GERENCIAMENTO ESCOLAR

O Mundo está cada vez mais necessitado de informações de acesso fácil e ágil, como evidencia Ramarkrishnan (2011) ao afirmar que os usuários necessitam simplificar tarefas de gerenciamento dos dados e a extração de informações úteis de forma oportuna a atender suas necessidades.

Com base no enunciado do autor evidenciado no parágrafo anterior, este projeto buscou desenvolver uma aplicação *web* que contribua com a eficácia das tarefas realizadas, mantendo a confiabilidade e o gerenciamento de informações de instituições de ensino que trabalhem do primeiro ao nono ano do ensino fundamental. Entretanto, a criação de uma aplicação não deve iniciar “de cara” com a codificação das rotinas que a compõe, muito pelo contrário, um *software* faz parte de todo um processo de engenharia previamente planejado e idealizado para resolver determinado problema do mundo real.

Este trabalho propôs elaborar um protótipo de um sistema para gerenciar informações de colégios da rede privada de ensino que trabalhem com o ensino fundamental, e sua produção deu-se seguindo diretrizes de ES, onde foi buscado informações da estrutura organizacional da instituição, análisado os requisitos, englobando principalmente os mais básicos (cadastro de séries, turmas, disciplinas etc.), confeccionando diagramas que auxiliaram durante todo processo de criação do projeto lógico de banco de dados, visualização das funcionalidades, principalmente aquelas indispensáveis a uma empresa do ramo educacional (como realizar as matrículas dos alunos, cadastrar funcionários, inserir as notas dos alunos etc.), gerando *scripts SQL (Structure Query Language)* para o projeto físico de banco de dados, além do desenvolvimento de *interfaces* que os usuários terão acesso através de páginas *web* e tecnologias correlacionadas.

3.1 Desenvolvimento da Aplicação

O *software* abordado neste projeto está incluso na categoria de *software* ou aplicação para *Web*, estando desde o início do projeto propenso a utilização de um servidor *Web,* sem necessidade de adaptações de *software* para este fim.

Entretanto, antes de iniciar o desenvolvimento, ou mesmo o levantamento de

requisitos e análises, foi trabalhado um planejamento para assegurar qualidade e padrões ao projeto. E o primeiro passo para iniciar o planejamento foi observar o funcionamento de alguns aplicativos já existentes sobre o tema proposto.

Com um conhecimento prévio sobre o que seria desenvolvido, buscou-se criar uma estrutura analítica do projeto e cronogramas para auxiliar no planejamento durante todo o processo de desenvolvimento e até mesmo gerar relatórios para o acompanhamento dos serviços.

3.1.1 Levantamento de Requisitos e Projeto

O levantamento de requisitos se deu a partir do momento que foi observado outros sistemas de gerenciamento escolar, vendo funcionalidades e os passos para se chegar até elas. Logo depois foi buscado uma instituição para se trabalhar o projeto. E o Instituto Educacional Frei Galvão, nos aceitou como desenvolvedores dentro de seu estabelecimento, onde tivemos a primeira reunião para entender como era o funcionamento da escola. Nesse primeiro encontro captamos apenas informações superficiais que nos ajudaram a preparar um projeto para gerenciar todo desenvolvimento.

Foram analisados sistemas de gerenciamento de projetos e adotado o *Project Plan Housatonic* 365[14], um sistema que assim como o *MS Project*[15] da *Microsoft* é um gerenciador de projetos, e permitem que as equipes de projeto vejam as tarefas, insiram quadros de horários e sinalizem problemas e riscos. O *Project Plan Housatonic* 365 pode ser utilizado gratuitamente via *web*, sem necessidade de instalação no computador. Conforme é exibido na Figura 19.

Uma vantagens que o *Project Plan* trouxe ao plano de desenvolvimento foi em forma de organização, simplificando o processo de visualizar o andamento do mesmo. Ele possui várias funcionalidades onde o gerente de projetos pode ou não dar direitos de edição para os demais membros da equipe de desenvolvimento e com esses direitos eles podem inserir horas, adicionar ou remover tarefas de quadros de horários, relatar o andamento das tarefas, adicionar novas tarefas, atribuir tarefas existentes e atribuir tarefas a outras pessoas da equipe, adicionar

14 Project Plan Housatonic 365 - https://www.projectplan365.com/

15 Microsoft Project - http://office.microsoft.com/pt-br/microsoft-project-*software*-de-gerenciamento-de-projetos-FX103472268.aspx

informações sobre os problemas e riscos do projeto e também permite que eles vinculem problemas e riscos a tarefas específicas no plano, com a facilidade de um sistema *web*.

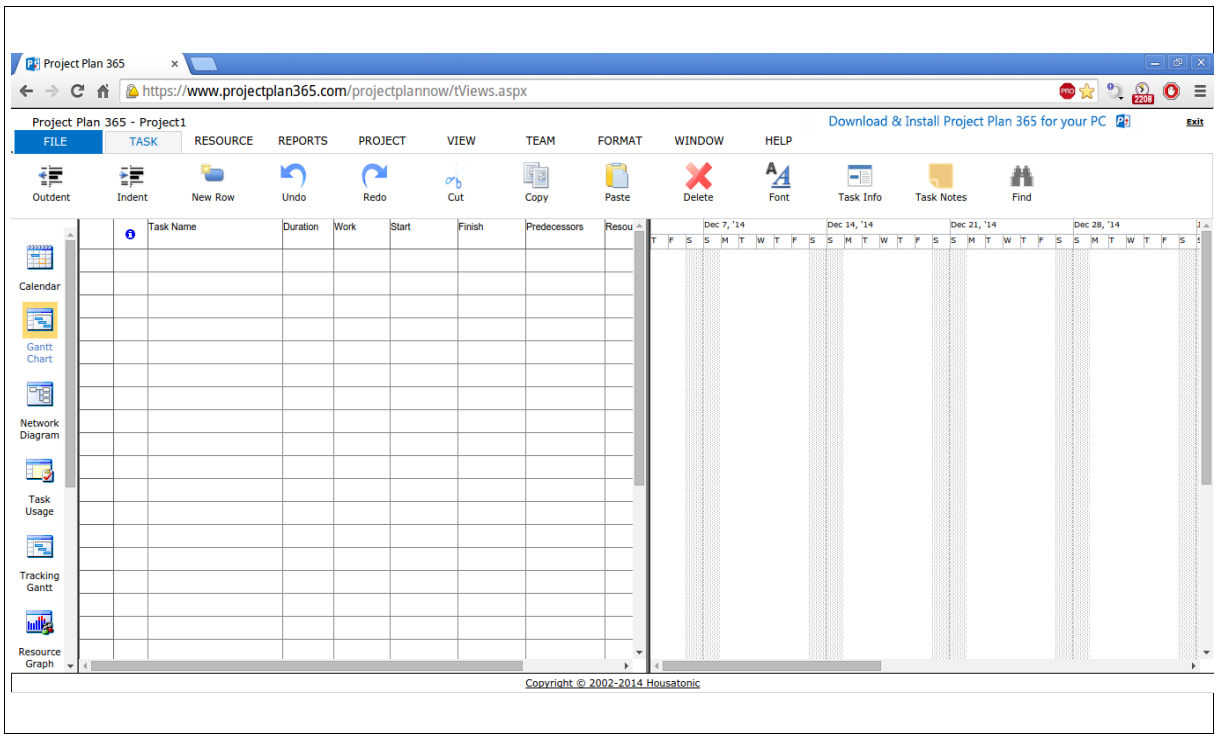


Figura 19 – Tela inicial do *Project Plan Housatonic* 365
Fonte:<http://www.projectplan365.com/projectplannow/tViews.aspx>

Além do que foi referenciado, o *Project Plan* conta ainda com versões para *Android*, *Iphone* e *Windows Phone*, proporcionando a liberdade para trabalhar e acompanhar os projetos mesmo em dispositivos móveis.

Após o gerenciador de projetos escolhido, foi definido como seriam realizados os trabalhos, dentro dos paradigmas de engenharia de *software* existentes, adaptando-os às necessidades do projeto.

E Como existem muitos paradigmas ou métodos que podem ser adotados no desenvolvimento de um *software*, a adoção do *Scrum* se deu devido a prioridade nos dias atuais da utilização de métodos de desenvolvimento ágil, já que constantemente ouve-se dizer que “o tempo é ouro” e quanto mais tempo a produção de um *software* levasse, menos eficiente ele poderia ser no mercado, que está constantemente evoluindo e sofrendo alterações.

Pois, embora hajam dados bastantes relevantes em detrimento da utilização de metodologias ágeis, dificilmente é utilizada apenas uma metodologia em um

projeto de desenvolvimento, ou ainda, sem adaptar essa metodologia a sua forma de trabalho. Como um projeto de *software* é formado por vários processos inclusos em um plano principal, tem a possibilidade de projetistas utilizarem metodologias diferentes em cada parte do trabalho.

E o nosso projeto teve essa caracteristica. Quando foi escolhido e utilizado o método *Scrum* como base de todo processo, tinhamos a convicção de que utilizariamos o modelo Incremental para criação do banco de dados, já que era uma prioridade nossa criar um banco de dados que atendesse todos os módulos que o sistema teria, e que sofresse poucas alterações com o andamento do projeto.

Piva (2010) afirma que o Modelo Incremental resulta da combinação do modelo linear (em cascata) com o de protótipos (evolutivo), tendo seu desenvolvimento dividido em etapas, denominadas incrementos, os quais conduzem aos aprimoramentos necessários, até que se chegue à versão final.

Além do mais, para *interface* do sistema foi utilizado o método espiral que de acordo com Pressman (2011) é um processo de *software* evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlados do modelo cascata.

Essa adaptação de utilizar mais de uma metodologia custou um tempo maior de dedicação aos levantamentos de requisitos iniciais e ao banco de dados cujo sofreu apenas pequenas atualizações durante o restante do processo.

Os primeiros levantamentos de requisitos foram focados na criação de um projeto de desenvolvimento, observando quais eram as principais necessidades da instituição, quais módulos poderiam ser desenvolvidos em primeira instância sem afetar o desenvolvimento das demais funcionalidades.

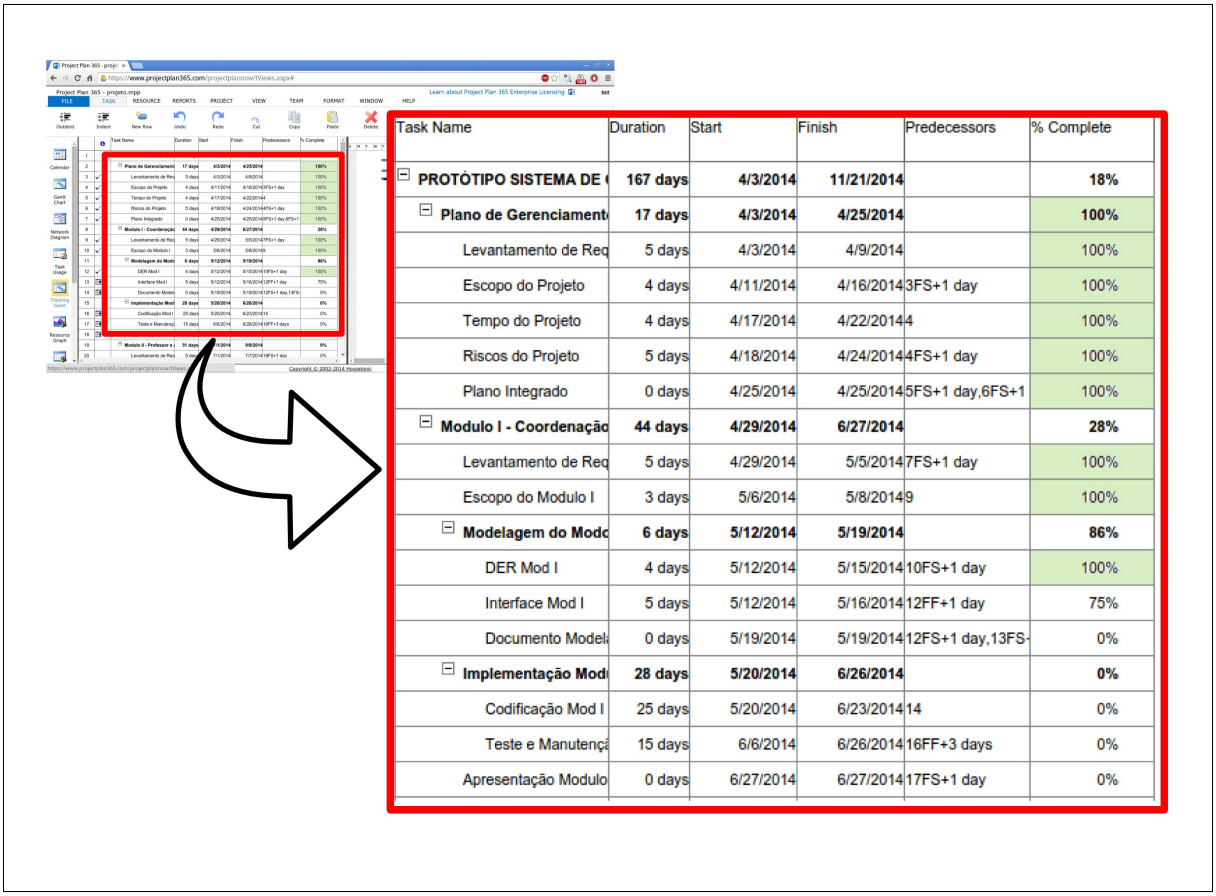
Na utilização do *Scrum*, isso foi onde o *Product Owner* (equipe do colégio) definiu as funcionalidades a serem desenvolvidas e suas prioridades, e ordenaram em detrimento das prioridades criando com isso o *product backlog*.

A partir deste momento com a utilização do Project Plan e definimos os tempos gastos em cada *sprint* a ser desenvolvido e elaboramos o projeto de desenvolvimento com o cronograma das atividades conforme Figura 20.

A primeira parte, o Plano de Gerenciamento, foi formado por levantamento de requisitos geral, escopo do projeto, definição de tempo, definição dos riscos, e o plano integrado, cujo agrupou todos os documentos elaborado em um só.

Algumas ações foram planejadas para serem preparadas simultaneamente,

como o planejamento de tempo e riscos, cujo a interferência de uma durante a criação do sistema afeta diretamente o resultado da outra, sendo proposto um dia a mais para os riscos, devido um processo de análise de risco e a busca por possíveis soluções para a condição, na hipótese dela acontecer.



Task Name	Duration	Start	Finish	Predecessors	% Complete
PROTÓTIPO SISTEMA DE	**167 days**	**4/3/2014**	**11/21/2014**		**18%**
Plano de Gerenciament	**17 days**	**4/3/2014**	**4/25/2014**		**100%**
Levantamento de Req	5 days	4/3/2014	4/9/2014		100%
Escopo do Projeto	4 days	4/11/2014	4/16/2014	3FS+1 day	100%
Tempo do Projeto	4 days	4/17/2014	4/22/2014	4	100%
Riscos do Projeto	5 days	4/18/2014	4/24/2014	4FS+1 day	100%
Plano Integrado	0 days	4/25/2014	4/25/2014	5FS+1 day,6FS+1	100%
Modulo I - Coordenação	**44 days**	**4/29/2014**	**6/27/2014**		**28%**
Levantamento de Req	5 days	4/29/2014	5/5/2014	7FS+1 day	100%
Escopo do Modulo I	3 days	5/6/2014	5/8/2014	9	100%
Modelagem do Modc	**6 days**	**5/12/2014**	**5/19/2014**		**86%**
DER Mod I	4 days	5/12/2014	5/15/2014	10FS+1 day	100%
Interface Mod I	5 days	5/12/2014	5/16/2014	12FF+1 day	75%
Documento Modela	0 days	5/19/2014	5/19/2014	12FS+1 day,13FS-	0%
Implementação Mod	**28 days**	**5/20/2014**	**6/26/2014**		**0%**
Codificação Mod I	25 days	5/20/2014	6/23/2014	14	0%
Teste e Manutenç	15 days	6/6/2014	6/26/2014	16FF+3 days	0%
Apresentação Modulo	0 days	6/27/2014	6/27/2014	17FS+1 day	0%

Figura 20 – Projeto de criação do protótipo do Sistema de Gerenciamento de Informações Escolares utilizando o Project Plan 365
Fonte: o Autor.

Posteriormente, foi elaborado um estudo de caso simplificado, do que já havia sido colhido no levantamento de requisitos inicial para que fosse elaborada a documentação inicial ou plano integrado, o que nos deu a possibilidade de fazer um acompanhamento detalhado de cada uma das etapas seguintes do processo.

Pudemos fazer um acompanhamento ainda mais detalhado do andamento do sistema com o *Project Plan*, que nos forneceu uma visualização através de um calendário mensal (ver Figura 21).

Em cada etapa de desenvolvimento, ou *sprint* de *backlog* existiu um período de tempo definido para levantamento de requisitos, remodelagem ou ajustes do MR, implementação, testes e entrega. Os novos levantamentos de requisistos que eram

feitos possibilitaram o acompanhamento do processo, verificando se o *sprint* estava dentro dos padrões pré-determinados e um regulamento, fazendo um realinhamento do desenvolvimento sempre que este estava fora de percurso.

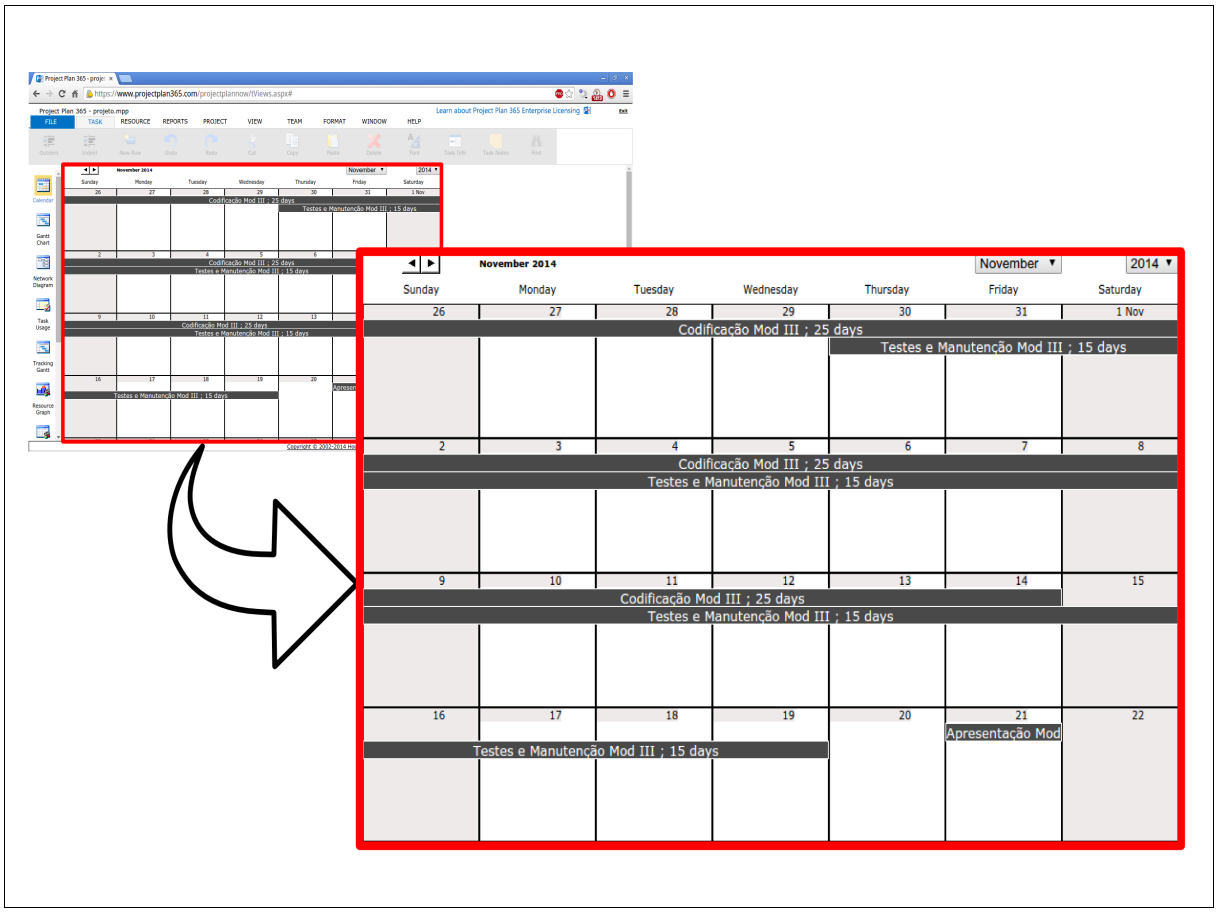


Figura 21 – Apresentação do calendário de atividades do Project Plan 365
Fonte: o Autor.

Piva (2010), diz que o levantamento de requisitos se dá a partir do momento que nos utilizamos de técnicas que buscam compreender e solucionar problemas encontrados no mundo real, onde avaliamos o tamanho do problema e o que se espera da solução. E Sommerville (2007) informa que os requisitos de um sistema não ficam restritos apenas aos processos técnicos, eles são diretamente influenciados pelas preferências, recusas e preconceitos dos usuários, além das questões políticas e organizacionais.

O novo MR que surgia, já estava pronto, então era transformado em linguagem SQL e depois recriado o Banco de Dados com as novas especificações, mantendo sempre o Banco mais atualizado em funcionamento.

A segunda *sprint* foi focada no gerenciamento do professor, como ele inseria as notas, as faltas e quais telas seriam visiveis para ele e as telas dos alunos, que na verdade é voltado para os pais ou responsáveis, já que o desenvolvimento foi definido para instituições de primeiro ao nono ano do ensino fundamental.

Encontramos alguns problemas no levantamento de requisitos por parte das entrevistas e observações, pois quando um funcionário do colégio discorria sobre as atividades que exercia na instituição, notavamos que eram ocultadas informações corriqueiras, não pelo fato da pessoa querer realmente deixar aquilo fora, mas por ser atividades tão comuns para o funcionário que ele não se dava conta que tinha ficado de fora.

Alguns dessas informações que não haviam sido relatadas eram de conhecimento nosso, ou por termos notado o acontecimento nas observações ou por já ter visto durante a análise dos sistemas existentes, então indagamos sobre o assunto com a equipe do colégio para relembrar que seriam informações importantes. Ainda assim nos deparamos com requisitos novos no momento da implementação, tendo que refazer algumas atividades, como alterações no Banco de Dados e formulários.

Foi o que aconteceu durante a coleta de informações para o funcionamento do lançamento de notas, informações como outros tipos de avaliações que alguns professores faziam ao final do ano letivo não foram repassados antes, e quando estavamos testando a função de gerar boletim foi informado que teriam essa avaliação, o que nos fez mudar o banco e a estrutura das páginas de avaliação.

Com isso, nesse *sprint* não foi possível concluir a parte de lançamento de faltas, o que foi realocado para o último módulo, juntamente com a parte de desenvolvimento dos gráficos de notas e faltas. Neste ponto do processo, foi preciso colocar em prática as orientações do desenvolvimento ágil, focando exclusivamente no que era necessário, esquecendo os outros módulos, procurando solução apenas para o que estava sendo desenvolvido no momento.

3.1.2 Modelo Relacional e o Banco de Dados

Foi no plano de gerenciamento, mais precisamente nos levantamentos de requisitos e criação do escopo que elaboramos a primeira versão do Modelo Relacional do Banco de Dados. Este foi criado para atender todos os campos, suas

próximas versões foram apenas o melhorando. Foi a partir deste modelo que tivemos a noção da dimensão que o sistema podia alcançar.

Desde o inicio da implementação do MR já tinhamos também definidos o SGBD de nossa aplicação, o MySQL, e a ferramenta utilização do *MySQL Workbench* da *Oracle* para a modelagem e também a implantação e testes no banco de dados.

A partir da segunda fase do projeto, foi que nos preocupamos com a normalização das tabelas do nosso modelo relacional. Damas (2007) informa que o processo de normalização consiste em submeter o esquema das relações a uma bateria de testes com o objetivo de determinar qual estado que o esquema se encontra e evitar assim duas das principais ameaças que assolam os bancos de dados. A redundância de dados e a existência de valores *NULL*.

Nesta fase estavamos focados no desenvolvimento do módulo de gerenciamento da secretaria, onde foi reformulado o banco evitando que existissem campos onde não eram chaves e causavam dependencias da tabela, criando assim uma outra tabela para esses atributos conforme Figura 22.

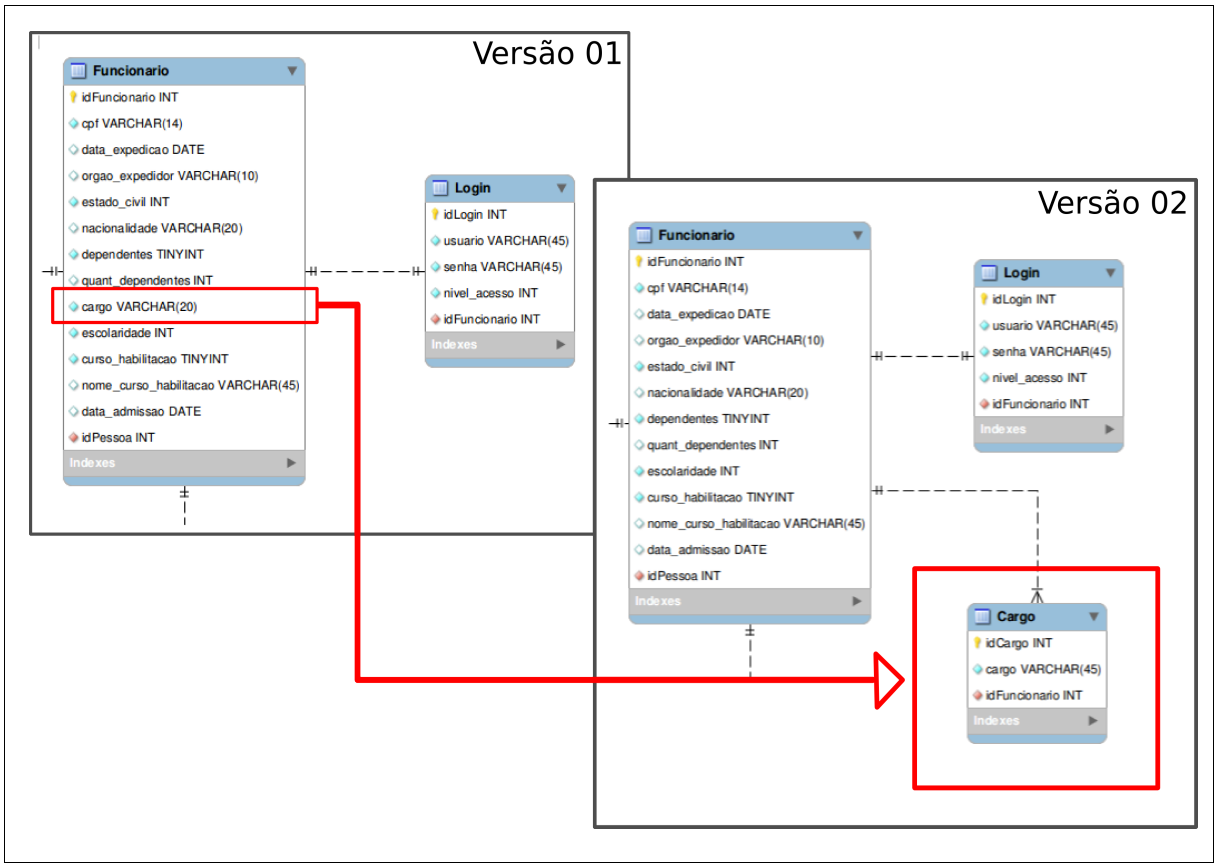


Figura 22 – Evolução do MR do Banco de Dados
Fonte: o Autor.

Ainda neste módulo foi criado o cadastro de disciplinas do colégio, as séries que o colégio pode ofertar e quais turmas cada série tem, assim já podiamos fazer o cadastro de um professor para uma turma específica (Ver Figura 23).

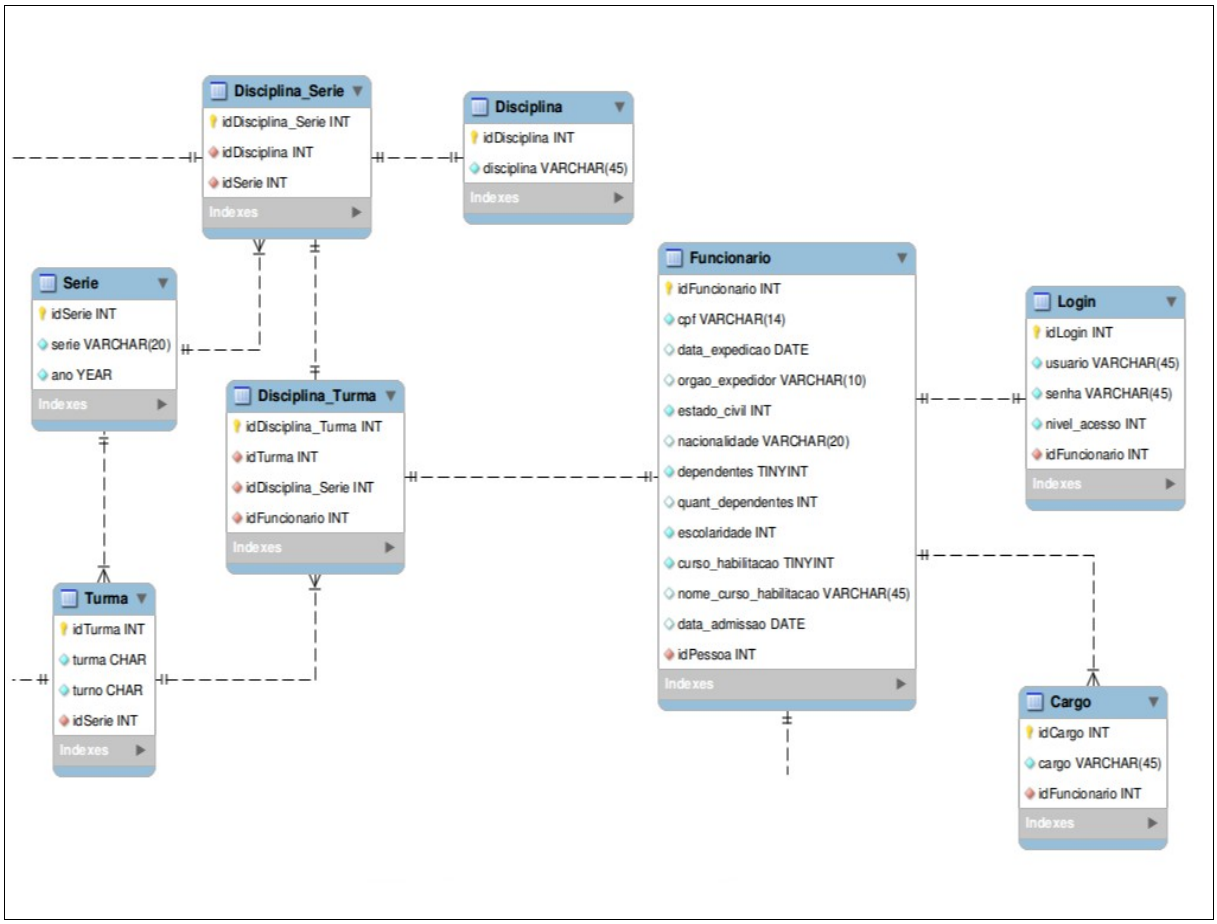


Figura 23 – Segunda versão do MR do Banco de Dados – Funcionalidades da Secretaria
Fonte: o Autor.

Depois de entregue este *sprint* passou-se a trabalhar no cadastro de notas, procurando aperfeiçoar o maximo o banco para evitar duplicação de dados, já que este é um dos principais dados a serem guardados no sistema, que não ficam disponíveis apenas no ano atual, mas são conservados para a confecção do histórico escolar, após saída do aluno da instituição.

Foi muito difícil trabalhar com o *Scrum* em banco de dados, pois o que poderia facilitar ao pensar somente no ponto em questão podeira faltar ao final do projeto, e como o banco de dados é um dos elementos mais importantes dentro de um sistema, uma alteração poderia destruir todo um trabalho já produzido. E por

mais que quiséssemos concluir cada etapa dentro do prazo, não podiamos deixar lacunas erroneas no banco de dados.

A Figura 24 mostra como ficou o MR, sendo que as notas ficaram para o próximo *sprint*, contendo neste MR apenas as notas.

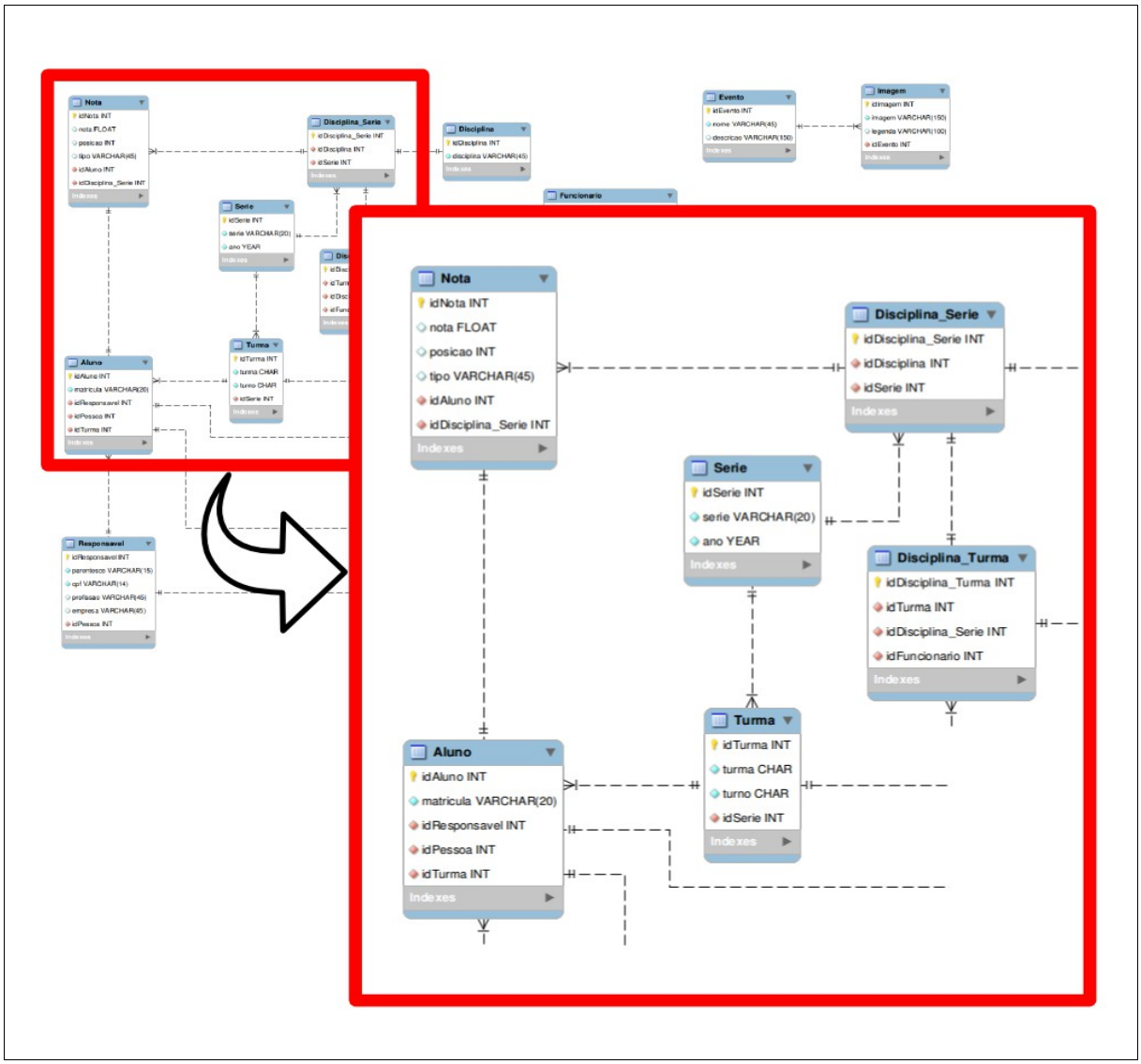


Figura 24 – Modelo Relacional - Setor de nota dos alunos
Fonte: o Autor.

O Banco de Dados ainda sofreu várias outras alterações antes da versão final, sendo aperfeiçoado a cada *sprint* finalizado e notando que eram necessários mais ajustes a cada novo levantamento de requisitos. O resultado de todo processo de produção agradou, e a versão final do BD, pelo menos para a utilização do prótótipo foi concluida.(Figura 25, MR completo para o protótipo).

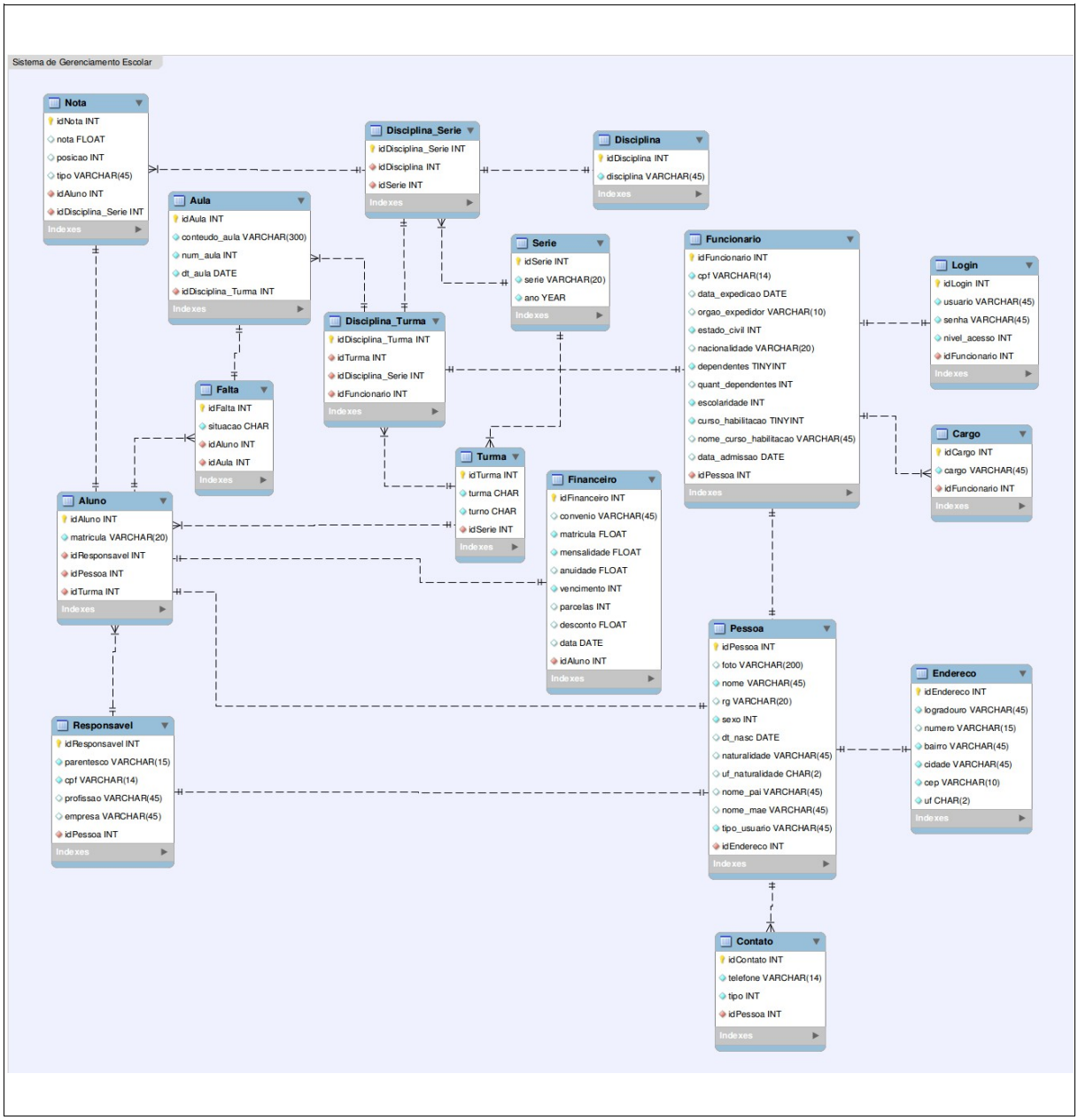


Figura 25 – Modelo Relacional (completo) do protótipo do Sistema de Gerenciamento de Informações Escolares

Fonte: o Autor.

3.1.3 Implementação do HTLM5, CSS3 e JavaScript

Para o desenvolvimento de um *software*, são necessários cuidados, principalmente nas primeiras fases, onde o levantamento de requisitos, tanto funcionais como não funcionais e as análises destes devem ser bem definidas para que abranjam todas as necessidades do cliente.

Mas após ter todo esse material em mãos, partimos para o desenvolvimento da aplicação, onde um ponto importante foi sempre estarmos em constante comunicação com a equipe do colégio, reuniões não só nas etapas de levantamento de requisitos, mas também nas apresentações das *sprints* de *backlog* onde os funcionários da instituição davam grande contribuição para o continuar do projeto.

O texto abaixo, retirado do livro de Sabbagh (2013), confirma e enfatiza melhor o que foi discorrido sobre o desenvolvimento de um *software* quando o cliente faz parte da equipe.

> Por melhores que sejam as técnicas de levantamento de requisitos utilizadas, não há como se definir de antemão o produto em detalhes. Assim, uma longa fase de detalhamento no início do projeto gera um grande desperdício. Os clientes aprenderão sobre o que está sendo desenvolvido ao longo do trabalho, à medida que veem ou utilizam as partes do produto demonstradas ou entregues. Assim, busca-se, a partir de seu *feedback*, desenvolver e entregar o produto certo. Da mesma forma, quando decisões equivocadas são tomadas, o *feedback* dos clientes e demais partes interessadas rapidamente gera visibilidade sobre o problema e permite a correção do rumo (SABBAGH, 2013, p.5).

Um detalhe que está intrinsecamente entendido e que não é mencionado pelos autores é o pós-desenvolvimento, onde é realizado um treinamento com o cliente ou disponibilizado um tutorial para utilização do produto desenvolvido. Quando o cliente que trabalhará com o sistema está em constante comunicação com a equipe de produção, ele já tem um conhecimento prévio de grande valia para a utilização do sistema, pois já estará familiarizado com as telas e funcionalidades da aplicação, preocupando-se apenas na inserção de conteúdo.

Outro fator relevante, é que a qualidade de um sistema não é medida apenas em função de sua programação ou algoritmos utilizados, um sistema pode ter excelentes algoritmos codificados em seu *software*, e ser de péssimo desempenho por defeito de desenho de seu *hardware*, rede ou banco de dados. Cada um destes elementos pode pôr a perder a confiabilidade e a usabilidade do sistema (FILHO, 2001).

Então, quando vai-se desenvolver um *software*, precisa-se analisar onde e como ele será utilizado, evitando assim que meios externos prejudiquem a desenvoltura e usabilidade do sistema, mantendo a qualidade na qual o *software* foi pensado.

E como já mencionado, este sistema foi totalmente projetado para funcionar

em um servidor *web*, sem a necessidade de aquisição de equipamentos pela empresa.

Para criar páginas *web* ou qualquer outra *interface* de usuário, o Pressman (2011) defende três regras consideradas por ele como regras de ouro, onde diz que o usuário sempre deve estar no comando, sempre se deve reduzir a carga de memória do usuário, e tornar a *interface* consistente.

Na maioria das vezes os programadores buscam facilitar a vida do cliente e seu modo de interação e o resultado disso pode muitas vezes até ser uma *interface* fácil de ser construída, mas frustrante do ponto de vista do usuário. Deve-se então colocar o usuário a frente de todo projeto de *interface*, deixando-o aprovar telas e sequências para utilização.

Pressman (2011) indica que quanto mais o usuário tiver que se lembrar de sequências e ações que devem ser tomadas, mais sujeita a erros será a interação com o sistema. E por essa razão deve-se fazer uma *interface* bem desenhada para as ações do usuário, mas nada disso deve atrapalhar na solidez das informações apresentadas. Todas as informações visuais devem ser organizadas e com mecanismos de navegação para passar de uma tarefa a outra de maneira consistente.

Outra questão bastante relevante para o nosso sistema, é que foi priorizado a usabilidade do mesmo, sendo que o usuário conseguisse sempre o que queria sem ter que passar por muitas páginas ou cliques, pois Pressman (2011) afirma que a complexidade do sistema está proporcionalmente ligada ao número médio de cliques que o usuário precisa dar para chegar a uma função ou conteúdo do site.

Com isso definimos que um dos requisitos do sistema era que todos os usuários deveriam conseguir realizar suas tarefas com no máximo quatro cliques do *mouse*, o que pode ser visto na Figura 26 com as atribuições do Secretário. Mostrando ainda que telas simples não deixam a página sem estilo, e ainda agilizam o trabalho.

As primeiras telas do sistema foram um pouco mais trabalhosas, pois tivemos que criar um design que atendesse todas as necessidades e não dificultasse os acessos. Os próximos módulos desenvolvidos ganharam agilidade, já que não era mais necessário ficar imaginando como apresentar as funcionalidades ao usuário, precisando apenas inserí-las.

Isso está voltado ao reaproveitamento de código, onde se preparou a base

da página e a partir dela eram feitas todas as outras.

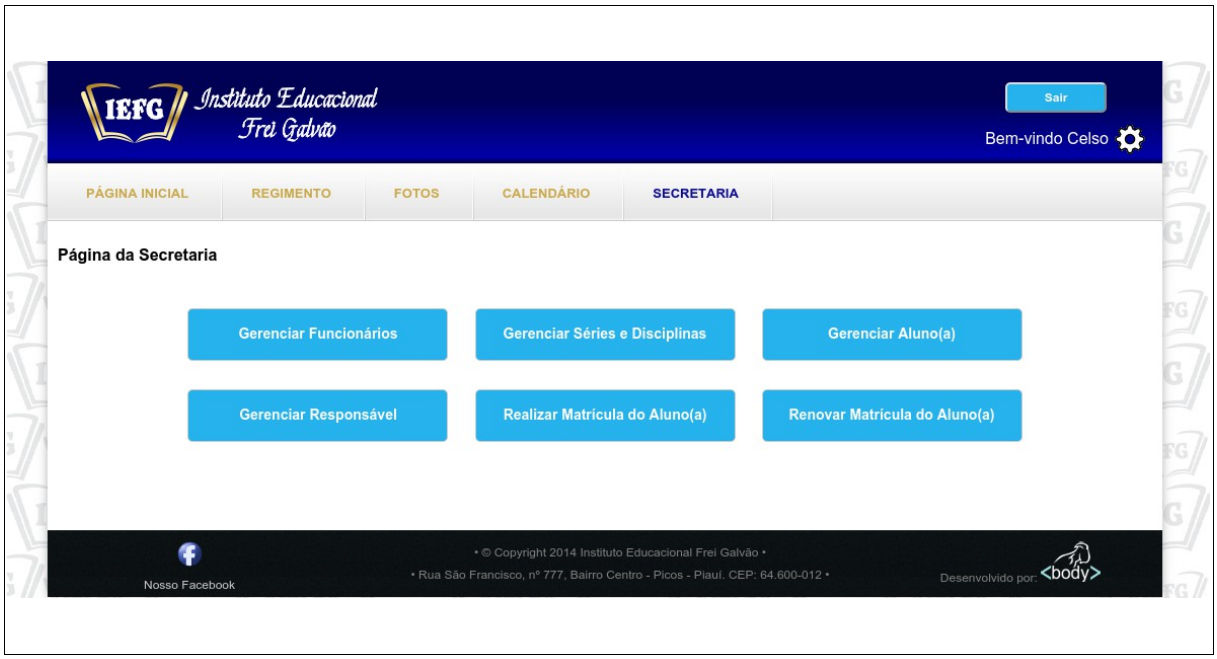


Figura 26 – Tela do sistema – Módulo *Secretaria*
Fonte: o Autor.

Desta forma, ficou definido o modelo da Figura 26 para todas as páginas iniciais, de usuário logado (professor e sercretaria) e páginas com abas para quando o usuário estiver dentro de uma das funcionalidades da página inicial (ver Figura 27).

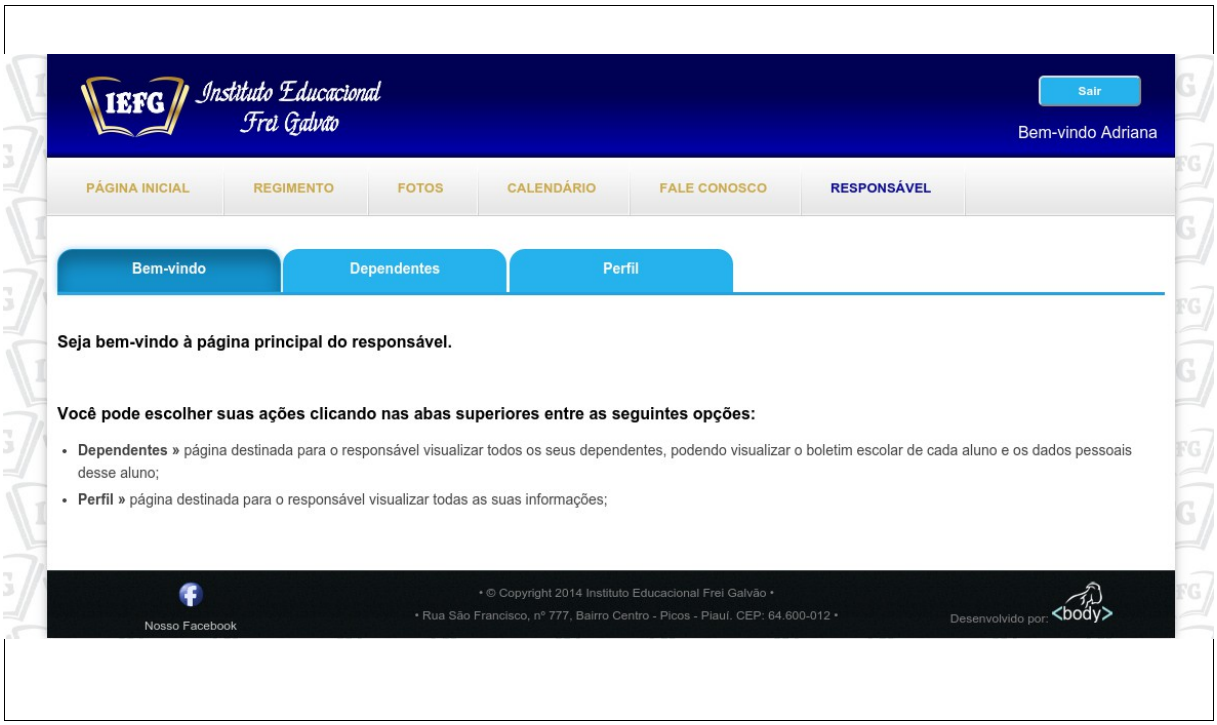


Figura 27 – Tela do sistema – Módulo Responsável
Fonte: o Autor.

3.1.4 Embutindo PHP na HTML

Por fim, chegamos aos “últimos passos” no desenvolvimento de um sistema *web*, a vinculação entre o banco de dados e as informações que são inseridas pelo usuário. Utilizando o PHP, os sites tornam-se mais dinâmicos e com possibilidade de obter informação em tempo real.

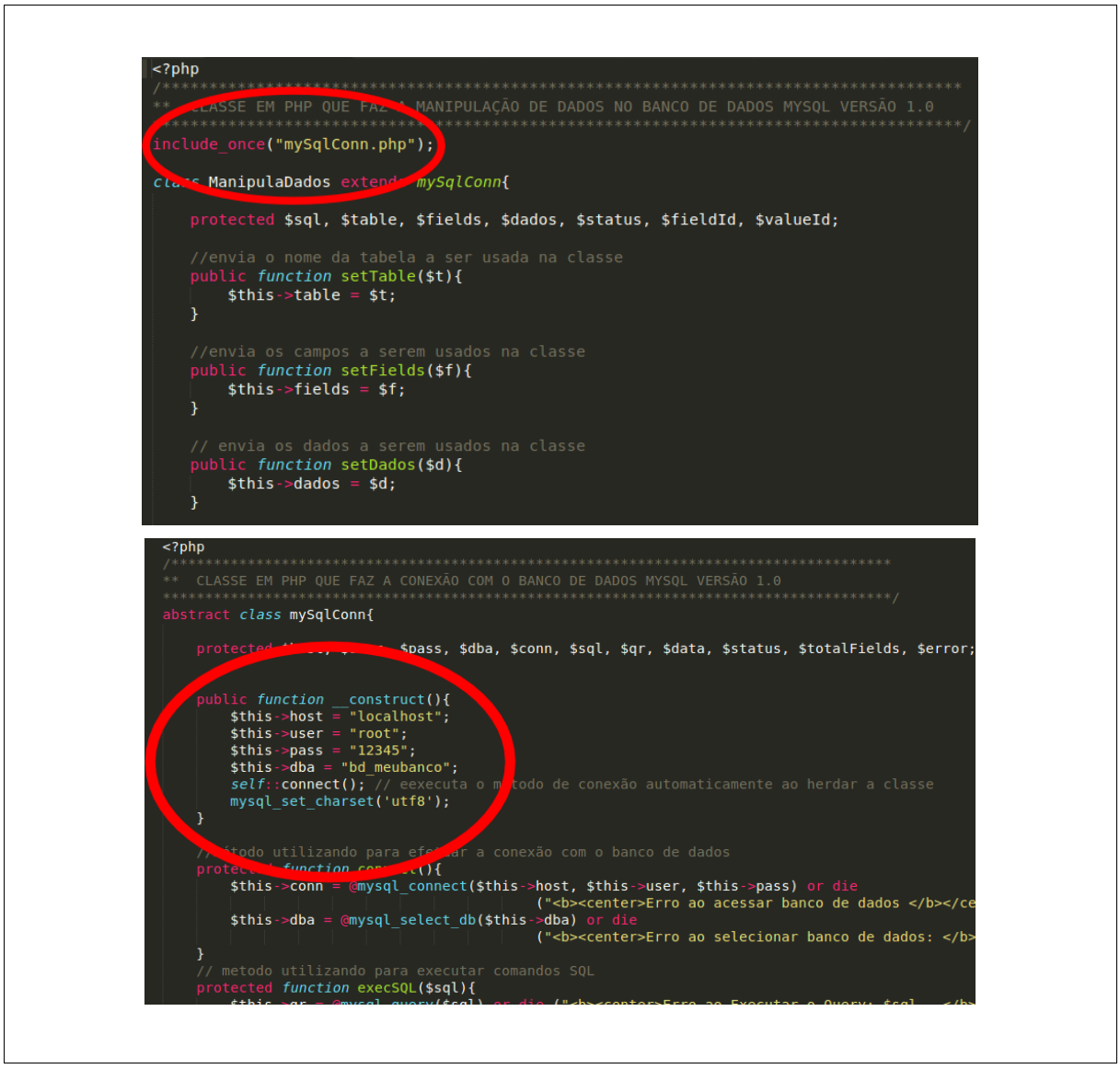


Figura 28 – Arquivos PHP responsáveis pela manipulação e conexão com o banco
Fonte: o Autor.

E na nossa aplicação ainda faltavam as ligações das páginas *web* com o banco de dados, o que foi realizado através de arquivos com a extensão “.php” contendo as principais funcionalidades que o banco de dados aceita, ou seja, aqui encontram-se as funções de *inserts*, *deletes* e *updates*. Só que para conseguir realizar essas funções precisou também da existencia de uma conexão com o banco e para isso, outro arquivo ".php" foi cirado e chamado através da função *include_once();*. Essa chamada pode ser vista na Figura 28, juntamente com o arquivo que faz a conexão abaixo.

Com essas duas páginas em funcionamento já era possível conectar e manipular os dados através de funções PHP, partindo então para a modificação das páginas estáticas criadas com o HTML e transformando-as em dinâmicas.

As funcionalidades de conexão e manipulação também são encaradas como um suspiro para o desenvolvedor *back-end*, já que com elas funcionando da forma correta, as demais páginas não terão mais dificuldade para enviar informações ao banco de dados.

Esses pontos curingas do desenvolvimento devem ser pensados pelo gerente de projetos, que ao definir prazos para cumprimento da criação de um sistema, possa ganhar tempo que podem significar muito quando se deparam com erros que geralmente tomam muito tempo para serem encontrados e corrigidos, principalmente quando não estão visíveis ao programador.

No nosso caso, nos deparamos com um erro até comum, porém difícil de ser detectado, ainda mais quando a experiência em programação *web* não é tão grande ainda. Ao desenvolver uma funcionalidade com um editor de texto, testá-la e notar que está funcionando corretamente, passamos para outro editor de texto e inserimos mais campos, e acabou que a funcionalidade já testada não funcionava mais da forma correta.

Os livros base que nos davam segurança no que faziamos não traziam nada a respeito, o que nos deixou ainda mais preocupados com a situação, o que nos levou a documentação dos editores de textos e encontramos a diferença na configuração de linguagem padrão dos caracteres de cada editor. O *notepad++* tem como padrão “*UTF8 without BOM*” e o *Sublime Text 2* apenas “*UTF-8*” sendo necessário a reescrita dos textos para resolver os problemas. A Figura 20 mostra a configuração padrão de caracteres no *sublime text 2*.

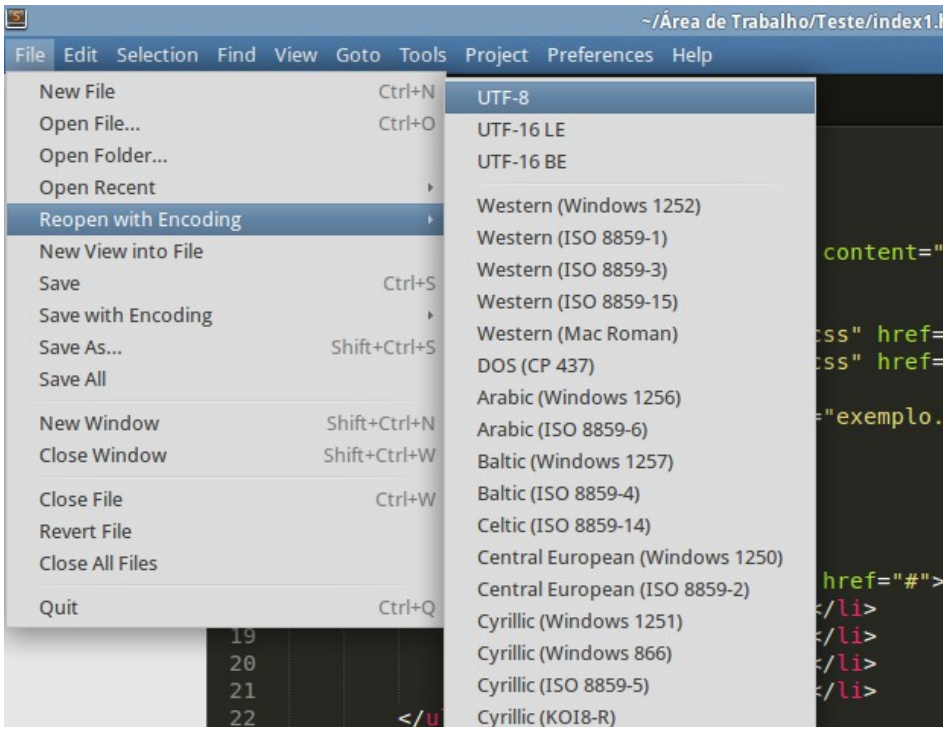


Figura 29 – Configuração padrão de caracteres do Sublime Text 2
Fonte: o Autor.

Com problemas superados pudemos concluir o desenvolvimento das funcionalidades de mais um *sprint*, agora o Módulo II já pôde ser entregue, pois esse já havia sido implementado e estavamos apenas fazendo correções de erros encontrados nos testes.

E dependendo das necessidades que a escola venha a ter futuramente poderá ganhar novos aperfeiçoamentos, como um gestor financeiro que, apesar de colher dados sobre os valores de matrícula dos alunos, não foi implementado.

4 RESULTADOS

O protótipo do sistema de gerenciamento de informações escolares foi desenvolvido a partir das informações colhidas durante todo o período de pré-desenvolvimento e desenvolvimento, atendendo as necessidades básicas do que foi proposto. Foi desenvolvido inteiramente para *web*, contendo menus simples e objetivos.

Foram dedicados mais de 170 dias para o desenvolvimento de todas as funcionalidades sugeridas, deixando o sistema utilizável e com boa aceitação nos testes realizados, porém algumas funcionalidades só poderão ser realmente testadas com o tempo de uso, onde a quantidade de dados existentes poderão gerar informações relevantes ao processo de funcionamento da instituição. Tendo em consideração ainda que a conclusão do projeto se deu justamente ao término do período letivo, só poderemos saber a real aceitação do sistema com o passar do tempo.

Foram entrevistados um total de 15 (quinze) responsáveis de alunos que acessaram o sistema pela internet. E como resultado tivemos a satisfação pela informação de forma prática e sem a necessidade de deslocamento até a escola.

Foram desenvolvidos no módulo de secretaria as funcionalidade:

- Cadastro de Disciplinas – inserindo cada disciplina que a instituição possui na sua grade curricular;
- Cadastro de Séries – onde é possível cadastrar todas as séries que o colégio trabalha, além de informar quais disciplinas pertencem a série inserida;
- Cadastro de Turmas – aqui é selecionado a série e montado uma turma, informando horário de funcionamento;
- Cadastro de Professores – cadastro de todas as informações pessoais do professor como informando quais são as turmas que ele irá estar vinculado;
- Matriculas e Renovação de Matriculas de Alunos – duas funcionalidades que diz respeito a inserção de dados dos alunos e responsáveis, depois do aluno matriculado, para renovar a matricula no ano seguinte acessa o menu Renovação e basta inserir os dados referentes a nova série e turma que este estará vinculado;
- Emissão de Boletins – emissão de boletins individuais do aluno, basta escolher a série, a turma, aluno e imprimir;
- Gerenciamento de Dados Pessoais – acesso aos seus dados cadastrais para correção de cadastro, e/ou troca de senha de acesso.

Para o módulo dos professores existe:

- Lançamento de Notas – o professor tem acesso a uma lista de disciplinas no qual possui vínculo e basta clicar em uma para aparecer a relação de alunos e possibilitá-lo inserir as notas;
- Visualização de Notas com Gráficos(médias do aluno, turma e colégio) – selecionando um aluno ele pode visualizar a média do aluno em comparação a média da turma e do colégio, além da ultima nota da disciplina selecionada, assim pode ver se o aluno melhorou ou diminuiu sua nota;
- Lançamento de Atividades – o professor tem acesso ao campo para cadastrar quantas aulas foram ministradas em uma data e inserir o assunto referente a data selecionada. A data pertinente a atividade cadastrada tornar-se-á disponível para lançamento das faltas dos alunos;
- Lançamento de Faltas – campo só fica habilitado caso haja atividades cadastradas para uma determinada data no mês buscado;
- Visualização de Faltas com Gráficos (porcentagem do número de faltas de um aluno) – apresentado para visualização um grafico de pizza constando a quantidade de faltas e presenças do aluno selecionado;
- Gerenciamento de Dados Pessoais – acesso aos seus dados cadastrais para correção de cadastro, e/ou troca de senha de acesso.

Para pais e alunos:

- Visualização de Notas com Gráficos (médias do aluno, turma e colégio) – pode ser visualizado a média do aluno em comparação a média da turma e do colégio, além da ultima nota da disciplina selecionada, assim pode ver se o aluno melhorou ou diminuiu sua nota;
- Visualização de Faltas com Gráficos (porcentagem do número de faltas)– apresentado para visualização um grafico de pizza constando

a quantidade de faltas e presenças do aluno selecionado;

- Impressão de Boletins – emissão de boletins individuais do aluno, caso tenha mais de um dependente cadastrado precisa selecionar primeiro o aluno.

Todos esses módulos e funcionalidades foram testados dentro do período de desenvolvimento, aguardando agora apenas colocar em pleno funcionamento para verificar as reais necessidades de manutenção e aperfeiçoamento.

5 CONCLUSÃO

A informática é uma área de atuação bastante abrangente, com diversas possibilidades de mercado, e para os mais variados gostos. Em apenas um projeto tivemos a oportunidade de conhecer diversos campos de atuação, como a gerência de projetos, a engenharia de *software*, engenharia de requisitos, gestão de banco de dados, design gráfico, programador front-end, programador *back-end* dentre muitas outras oportunidades. Na qual, tivemos a possibilidade de conhecer melhor dentro de um projeto real e ainda o contentamento de após a conclusão, vê-lo funcionar e saber que cumprimos nossos objetivos. Desenvolver um protótipo de um sistema de gerenciamento de informações escolares totalmente *web* não foi uma das tarefas mais simples, a cada novo requisito havia planejamento, reestruturação do banco de dados, reorganização dos formulários e paginas *web* e mudanças no php, mas com certeza, nos gerou muito aprendizado. Tendo em vista que a programação *web* está crescendo mais e mais a cada dia, abre um leque ainda maior de opções para quem deseja aprofundar-se nesse setor. No entanto, nem todas as abordagens constantes no protótipo foram testadas em um ambiente real, podendo assim apresentar algumas falhas que poderão ser corrigidas em trabalhos futuros e com o amadurecimento da ideia. E, visto que, é praticamente impossível prever como acessaremos a *internet* no futuro, principalmente com a quantidade de dispositivos existentes, e com tamanho de telas e resoluções variadas, o Design Responsivo, que trabalha cada página para se ajustar a tela independente do tamanho ou configuração, acaba sendo um dos poucos mecanismos que podem garantir uma assessibilidade adaptável em todas essas tecnologias.

REFERÊNCIAS

ANGELOTTI, Elaini Simoni. **Banco de dados.** – Curitiba: Editora do Livro Técnico, 2010.

BECK, Kent et al. **Manifesto para Desenvolvimento Ágil de *Software*.** Disponivel em: <http://agilemanifesto.org/iso/ptbr/manifesto.html> Acesso em: 05 nov. 2014.

BENTO, Evaldo Junior. **Desenvolvimento *Web* com PHP e MySQL.** São Paulo: Casa do Código, [2012?].

DA ROCHA, Ana Regina Cavalcanti et al. **Qualidade de *Software*: Teoria e Prática.** São Paulo: Pearson, 2004.

DA SILVA, Alberto Manuel Rodrigues; VIDEIRA, Carlos Alberto Escaleira. **UML, Metodologias e Ferramentas CASE.** Lisboa: Centro Atlântico, Lda., 2001.

DAMAS, Luís. **SQL, structured query language.** 6ª Edição – Atualizada e Ampliada. Rio de Janeiro: LTC, 2007.

FERREIRA, Elcio; EIS, Diego. ***HTML*5 - Curso *W3C* Escritório Brasil.** [S. l.]: [2010?]. Disponivel em: <http://www.*w3c*.br/pub/Cursos/CursoHTML5/html5-*web*.pdf> Acesso em: 20 nov. 2014.

FILHO, Wilson de P. Paula. **Engenharia de *Software*: fundamentos, métodos e padrões**. Rio de Janeiro: Editora LTC, 2001.

GOMES, André F. **Agile: Desenvolvimento de *software* com entregas frequentes e foco no valor de negócio.** São Paulo: Casa do Código, 2013.

GUERRA, Ana Cervigni; COLOMBO, Regina Maria Thienne. **Tecnologia da Informação: qualidade de produto de *software*.** Brasilia: PBQP, 2009.

K19. **Desenvolvimento *Web* com *HTML*, *CSS* e *Javascript*.** [2013]. Disponível em: <http://www.k19.com.br/cursos/desenvolvimento-*web*-com-html-*css*-e-javascript> [S. l.] Acesso em: 03 set. 2014.

MANFIO, Marcel Carvalho. **Engenharia de *Software*: uma nova abordagem.** Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Instituto Municipal do Ensino Superior de Assis – IMESA e Fundação Educacional do Município de Assis – FEMA, Assis-SP, 2012. [Orientador: Luiz Carlos Begosso].

PIVA, Gustavo Dibbern. **Informática, análise e gerenciamento de dados.** São Paulo : Fundação Padre Anchieta, 2010.

PRESSMAN, Roger S. **Engenharia de *software* [recurso eletrônico]: uma abordagem profi ssional.** 7ª Edição – Dados eletrônicos. – Porto Alegre: AMGH, 2011.

RAMARKRISHNAN, Raghu. **Sistemas de Gerenciamento de Banco de Dados.** [recurso eletrônico] – 3ª Edição. Porto Alegre: AMGH, 2011. Disponivel em: <http://books.google.com.br/books?id=COUJpkH5v38C&printsec=frontcover&hl=pt-BR> Acesso em: 10 out. 2014.

RODRIGUES, Andréa. **Desenvolvimento para *Internet*.** Curitiba: Editora do Livro Técnico, 2010.

SABBAGH, Rafael. ***Scrum*: Gestão ágil para projetos de sucesso.** São Paulo: Casa do Código, 2013.

SOMMERVILLE, Ian. **Engenharia de *Software* – 8a Edição.** São Paulo: Pearson Education: Addison-Wesley Brasil, 2007.

WELLING, Luke; THOMSON, Laura. **PHP E MySQL – Desenvolvimento *Web*.** Rio de Janeiro: Elsevier, 2005.

APÊNDICES

APÊNDICE A – TELAS DO SISTEMA

Imagem 1: Secretaria - Página Principal

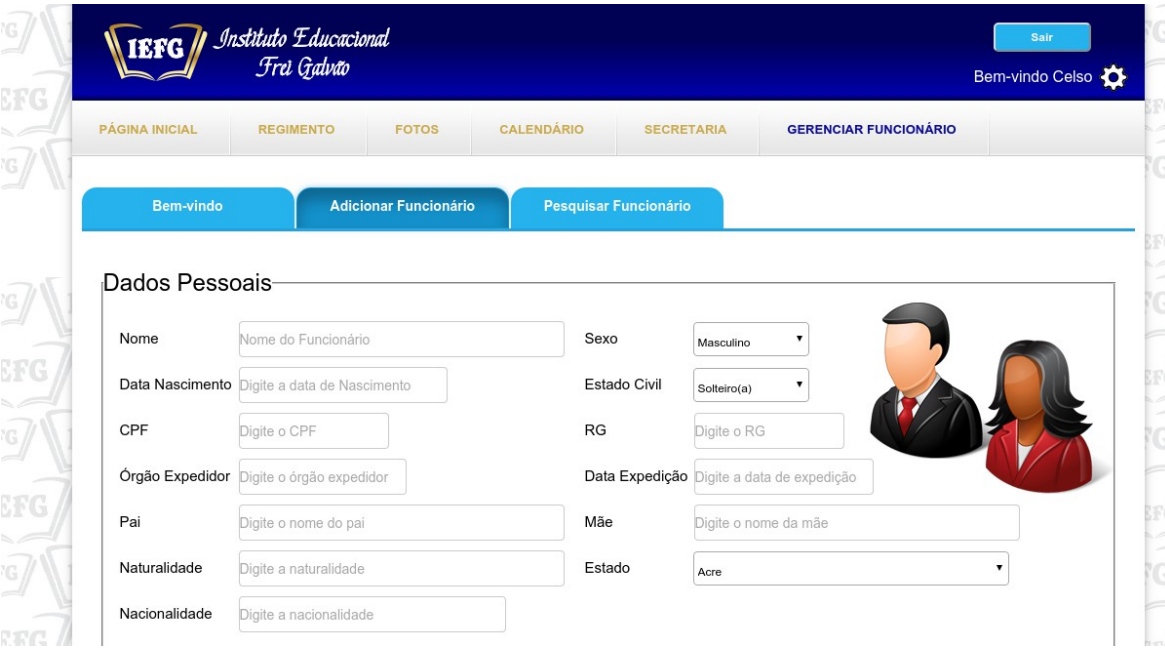


Imagem 2: Secretaria - Cadastrar de Funcionários

Imagem 3: Secretaria - Listar Funcionários

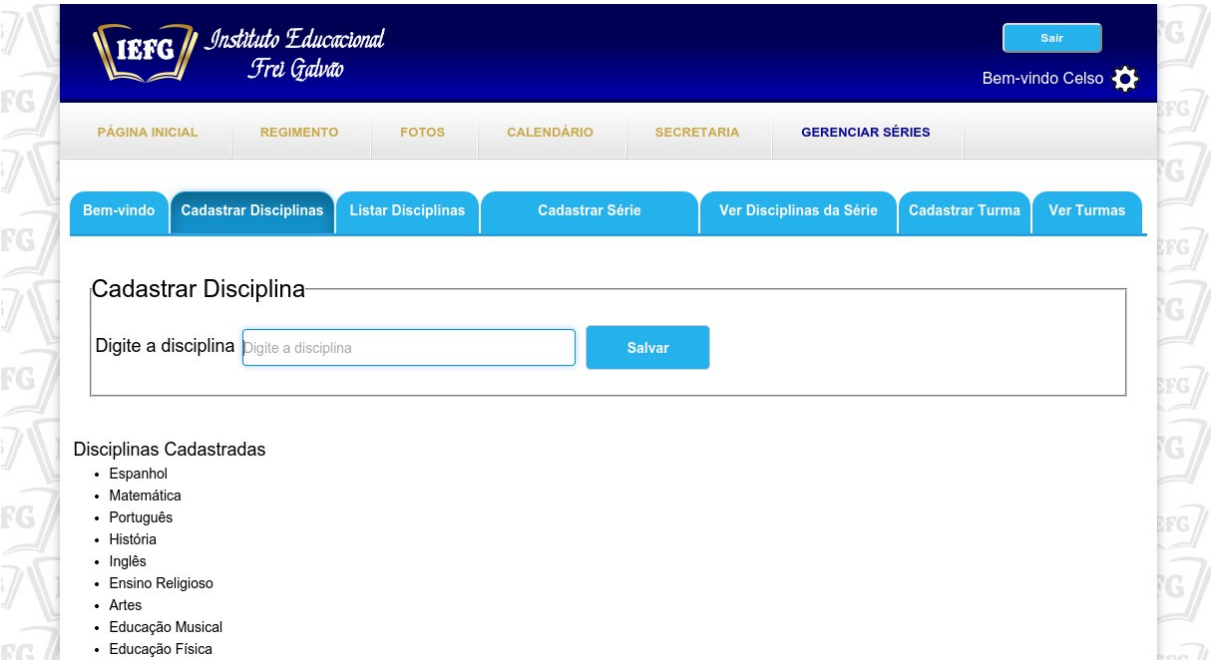


Imagem 4: Secretaria - Cadastrar disciplina

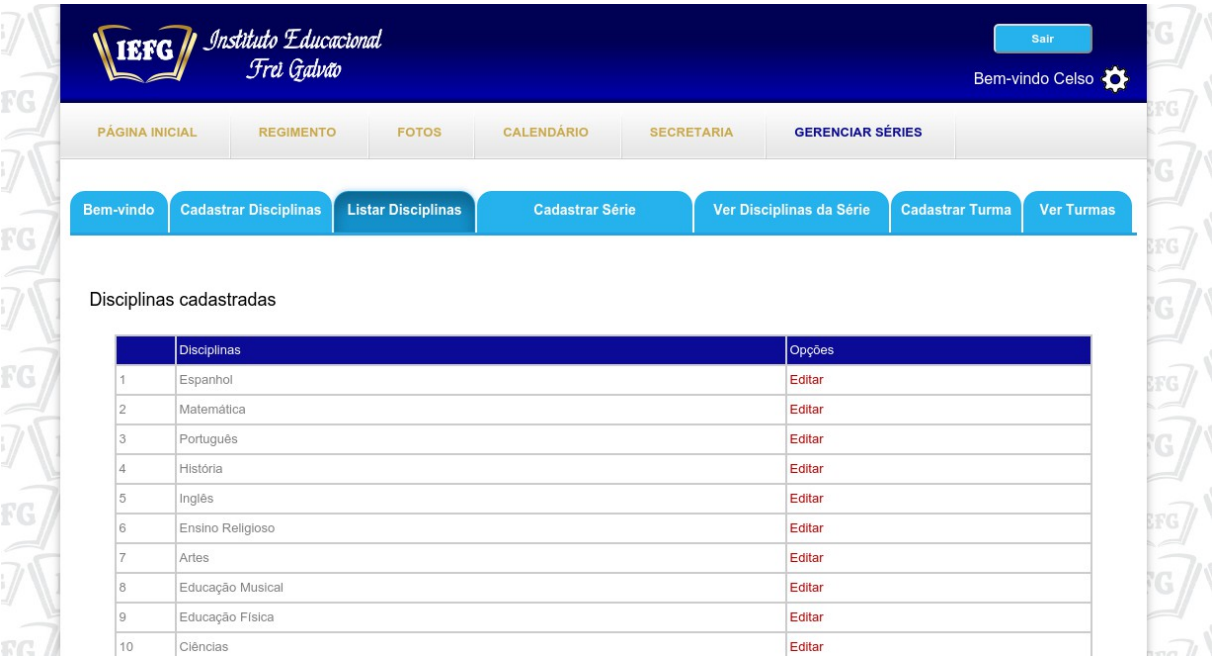


Imagem 5: Secretaria - Listar Disciplina

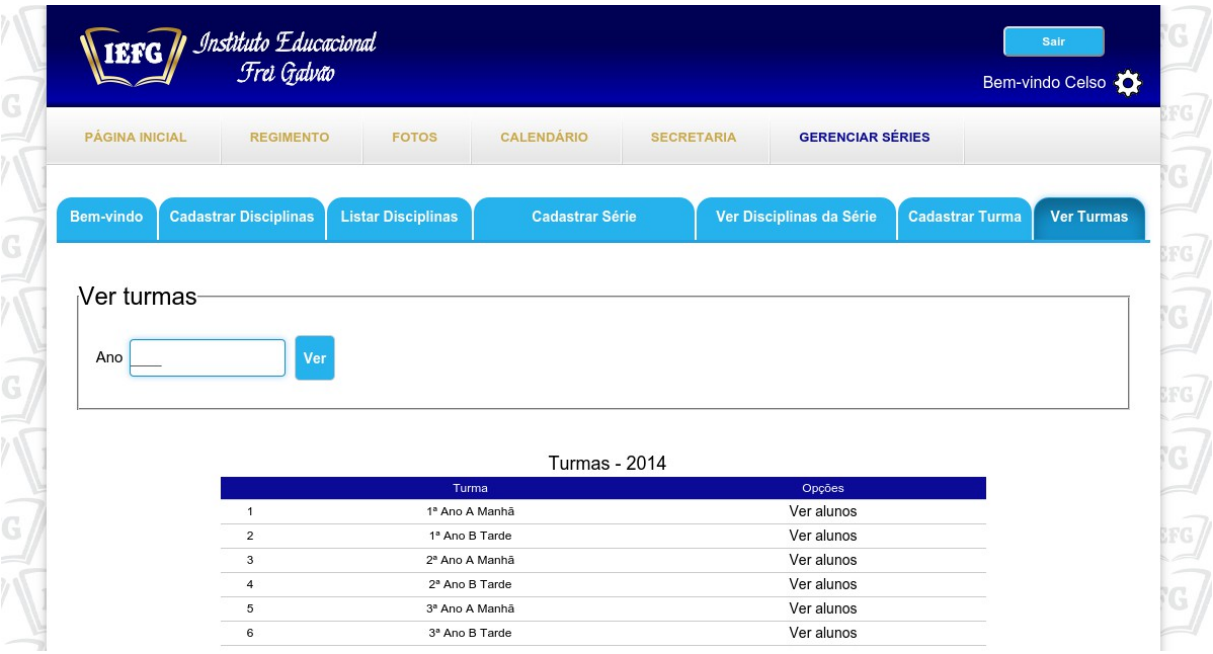


Imagem 6: Secretaria - Ver Turmas

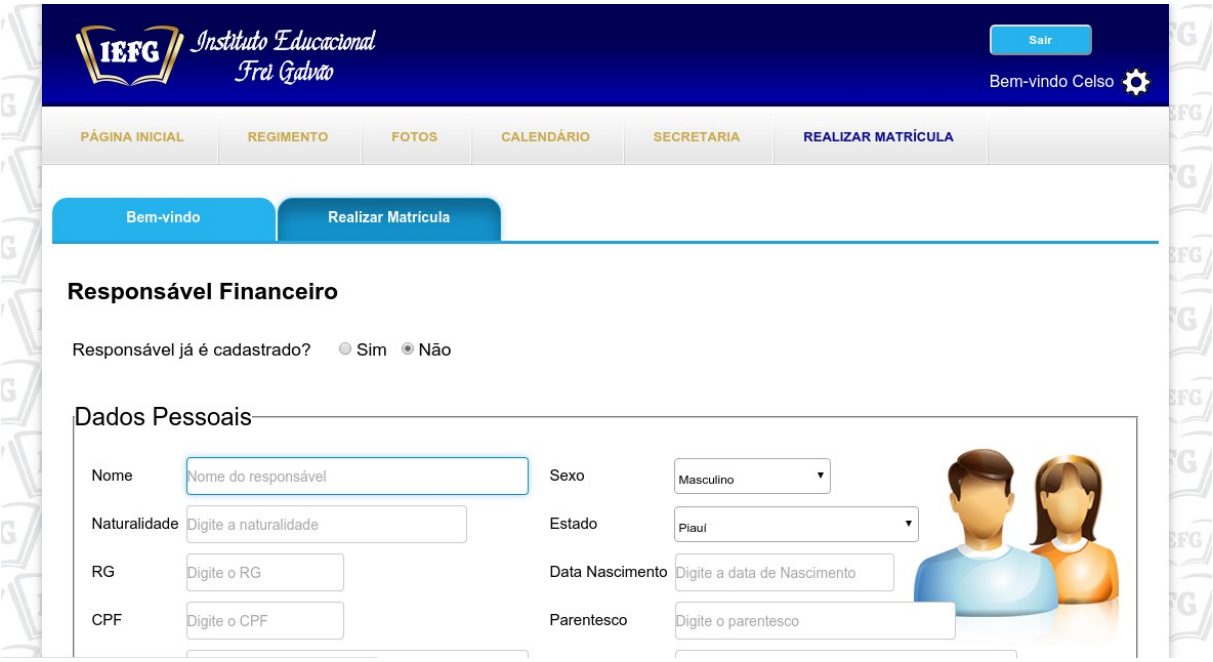


Imagem 7: Secretaria - Realizar Matrícula

Imagem 8: Secretaria - Gerenciar Alunos

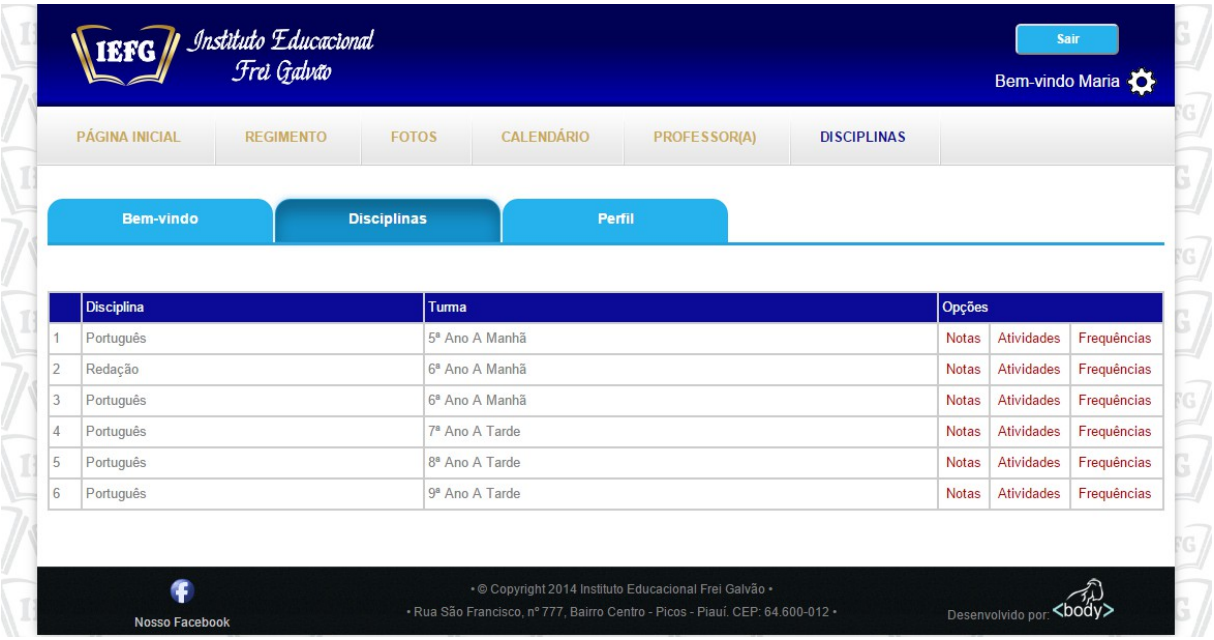


Imagem 9: Professor - Lista de Disciplinas

Imagem 10: Professor - Lista de Notas

Imagem 11: Professor - Grafico de Notas

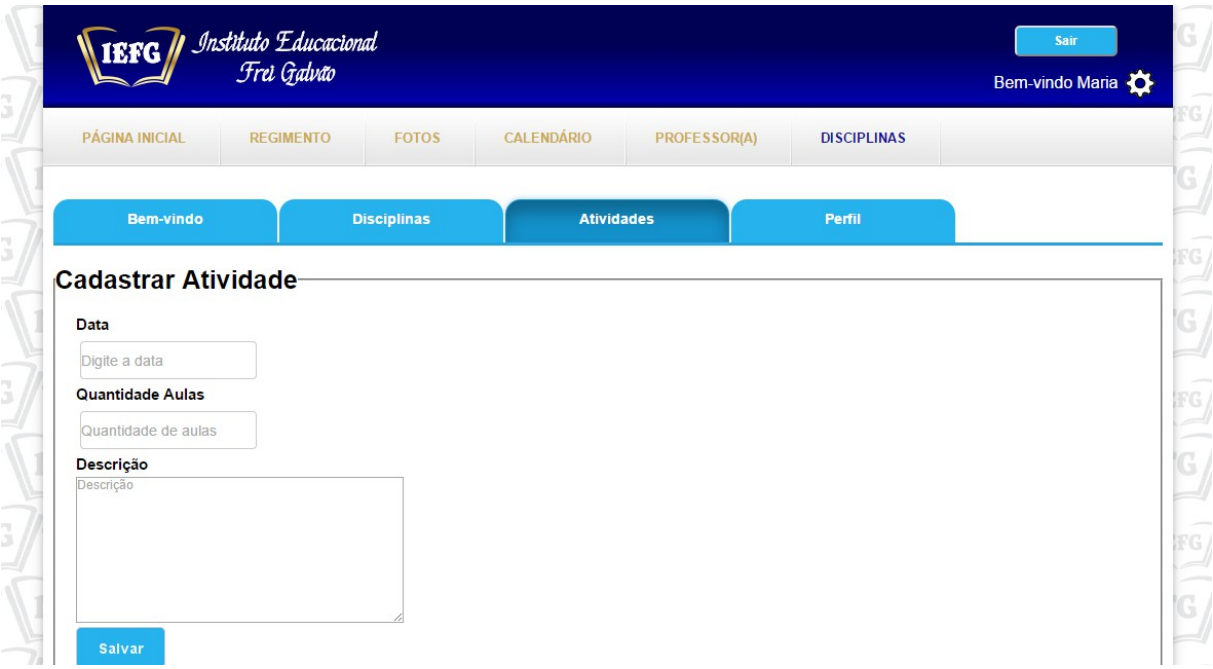


Imagem 12: Professor - Cadastrar Atividades

Imagem 13: Professor - Lançar Frequência - Selecionar Dias

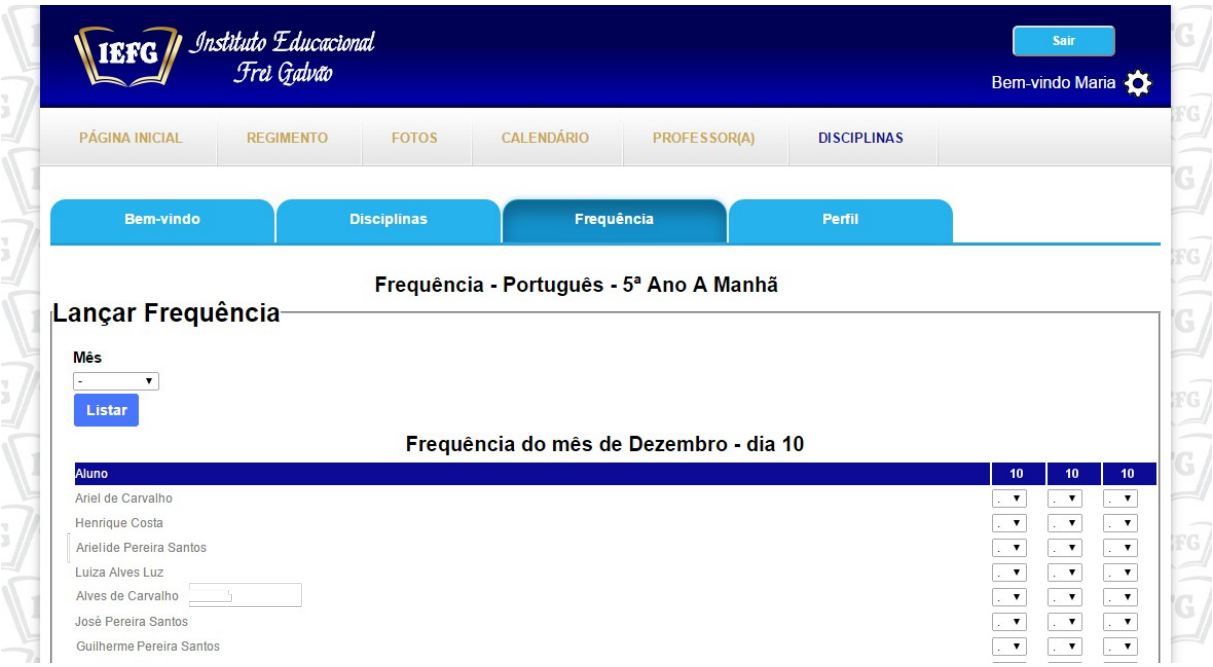


Imagem 14: Professor - Lançar Frequências

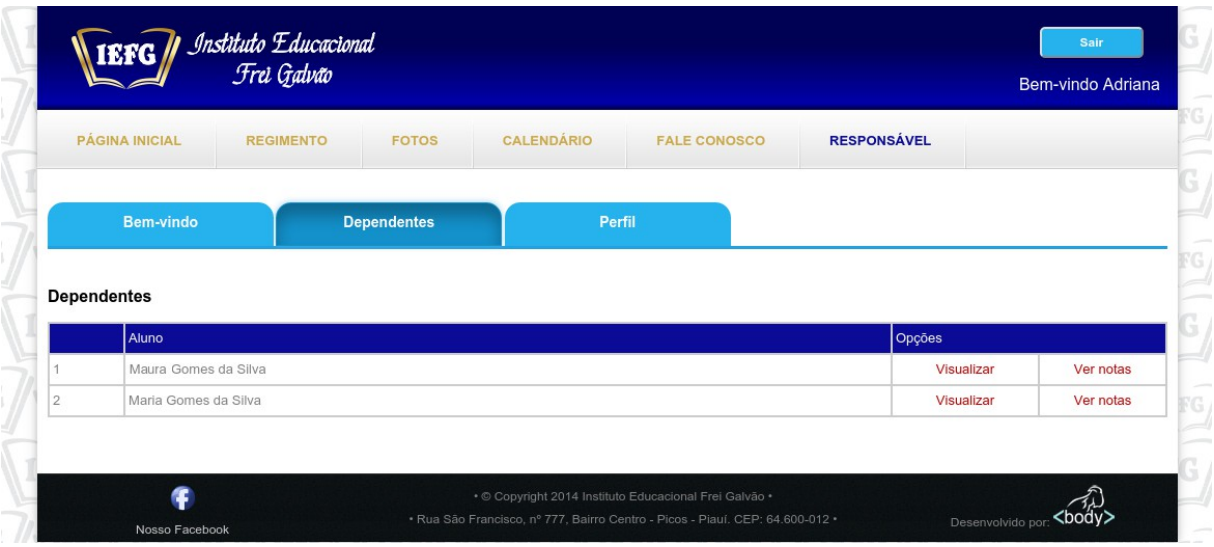


Imagem 15: Responsável - Visualizar Dependentes

Imagem 16: Responsável - Notas de Dependente

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, **ANDERSON ARRAES DE MORAES MONTE**, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE INFORMAÇÕES ESCOLARES de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI, 30 de janeiro de 2015.

Anderson Arraes de Moraes Monte

Assinatura