-

PROJETO FINAL DE GRADUAÇÃO

Proposta de Arquitetura de Sistema Web/Aplicativo Mobile para Sistema de Gerência de Projetos do Latitude

Pereira João Agostinho

Caio Cezar Ninaut Heleno

Brasília, 27 de novembro de 2018

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

PROJETO FINAL DE GRADUAÇÃO

Proposta de
Arquitetura de Sistema Web/Aplicativo Mobile para Sistema de Gerência de Projetos do Latitude

Pereira João Agostinho
Caio Cezar Ninaut Heleno

Relatório submetido ao Departamento De Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação.

Banca Examinadora

Prof. Georges Daniel Amvame Nze, Dr., ENE/UnB _____
Orientador

Fábio Lúcio Lopes Mendonça , Homologador ENE/UnB _____
Membro Interno

Prof. Rodrygo Torres Cordova, Msc – PPG/IESB _____
Membro Externo

Dedicatória (s):

Dedico esse projeto à Deus, Pois sem ele não teria forças para essa, longa caminhada, aos meus familiares, amigos.

Caio Cezar Ninaut Heleno

Dedico esse projeto a memória da minha irmã Tamboa João Francisco, tua luta sempre será motivação para mim.

Pereira João Agostinho

Agradecimentos

Primeiramente devo agradecer a Deus por ter me dado saúde e força para superar as dificuldades ao longo de todo esse tempo longe de casa, por me proteger, iluminar o meu caminho e nunca me deixar se perder durante esta caminhada. Agradeço aos meus pais, por me terem gerado, criado se dedicado à minha educação e formação como ser humano. Aos meus irmãos e a minha Tia Domingas Albano Cruz, por cada palavra de apoio, por cada dia de oração, por cada mensagem de motivação, por sempre estarem lado a lado, ombro a ombro me mostrando a direção. De maneira especial agradeço ao Meu Irmão Lello Francisco, desde a partida do pai que tem sido como um pai para mim, certamente nada era possível sem ti. Aos meus amigos e colegas, que sempre se importaram, e se mostraram disponíveis para me escutar em todos momentos que me senti só e precisei conversar. A minha namorada Marilda Augusto pela paciência, todo amor e carinho e cada palavra fortalecedora. Finalmente agradeço ao Prof. Dr. Georges Daniel Amvame Nze, por toda disponibilidade, paciência que demonstrou durante esse projeto.

Pereira João Agostinho

Agradeço primeiramente à Deus por estar ao meu lado e me dar saúde e forças para superar todos os desafios que enfrentei durante essa caminhada. A todos que estiveram do meu lado desde o início, amigos e familiares que estão em meu coração, em especial agradeço ao professor e orientador Georges Daniel pela paciência e dedicação continuamente, a minha avó Therezinha Scheid Ninaut, que é minha motivação para este trabalho e um exemplo de ser humano, que durante essa jornada me apoiou e acreditou em mim mesmo quando eu não acreditava. A todos os meus amigos e familiares que, de qualquer forma, estiveram do meu lado durante essa conquista e por fim, agradeço ao meu amigo e companheiro de curso Pereira João Agostinho por ter batalhado comigo e trabalharmos juntos no desenvolvimento desse projeto que finaliza uma das fases mais importantes de nossas vidas. Fico extremamente agradecido. Nada disso ocorreria se não fossem vocês.

Caio Cezar Ninaut Heleno

RESUMO

Gerenciar projetos tem se tornado cada vez mais difícil, especialmente quando não se tem um ambiente estruturado, isso é um ambiente em que o gerente disponha das ferramentas mais adequadas para fazer um trabalho com eficiência. Por essa razão são inúmeras, as ferramentas tecnológicas que cada vez mais tem sido desenvolvida, visando garantir que projetos atinjam o produto final, de forma eficiente.

Assim sendo, o presente trabalho tem como objetivo a elaboração da arquitetura de um sistema web/aplicativo mobile, para as tarefas de gerenciamento de projetos e controle de estudantes bolsistas, sobre tutela do departamento Latitude da Universidade de Brasília. O sistema foi pensando e concebido para auxiliar o gerente de projetos do Latitude, nas atividades básicas diárias, relacionadas aos projetos orientados pelo departamento como, cadastro de estudantes, controle de projetos cadastrados, vinculação de atividades aos estudantes, e controle de prazos, respeitando o atendimento às boas práticas propostas no Guia de Conhecimento em Gerência de Projetos, PMBOK.

Palavras chaves: Gerenciamento de projetos, sistema web, Aplicação *mobile*.

ABSTRACT

Managing projects has become increasingly difficult, especially when you do not have a structured environment; this is an environment where the manager has the most appropriate tools to do a job efficiently. For this reason, there are countless technological tools that have been increasingly developed, in order to ensure that projects reach the final product efficiently.

Therefore, the present work has the objective of elaborating the architecture of a web / mobile application system, for project management tasks and control of scholarship students, on tutelage of the Latitude Department of the University of Brasília. Designed to assist the Latitude project manager in the daily basic activities related to the projects oriented by the department such as, registration of students, control of registered projects, linking of activities to students, and control of deadlines, respecting the good service practices proposed in the Project Management Knowledge Guide, PMBOK.

Keywords: Project Management, Web System, Mobile application.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ACRÔNIMOS

API	Interface de Programação de Aplicativos
DOM	*Document Object Model*
DBI	*Data Base Interface*
SQL	*Structured Query Language*
NoSQL	*Not Only SQL*
HTML	*Hyper Text M Language*
CSS	*Cascading Style Sheets* (Folha de Estilo em Cascatas)
GUI	Interface gráfica do utilizador
SDK	*Software Development Kit*
IDE	Ambiente de Desenvolvimento Integrado
JSON	*Java Script Object Notation*
SGBD	Sistema de Gerenciamento de Banco de Dados
JS	*Javascript*
BaaS	*Backend as a Service*
SGBD	Sistema de gerenciamento de banco de dados
VScode	*Visual Code*
APP	aplicativo
iOS	Sistema Operacional iPhone
HTTP	*Hypertext TransferProtocol*
ADM	Administrador
BaaS	*Backend as a Service*
PMI	*Project Management Institute*
PMBOK	*Project Management Body of Knowledge*
TI	Tecnologia da Informação
UML	*Linguagem de Modelagem Unificada*
OMG	*Object management Group*

1 INTRODUÇÃO

1.1 Motivação

Como sabido o Laboratório de Tecnologias da Tomada de Decisão (LATITUDE.UnB) articula competências nas áreas de engenharia de redes, sistemas distribuídos, engenharia de software, busca e indexação, arquitetura da informação, exploração multidimensional da informação e arquitetura de sistemas de informação, para aplicação ao suporte e ajuda à tomada de decisão, à inteligência nos negócios e comportamentos organizacionais (*business intelligence)* e à gestão com base em tecnologias da informação. Atualmente, o processo de cadastro e preenchimento da documentação de novos bolsistas e pesquisadores dos projetos do LATITUDE é feito manualmente pela equipe de coordenação de projetos. O problema inicia-se quando o usuário deve preencher diversos arquivos que não ficam em uma base de dados, e sim armazenados em um repositório, dificultando assim o controle destes bolsistas e pesquisadores.

Motivados pelos fatos supracitados, isso é, devido a diversidade de projetos que o LATITUDE se propõe dirigir, a necessidade de se ter uma ferramenta para organizar, cadastrar, monitorar e avaliar, em fim automatizar e sistematizar o controle dos projetos, para bolsistas e pesquisadores orientados pela equipe de coordenação de projetos LATITUDE se torna urgente, sobre pena de se fazer uma tarefa mais bem-feita, com eficiência e passível de menos erros. Foi motivado por essas razões que nos dispusemos a projetar uma arquitetura, que pudesse implementar um sistema web e uma aplicação mobile que funcionasse, de modo que ajudasse o gerente, no controle dos Bolsistas e Pesquisadores, controlados pela equipe de coordenação de projetos LATITUDE.

1.2 Objetivos

1.2.1 Objetivo geral

Desenvolver um Sistema Web/Aplicativo móvel que auxilia o(s) gestor(es) de projetos nas tarefas de planejar, monitorar e controlar os usuários, bem como os projetos vinculados a eles a qualquer hora e em qualquer lugar.

1.2.2 Objetivos específicos

- Permitir que o Administrador crie, edite, e remova usuários do sistema;
- Permitir que o Administrador, criar, editar, e remova bolsistas e pesquisadores;
- Permitir que o Administrador consulte os bolsistas e pesquisadores;

- Permitir que o Administrador cadastre projetos;
- Permitir que o Administrador cadastre atividades e vincule as mesmas aos bolsistas e pesquisadores.

1.2.3 Estrutura do trabalho

Para melhor compressão do texto o trabalho está estruturado da seguinte forma:

Capitulo 2: Fundamentação Teórica

Apresenta-se os conceitos centrais do projeto, e conceitua-se detalhadamente cada ferramenta usada.

Capitulo 3: Metodologia

Fala-se sobre qual a didática foi abordada na concepção do trabalho e como o aplicativo em si funciona. Apresenta-se as etapas para a realização do projeto, e qual foi a motivação para escolha dos softwares utilizados no desenvolvimento do aplicativo.

Capítulo 4: Resultados

Aqui apresenta-se os resultados finais atingidos, evidenciando nossos objetivos iniciais.

Capítulo 5: Conclusão e Trabalhos Futuros

Mostra-se até que ponto o aplicativo pode ser aprimorado.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Gerenciamento de projetos

O gerenciamento de projetos é um campo crescente cada vez mais utilizado por organizações de todos os tamanhos, principalmente pelo fato de os empresários e executivos lidarem mais com as responsabilidades diárias de gestão de organização, automaticamente torna-se importante o uso de especialistas para supervisionar projetos desde a concepção até a conclusão. A gestão de projetos é muito importante, principalmente porque fornece uma estrutura para ajudar a alcançar os objetivos de determinada organização. Para entendermos bem o conceito de gerenciamento de projetos, começamos por definir o que seria um projeto em si. Segundo o PMBOK: "Um projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. Os projetos e as operações diferem, principalmente, no fato de que os projetos são temporários e exclusivos, enquanto as operações são contínuas e repetitivas. " Resumidamente, projeto é um evento que tem início e fim (escopo) bem definidos. Diferente de uma operação, execução de backup, por exemplo, que é uma tarefa diária e não se sabe quando ela não será mais necessária. Um projeto é único no sentido de que não se trata de uma operação de rotina, mas um conjunto específico de operações destinadas a atingir um objetivo em particular. Assim, uma equipe de projeto inclui pessoas que geralmente não trabalham juntas e algumas vezes vindas de diferentes organizações e de diferentes lugares.

O *Project Management Institute* (PMI) define Gerenciamento de Projetos como "a aplicação de conhecimento, de habilidades, de ferramentas e técnicas a uma ampla gama de atividades para atender aos requisitos de um determinado projeto" (PMI, 2008). O Gerenciamento de Projetos, hoje é acessível às pequenas e organizações medias, podendo ser o diferencial entre o sucesso e o fracasso. O entendimento do conceito de projeto (qualquer atividade com início, meio e fim e cujo resultado deve ser único), surgi como um do requisito principal usado como base para definirmos quais organizações precisam ou não de gerenciamento de projeto. Por exemplo se uma organização trabalha com linha de produção em série, sem nenhum tipo de customização ao cliente, não tem motivo para usar o Gerenciamento de Projetos. Já uma empresa que oferece a seus clientes produtos únicos pode iniciar procedimentos para "projetar" a gestão (Gerenciamento de Projetos, Roberto Candido).

Existem 5 grupos de processos do gerenciamento de projetos identificados pelo guia de gerenciamento de projetos PMBOK e nenhum desses processos podem ser dispensados ou tratados com menos cuidado:

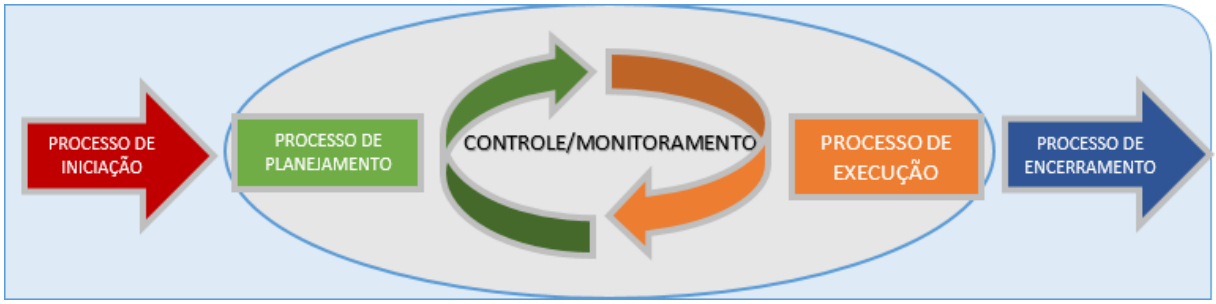


Figure 2.1 Processos de um projeto de gerenciamentos *Fonte: PMI, 2004.*

- **Processo de Inicialização:** Os processos de iniciação são aplicados para oficializar o início do projeto ou de uma nova etapa de um planejamento que já existe. Se o projeto for composto por várias etapas é necessário que em cada uma delas sejam abordados os termos essenciais dos grupos de processos de iniciação. Durante esse processo é feita definição do escopo preliminar e liberação do capital financeiro para o início da execução do planejamento. Nesse passo, também há a possibilidade de analisar se o projeto tem que ser interrompido, continuado ou postergado. Participam dessa etapa os gestores, que dão o aval para a execução ou da iniciativa e o gerente ao qual o projeto é atribuído. Fase em que é realizado o levantamento de todas as necessidades físicas, financeiras e de pessoal para a concretização do projeto. As análises são feitas pela alta gerência da organização, que deve autorizar ou não a execução do projeto, balizada por um criterioso estudo de viabilidade. As atividades típicas usuais do processo de inicialização são normalmente: Elaboração da proposta do projeto e aprovação da gerência, Seleção de projetos, Aprovação dos clientes e Autorização para realização do projeto. No desenvolvimento dessas atividades, a documentação é peça fundamental para o sucesso. Devem ficar evidentes os rumos e objetivos do projeto; também precisam ser definidos seu escopo, recursos e prazos.
- **Processo de Planejamento:** O objetivo do processo de planejamento é detalhar o que foi definido na etapa de iniciação. Essa fase fica a cargo do gerente de projeto, sendo responsável por desenvolver o plano de projeto e os seus planos complementares. Essa etapa define os caminhos para que os objetivos do projeto sejam alcançados. Nessa etapa é elaborado o Plano de Gerenciamento de Projetos (*Project Charter*), documento que deve contemplar todos os processos desse gerenciamento. A profundidade e complexidade do planejamento estão diretamente ligadas ao tamanho do projeto. É por norma o passo mais complexo, pois exige que o escopo seja esmiuçado, estudado e definido por partes. É necessário estabelecer qual o valor do investimento e o prazo do

projeto, como o trabalho será organizado, a estratégia de comunicação, os membros da equipe que irão executá-lo, sendo que devem ser escolhidos de acordo com as habilidades que possuem e as necessidades da proposta.

- **Processo de Execução:** Nessa etapa o gerente se encarrega de coordenar os recursos disponíveis para executar o que foi planejado, se baseando no plano de gerenciamento do projeto e nos planos auxiliares feitos anteriormente. Aqui, as tarefas são delegadas para os integrantes da equipe, conforme as suas funções. No decorrer dessa fase, o gerente verifica se as entregas estão em harmonia com o escopo do projeto, o defende de possíveis mudanças e reafirma o nível da qualidade prevista para o trabalho que está em processo de execução. São atividades típicas desse processo: Distribuição de informações, Garantia da qualidade, Solicitação das propostas de fornecedores, Controle dos fornecedores, Controle ou mobilização da equipe, Desenvolvimento da equipe de projeto.
- **Processo de Monitoramento e controle:** No processo de monitoramento e controle, o gerente de projeto supervisiona todas as variações ocorridas. Para identificá-las, se faz uma análise comparativa entre o que foi concretizado com as linhas de base de prazo, escopo e custo determinados no planejamento. É considerado etapa vital para o sucesso do projeto, pois permite a percepção de problemas em tempo hábil para solucioná-los. Esse procedimento deve possibilitar medições regulares do projeto para avaliação de desempenho. São atividades típicas desse processo: Controle do desempenho do projeto, Realização do controle integrado de mudanças, Monitoramento e controle de riscos, Obtenção da aceitação do escopo, Administração de contratos, Controle da qualidade, Gerenciamento de partes interessadas, Gerenciamento da equipe do projeto. O controle garante a qualidade do projeto e sua conformidade com o planejamento durante a execução. Quanto mais tarde forem detectados os problemas, mais dispendiosas serão as correções.
- **Processo de Encerramento:** O processo de encerramento consiste em formalizar o fechamento do projeto ou de uma das suas fases, balancear o registro de erros e acertos, a fim de se preparar melhor para os próximos projetos. Para oficializar o encerramento é preciso criar diversos documentos, tendo a aceitação do cliente, do patrocinador ou da alta administração. A etapa final também exige uma revisão pós-projeto para assegurar que todos os planos projetados foram cumpridos. Isso é normalmente feito a partir de uma reunião do gerente de projeto com todos os membros da equipe e responsáveis por cada tarefa. O ponto crítico desse processo é a aceitação do usuário final do produto ou

serviço desenvolvido durante o projeto, bem como se a sua finalidade foi atingida. Pode acontecer de determinada funcionalidade criada não ser aceita ou não surtir efeito exatamente conforme o planejado. Se esse problema for apontado, há que se avaliar o que deu errado, rever e consertar juntamente com a equipe de execução. Todos os processos de gerenciamento de projetos ajudam a colocar os novos planos em prática com mais segurança e facilidade, além de aumentar o grau de eficácia da empresa. Portanto o encerramento do projeto pressupõe que todos os contratos firmados durante a execução sejam encerrados formalmente, gerando imediatamente condições para a avaliação de desempenho, realizada de acordo com métricas preestabelecidas.

2.1.1 Áreas de conhecimento de gerenciamentos de projetos

A administração de um projeto abrange dois aspectos principais. O primeiro tem a ver com a administração de projetos em si, como sistema de recursos e ações que buscam entregar um produto dentro de um prazo. Como sabemos, o projeto é uma iniciativa temporária que exige o detalhamento das necessidades a serem atendidas, para que um escopo coerente possa ser definido e, em seguida, o prazo e o custo possam ser planejados. O segundo aspecto é o de gerenciar o projeto dentro de um contexto empresarial. Os projetos exigem sinergia, equipes coesas, divisão do trabalho, apoio da direção. Com isso o gerenciamento de Projetos envolve inúmeras atividades a serem desenvolvidas, monitoradas e concluídas. Para facilitar sua aplicação e garantir bons resultados, o processo de gerenciamento é dividido em áreas ou etapas definidas pelo Guia PMBOK (PMI, 2008).

2.1.2 Gerenciamento da integração:

A integração, no contexto do gerenciamento de um projeto, consiste em fazer escolhas sobre em que pontos concentrar recursos e esforço, antecipando e tratando possíveis problemas, antes de se tornarem críticos, coordenando o trabalho visando o bem geral do projeto. O esforço no gerenciamento de integração também envolve fazer compensações entre objetivos e alternativas conflitantes, além de abordar todos os aspectos envolvidos na etapa da integração e, consequentemente, as formas de executá-la com consistência. Cabe ao gerenciamento da integração, a função de coordenar as diferentes gestões de um projeto para alcançar os

resultados, pois as equipes são multidisciplinares e seus membros, originários de diversas áreas funcionais. Essa atividade objetiva agrupar sistematicamente conhecimento e competências de forma harmônica. Por essa razão, em geral quando projetos são de grande porte, podem ser desmembrados em pequenos subprojetos, que podem ser divididos em pequenos pacotes de atividades, similares e complementares. Tudo isso acontece para facilitar o processo de gerenciamento do projeto e maior controle por parte do gerente. Podemos definir um subprojeto, como sendo, a parte menor de um projeto criado quando há a necessidade da subdivisão do esforço planejado em componentes mais facilmente gerenciáveis. O controle centralizado, na forma de um único cronograma, com centenas de atividades pode facilmente sair do controle. A divisão das atividades de um projeto em subprojetos permite a delegação de responsabilidades e possibilidade de vários gerentes atuarem de forma integrada. Dessa forma, o sucesso de determinado projeto está relacionado ao alcance dos objetivos dos subprojetos a ele vinculados. Na figura 2.1 ilustramos um pequeno diagrama que mostra o processo de integração das partes de um projeto.

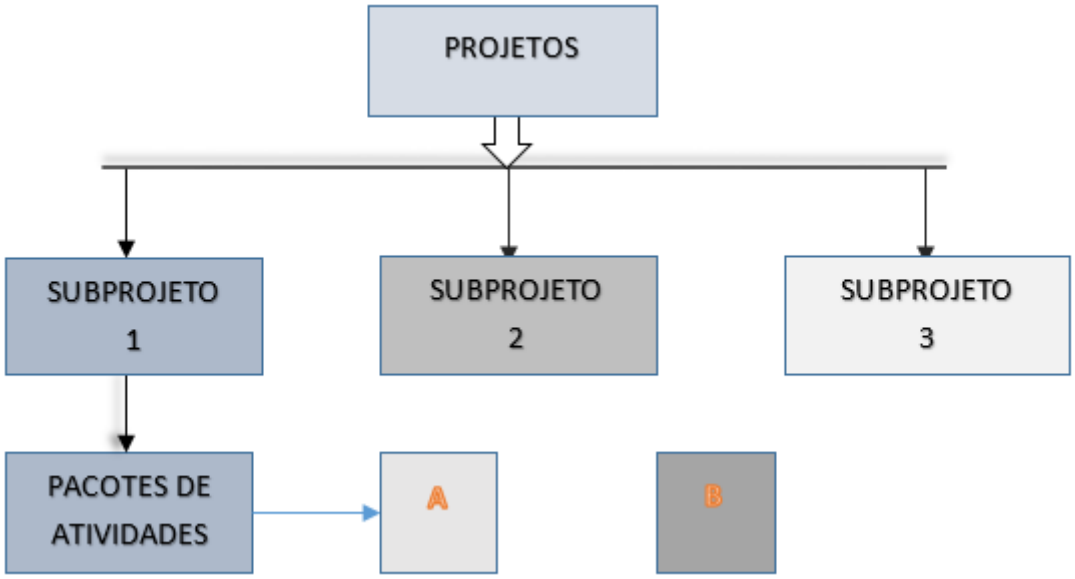


Figura 2.2 integração das partes de um projeto Fonte: Gerenciamento de projeto, Roberto Candido.

Em termos de gerenciamento de projetos dividir um projeto em subprojetos distribui responsabilidades àqueles que efetivamente realizam o trabalho ou podem supervisioná-lo de forma mais direta, além de combinar a autoridade do gerente com prestação de contas em tempo

real. A criação de subprojetos dentro de um projeto principal ajuda os gerentes de projetos a obter acesso às partes do cronograma e a ter controle sobre essas partes.

Como já foi dito acima segundo o Guia PMBOK, o gerenciamento da integração do projeto inclui os processos e as atividades necessárias para identificar, definir, combinar, unificar e coordenar os vários processos e atividades dos grupos de processos de gerenciamento. A comunicação e o alinhamento da linguagem entre os participantes são fundamentais para o sucesso do gerenciamento da integração. Este, em síntese, refere-se à gestão do conjunto de atividades e subprodutos, que, de forma sistêmica, geram o resultado esperado para o projeto. O gestor de integração precisa ter conhecimento técnico e administrativo, além de habilidade para gerenciar equipes, agindo como integrador dos processos e das pessoas. Um projeto de sucesso deve ser realizado conforme o planejado. Em todas as fases do projeto, há intervenientes que podem contribuir para a ocorrência de problemas relacionados, entre outros, ao escopo (mudanças de requisitos e especificações), tempo (atrasos) e custo do projeto (superfaturamento). Há uma série de conhecimentos e práticas que favorecem o efetivo gerenciamento de projetos e, assim, contribuem para minimizar suas falhas mais comuns. O PMBOK estruturou essas melhores práticas em 10 áreas de conhecimento, com base nos processos que os compõem, como ilustramos na figura 2.3.

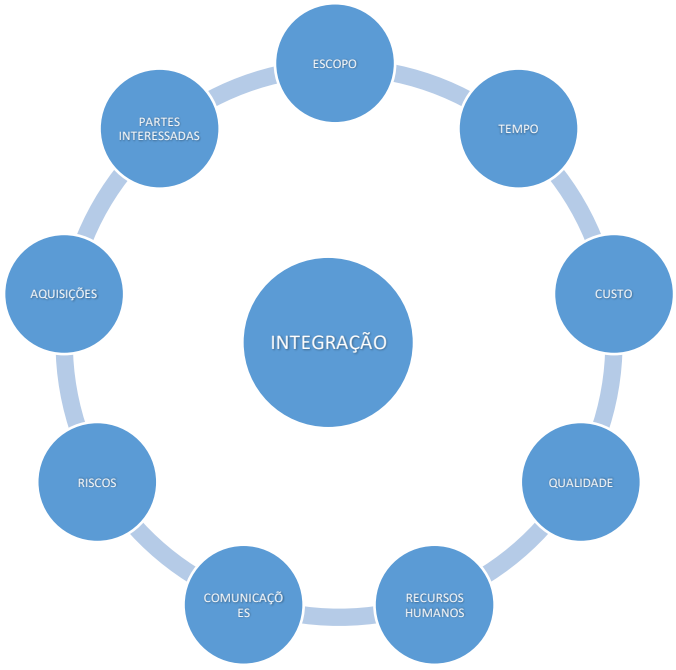


Figura 2.3 Gerenciamento da integração de projetos Fonte: Guia da PMBoK,5ª edição

Para cada uma dessas áreas foram definidos, segundo o PMI, os vários processos relacionados ao gerenciamento de projetos, que são os cinco grupos, denominados grupos de processos, que definimos mais acima, nomeadamente: Iniciação, Planejamento, Execução, Monitoramento e controle e Encerramento, sendo a Integração a área de conhecimento, que condensa informações de todas as demais áreas de conhecimento. As principais atribuições do gerente de integração são:

- **Desenvolvimento do plano** – O plano do projeto deve agregar os resultados dos outros processos de planejamento, constituindo um documento coerente e consistente.
- **Execução do plano** – Nessa etapa, todas as atividades previstas no plano do projeto precisam ser efetivadas. É fundamental o acompanhamento da execução para garantir o sucesso do projeto.
- **Controle geral de mudanças** – Todas as mudanças ocorridas no projeto precisam ser registradas, porque podem resultar em aumento de prazo e geralmente de custos.

Gerenciar projetos é uma atividade complexa e que necessariamente depende de recursos, requisitos específicos definidos pelo cliente e sincronia de ações em seu desenvolvimento, controle e negociação. Por essa razão o gerenciamento da integração é de extrema importância na execução do projeto atuando no início do projeto, no processo de elaboração do escopo. Durante a execução auxilia no monitoramento e controle do projeto. Por fim, na fase de encerramento contribui para a organização dos materiais do projeto e registro das lições aprendidas. De acordo com o fluxo de processos do PMBOK, definir um bom plano de gerenciamento certamente contribuirá para a efetividade das etapas previstas no projeto e, consequentemente, suas entregas. Em seguida passamos a aprofundar cada uma das partes que compõem o gerenciamento da integração.

2.1.3 Gerenciamento de escopo

O gerenciamento de escopo é um dos processos mais importantes do gerenciamento de projetos. Pois sem o escopo não existe projeto, pois é no escopo que está definido todo o trabalho que o gerente pretende realizar garantindo que apenas o que foi definido seja realizado. Ou seja, o líder de projeto deve assegurar que não haja trabalhos além do que foi planejado. No contexto do projeto, o termo escopo pode se referir a Escopo do produto. As características e funções que descrevem um produto, serviço ou resultado, ou seja, é todo trabalho que precisa

ser realizado para entregar um produto, serviço ou resultado com as características e funções especificadas. Um escopo bem definido direciona as futuras ações, pois possibilita que as partes envolvidas no projeto tenham expectativas realistas e que os recursos possam ser adequadamente dimensionados. A figura 2.3 identifica os principais processos envolvidos no gerenciamento de escopo:

Figura 2.4 processos de gerenciamento do escopo fonte: Gerenciamento de projeto, Roberto Candido

- **Planejamento:** Planeja o gerenciamento do escopo define e documenta como a equipe do projeto irá definir, validar e controlar o escopo, entradas, ferramentas e saídas do processo.
- **Detalhamento:** Define os subprojetos que compõem o projeto principal. É o momento de especificar os subprodutos que vão constituir o produto final e de dividir os subprojetos em pacotes de atividades para facilitar o acompanhamento na fase de desenvolvimento.
- **Controle de mudanças:** Consiste em acompanhar e registrar as mudanças sucedidas no escopo do projeto. O monitoramento e controle do escopo são pontos importantes no processo de acompanhamento da geração do escopo. O controle do escopo é o processo de monitoramento do andamento do escopo do projeto e do produto e gerenciamento das mudanças feitas na linha de base do escopo. O controle do escopo do projeto assegura que todas as mudanças solicitadas e ações corretivas ou preventivas recomendadas sejam corretamente aplicadas. Os processos de monitoramento e controle devem incluir o monitoramento das atividades em andamento do projeto em relação ao plano de gerenciamento do projeto e à linha de base do desempenho do projeto e o controle dos fatores que poderiam dificultar o controle integrado de mudanças de forma que somente mudanças aprovadas sejam implementadas. O processo de gerenciamento de escopo abrange os seguintes sub processos.
- **Coletar os requisitos:** Cada projeto exige um balanceamento cuidadoso de ferramentas, fontes de dados, metodologias, processos e procedimentos e de outros fatores, para garantir que o esforço gasto nas atividades de determinação do escopo esteja de acordo

com o tamanho, complexidade e importância do projeto. Por outras palavras requisitos são o que a parte interessada no projeto precisa em um produto ou projeto. Os requisitos podem referir sobre uma funcionalidade no produto, podem estar relacionados a qualidade, a processos de negócios, à conformidade, ou até ao gerenciamento de projetos. Esse é um processo crucial para o projeto, pois falhas nele podem resultar em muitas mudanças e até mesmo no fracasso do projeto. Algumas ferramentas como entrevistas, revisões de registros históricos, brainstorming, podem ser usadas para a coleta de requisitos.

- **Definir o escopo**: Esse processo refere-se no que está ou não incluído no projeto e nas entregas. Resulta na declaração do escopo do projeto e pode incluir: Escopo do produto (o que será feito?); Escopo do projeto (como será feito?); Entregas (para o produto e projeto); Critérios de aceitação; O que não faz parte do projeto? Premissas e restrições.
- **Criar a EAP**: A estrutura analítica do projeto é uma ferramenta organizacional que mostra visualmente todo o escopo do projeto, dividido em entregas. Basicamente a EAP Subdivide os produtos do projeto e o trabalho do projeto em componentes menores e mais gerenciáveis, organizando e definindo o escopo total do projeto, de uma forma em que cada nível descendente da EAP represente uma definição mais detalhada do trabalho do projeto.
- **Verificar escopo:** é o processo de formalizar a aceitação das entregas do projeto. Esse processo ocorre sempre que uma entrega for concluída de modo a verificar e documentar o nível e grau de conclusão da entrega em relação aos seus requisitos. A verificação do escopo do projeto inclui a revisão das entregas para garantir que cada uma delas foi terminada de forma satisfatória.

- **Controlar escopo:** basicamente é o processo de monitorar o status do escopo do projeto e do produto e gerenciar as alterações na linha de base de escopo. Esse processo envolve medir e avaliar a execução do escopo para identificar variações e verificar se são necessárias mudanças, é um processo extremamente proativo, ou seja, o líder deve sempre estar atento para que o trabalho esteja sendo executado conforme a definição do escopo.

2.1.4 Gerenciamento de tempo

A questão tempo, é sempre de extrema importância, dentro de qualquer organização que visa cumprir seus objetivos no prazo. É primordial que cada projeto tenha seu tempo determinado e certo dentro do cronograma traçado pela equipa de gerenciamento de projetos, a fim principalmente de se evitar grandes atrasos que possam comprometer o cumprimento do planejamento inicial. Com isso podemos dizer que o Gerenciamento de tempos vem para garantir que cada etapa do projeto seja cumprida, assegurar pontualidade, cumprimento do tempo de execução e da programação, bem como acompanhar as atividades. De acordo com a Guia PMBOK (PMI, 2008), o gerenciamento de Tempo é dividido por etapas que passaremos a definir em seguida: 1-Definir atividades, 2- sequenciar atividades, 3-estimar recursos da atividade, 4-estimar duração da atividade, 5-desenvolver cronograma, 6-controlar cronograma.

- **Definição das atividades:** Segundo a guia PMBOK (PMI, 2008), é nessa etapa que O processo identificará as entregas no nível mais baixo da estrutura analítica do projeto (EAP), denominado pacote de trabalho. Os pacotes de trabalho do projeto são basicamente planejados (decompostos) em componentes menores, chamados de atividades, para fornecer uma base para a estimativa, elaboração de cronogramas, execução, e monitoramento e controle do trabalho do projeto. Por outras palavras é nessa etapa do gerenciamento de tempo em que são separadas as diversas atividades a serem desenvolvidas em subprojetos e onde são identificadas as atividades prioritárias, (as que impactam na realização de outras tarefas) agindo como ações facilitadoras do Gerenciamento de Projetos, possibilitando assim determinar o sequenciamento das ações. A etapa de definição de atividades, é composta por alguns campos como, Atributos da atividade, que incluem identificador, códigos, descrição da atividade, atividades predecessoras, sucessoras, relacionamentos lógicos, antecipações e atrasos, recursos necessários, datas impostas, restrições e premissas. Podem também incluir a pessoa responsável, **e o** tipo de atividade do cronograma, como nível de esforço, esforço distinto e esforço distribuído. Lista de marcos, que indica se o marco é obrigatório ou opcional. E por fim um campo com Mudanças solicitadas, que abre a possibilidade para listar e efetuar todas as mudanças efetuadas ao longa do projeto.
- **Sequenciar atividades:** Esse processo é responsável por identificar e documentar os relacionamentos entre as atividades do projeto. Basicamente estabelece o sequenciamento das atividades, as interdependências, com fim de evitar atrasos em

atividades interdependentes evitando assim um impacto negativo no tempo total de execução do projeto. O objetivo nesse processo é de obter um nível de eficiência superior sem deixar de olhar as restrições do projeto. Como saída são gerados os diagramas de rede do cronograma do projeto e atualizações nos documentos do projeto, além de atributos da atividade e eventuais mudanças solicitadas. Normalmente a construção de um diagrama de rede do cronograma do projeto usa nós para representar atividades e os conecta por setas que mostram as dependências.

- **Estimativa da duração das atividades**: Nessa etapa é realizada a estimativa do número de períodos de trabalho que serão necessários para terminar atividades específicas e com os recursos estimados. Este processo fornece uma quantidade de tempo necessário para concluir cada atividade. Com fim de facilitar o acompanhamento as ações podem ser divididas em formato de pacotes. Projetar a durabilidade para cada atividade, é importante porque principalmente permite identificar quais atividades estão com atrasos e demandam mais tempo e quais estão em conforme com o esperado. Como saída nessa etapa são gerados as Estimativas das durações das Atividades e as Atualizações no Documentos do Projeto (Guia do PMBOOK, 5ªedição).
- **Estimar recursos das atividades:** Normalmente é nessa etapa onde se estima os tipos e quantidades de material, recursos humanos, equipamentos, para efetuar cada atividade do projeto. Como saída é esperado, que se obtenha uma lista com todos os recursos estimados para realização das atividades, atributos da atividade atualizado, A estrutura analítica dos recursos (EAR) e uma lista com mudanças solicitadas (Guia do PMBOOK, 5ªedição). Neste processo são identificados também os tipos quantidade e características dos recursos exigidos para concluir a atividade, permitindo uma estimativa de custos e duração mais exatas.
- **Desenvolver cronograma:** realiza e a organiza bem como analisa a sequencias das atividades, suas durações, recursos e restrições visando criar o modelo do cronograma do projeto. Como saída são gerados a Linha de base do cronograma, o Cronograma do projeto, Calendário de projeto, mudanças solicitadas, plano de gerenciamento do projeto e plano de gerenciamento do cronograma (Guia do PMBOOK, 5ªedição).
- **Controlar o cronograma:** Essa etapa é responsável por monitorar o andamento das atividades do projeto, atualização no seu progresso acompanhando assim os fatores que criam mudanças no cronograma. O controle do cronograma é uma parte do processo Controle Integrado de Mudanças. Este processo fornece meios para reconhecimento dos

desvios com relação ao caminho planejado e tomada de medidas corretivas e preventivas para com isso minimizar os riscos. Um controle eficaz do cronograma de atividades pode ser feito por meio de um preenchimento de formulário, com campus que facilite o gestor observar de perto toda e qualquer alteração dos itens planejados, promovendo assim de forma mais rápida os melhores ajustes para garantir os resultados esperados no planejamento. Como saída é esperado que seja gerado Informações sobre o Desempenho do Trabalho, as Previsões de Cronograma, as Solicitações de Mudança, as Atualizações no Plano de Gerenciamento do Projeto, as Atualizações nos Documentos do Projeto e as Atualizações nos Ativos de Processos Organizacionais (Guia do PMBOOK, 5ªedição).

2.1.5 Gerenciamento de custo

Basicamente o gerenciamento de custos consiste em estimar, alocar e controlar os gastos de um projeto. É através do gerenciamento de processos que o gerente assegura execução do projeto dentro do orçamento, mantendo-se limitado aos recursos previstos no planejamento. Os custos do projeto são calculados em norma durante a fase de planejamento e devem ser aprovados antes do começo do projeto. É nessa etapa que deve se estimar e determinar todos os custos que serão envolvidos em cada atividade, assim como estimar todos recursos humanos, matérias e equipamentos. O guia da PMBOK divide o processo de gerenciamento de custos em três etapas: 1-Estimar os custos, 2-Determinar orçamento, 3-Controlar os custos.

- **Estimar os custos:** Nessa etapa o objetivo é determinar por meio de um estudo, quanto custará cada recurso necessário. Nessa etapa o gestor começa a ter contato com os valores envolvidos no projeto e deve aproveitar para pedir orçamentos e avaliar qual será a projeção de gastos do empreendimento. A estimativa de custos da atividade do cronograma envolve o desenvolvimento de uma aproximação dos custos dos recursos necessários para terminar cada atividade do cronograma.
- **Determinar o orçamento:** Depois de estimado os custos os gestores estão em condições de prever, com exatidão, o tamanho do investimento necessário para a conclusão do projeto. É nessa etapa que se define os fornecedores quando se der o caso, e se acorda valores. Segundo a Guia da *PMBOOK 5ªedição*, as estimativas de custos da atividade do cronograma ou do pacote de trabalho são preparadas antes das solicitações de orçamento detalhado e da autorização do trabalho.

- **Controlar custos:** é nessa etapa que é feita o acompanhamento, ao longo da execução do projeto, dos gastos reais. Nessa fase os acordos em relação às mudanças solicitadas estão garantidos; é nessa fase que é feita o monitoramento das mudanças reais quando e conforme ocorrem; é garantido que os possíveis estouros nos custos não ultrapassam o financiamento autorizado periodicamente e no total para o projeto (Guia da PMBOOK,5ªedição). A figura 2.5 resume basicamente as etapas do processo de gerenciamento de custo.

Figura 2.5 Gerenciamento dos custos do projeto Fonte: gerenciamento de projetos, Roberto Candido

2.1.6 Gerenciamento de qualidade

Qualquer organização presa pela busca de bons resultados, conseguidos com eficiência, visando menores custos. Tais objetivos são passiveis de serem atingidos, através de um bom processo de gerenciamento de qualidade. O Gerenciamento de qualidade é um processo de melhoramento baseado especialmente na redução contínua de custos, aumento da produtividade, e melhoria da qualidade. Essa atividade implica o mapeamento dos processos e o rastreamento dos componentes gerados nos diferentes subprojetos que, pela integração, geram o produto final, ” Gerenciamento de Projetos, Roberto Candido”. O que se quer assegurar com o gerenciamento de qualidade é que para além de bons resultados, se garanta também efetividade na realização de cada fase do projeto. Graças a gerencia de qualidade é possível verificar e assegurar além da qualidade dos serviços a serem feitos, a padronização de cada fase,

facilitando assim a especialização dos colaboradores no que precisam fazer minimizando assim as chances de riscos do projeto a ser desenvolvido e satisfação do cliente. Segundo os atores do livro Gerenciamento de projetos, a qualidade pode ser gerenciada através de 3 processos básico – planejamento, garantia e controle como podemos ilustrar na figura 2.6:

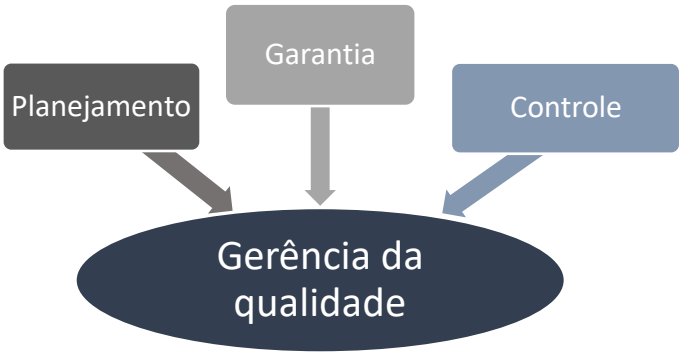


Figura 2.6 Principais processos da gerência da qualidade Fonte: Gerenciamento de projetos, Roberto Candido

- **Planejamento:** Nesse processo, são definidos os requisitos do produto principal e dos subprojetos e também os recursos necessários para a realização do projeto. Além disso, são determinados os padrões a serem atendidos, os testes a serem feitos e o tempo para a execução deles (Gerenciamento de Projetos, Roberto Candido).
- **Garantia:** Corresponde basicamente à realização de auditorias para verificar se os requisitos de qualidades definidos estão a ser cumpridos. A garantia de qualidade pode ser efetuada mediante acompanhamento regular, supervisão essa que permitirá reduzir desperdícios e eliminar atividades desnecessárias reduzindo com isso custos, garantindo assim eficácia na realização do projeto.
- **Controle:** Normalmente o controle de qualidade é adotada por organizações visando definir padrões, mirando a satisfação total do cliente. Basicamente consiste em acompanhar os resultados dos subprojetos e compará-los com os do projeto global para identificar desvios e apresentar correções de rota, quando necessárias, ainda durante o desenvolvimento do processo (Gerenciamento de projeto, Roberto Candido).

2.1.7 Gerenciamento de recursos humanos

Segundo o Guia da PMBOOK 5ªedição, "o gerenciamento de recursos humanos determina funções, responsabilidades e relações hierárquicas do projeto e cria o plano de gerenciamento de pessoal". Por outras, palavras permite definir e organizar as pessoas que

fazem parte da equipa de projeto. A formação da equipa, varia conforme a necessidade do projeto, podendo ser compostas por pessoas internas ou/e externa a organização, distribuídas de acordo com funções e responsabilidades, do projeto. Esse processo de gerenciamento, exige que se crie um planejamento de gerencia de pessoal que normalmente inclui, informações como o número de pessoal a ser contratado para fazer parte da equipa, identificação das necessidades de treinamento, planos de motivação e estudo com fim de extrair o máximo do potencial de cada membro de modo que todos impactem bastante nos projetos que forem inseridos. O PMBOOK define os processos que fazem parte do processo gerenciamento de recursos humanos. Na figura 2.7 passamos a ilustra-los:

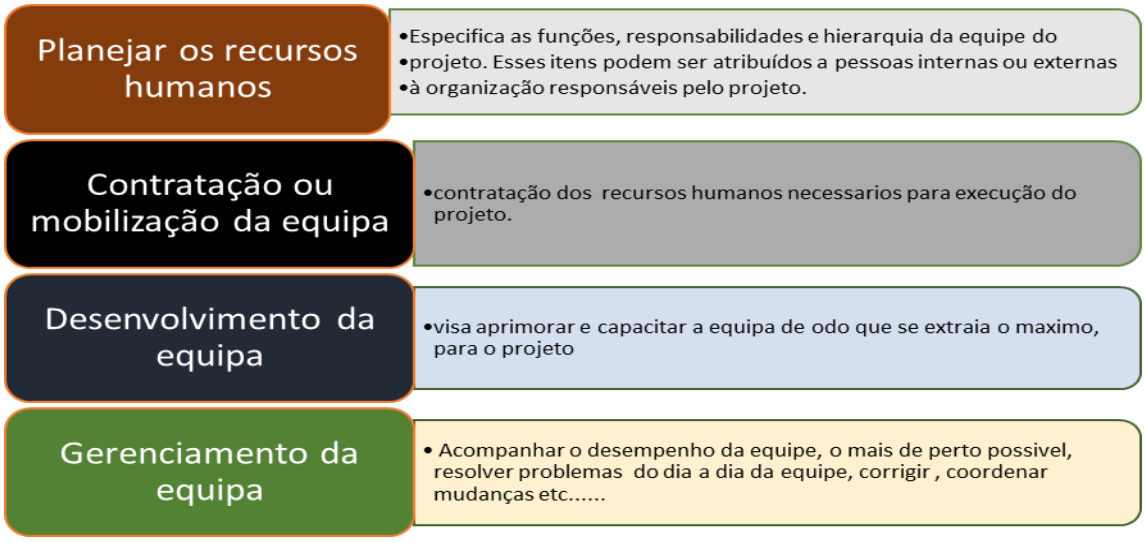


Figura 2.7 processos de gerenciamento de Recursos humanos ***Fonte****: Guia da PMBOOK,5ªedição*

- **Planejar os recursos humanos:** Segundo a guia da PMBOOK 5ªedição, essa etapa consiste em: "criar o plano de planejamento recursos humanos". Isso é documentar as funções, delegar responsabilidades de acordo com competências necessárias e relações hierárquicas. Pode definir também, o plano de treinamento, definir plano de premiação motivacional etc.
- **Contratação ou mobilização da equipe:** Basicamente essa etapa consiste na contratação e mobilização dos recursos humanos necessários para concretização das atividades do projeto. Normalmente, tais escolhas são baseadas na disponibilidade, capacidade, interesses, e custo, tudo conforme a necessidade do projeto.

- **Desenvolvimento da equipe:** O que se pretende nessa etapa, é melhorar as capacidades e competências, incluindo nas relações de interação entre membros da equipa. Parte do que se quer nessa etapa é o aprimoramento das habilidades dos membros, o fortalecimento da confiança e da coesão entre os integrantes e a melhoria da produtividade e da qualidade do trabalho.
- **Gerenciar a equipe do projeto**: Realiza o acompanhamento da qualidade e da produtividade, visando um melhor desempenho por parte da equipe fornecer feedback, assim como gerencia conflitos dentro da equipe. O guia da PMBOOK 4ªedição, chama atenção para eventuais problema de dupla subordinação, que acontece por exemplo em casos em que os membros da equipa respondem, para um gerente funcional e também para um gerente de projeto, aconselhando nesse caso, no que chamam de ampla ação de mudança da cultura da organização como medidas preventivas, evitando assim conflitos que poderiam contribuir para o insucesso do projeto.

2.1.8 Gerenciamentos das comunicações

O processo de gerenciamento de comunicações em projetos é de suma importante, no seio de qualquer organização, por essa razão qualquer gerente que se prese despende boa parte do seu tempo, pensando na forma mais eficaz de gerir a comunicação entre os elementos dentro das equipas, bem como arranjando forma de como melhorar a comunicação sob pena de evitar problemas. Basicamente um projeto não funciona sem que haja, um sistema de comunicações eficiente, devendo por essa razão o gerente de projeto, despender esforço significativo para alcançar uma comunicação eficaz entre os membros e equipas pertencentes ao projeto, facilitando a solução de problemas e agilizando a tomada de decisões. Segundo o Guia PMBOK, planejar o gerenciamento das comunicações é o processo de determinar as necessidades de informação das partes interessadas no projeto e definir uma abordagem de comunicação. O que se quer no final é garantir a geração, coleta, distribuição, armazenamento, recuperação e destinação final das informações sobre o projeto. O guia PMBOOK apresenta os processos que compõem o gerenciamento das comunicações, nós ilustramos na figura 2.8:

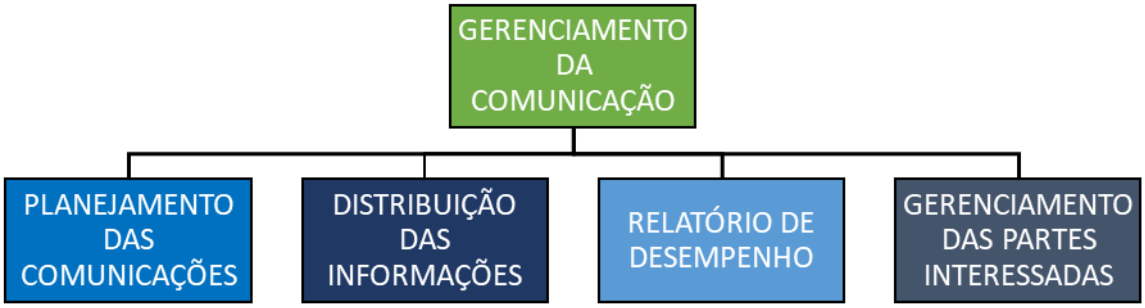


Figura 2.8 Processos de gerenciamento de comunicações fonte: Guia PMBOOK

- **Planejamento das comunicações:** Segundo a guia PMBOOK "o planejamento das comunicações determina as necessidades de informações e comunicações, e a definição de uma abordagem de comunicação". O plano de gerenciamento de comunicações deve conter informações, sobre os responsáveis pelo fornecimento das informações; as ferramentas de comunicação a serem adotadas; a periodicidade e as diretrizes para as reuniões do projeto; as ferramentas para as reuniões a distância; o cronograma de disponibilidade das informações, "Gerenciamento de projeto, roberto cândido".
- **Distribuições das Informações**: Nessa etapa o gerente de projeto trata de colocar todas as informações necessárias à disposição das partes interessadas no projeto. Por outras palavras o objetivo nessa etapa é garantir que as informações sejam amplamente distribuídas a cada membro integrante do projeto no tempo oportuno.
- **Relatório de desempenho:** Nessa etapa, faz-se a coleta e distribuição de informações sobre o desempenho, incluindo relatórios de andamento, medições do progresso e previsões. Normalmente o relatório de desempenho, são distribuídos para as pessoas envolvidas no projeto e fornece as informações sobre escopo, cronograma, custo qualidade risco e aquisições, assim como um panorama geral da situação atual das entregas, isso é progressos e custos incorridos.
- **Gerenciamento das partes interessadas**: Essa etapa prove os meios de comunicação e interação entre a gerencia do projeto e as partes interessadas visando atender as suas necessidades e solucionar as questões à medida que ocorrerem da melhor forma possível. O que se quer aqui é manter o projeto o escopo, e diminui no máximo interrupções no projeto, por problemas oriundos da falta de comunicação entre a gerencia e as partes interessadas.

2.1.9 Gerenciamentos dos riscos de projetos

Qualquer atividade ou projeto dentro de uma organização, está sujeito a ser afetado por determinados riscos, que dificultam o andamento até mesmo o termino do projeto, dentro dos prazos previamente estabelecidos. Para evitar tais riscos, as equipas de projeto precisam contar com um gerente de projeto proativo, capaz de desenvolver vários planos de contingência caso tais riscos venham a se concretizar. O Gerenciamento de Riscos identifica, analisa e da resposta a fatores de risco ao longo da vida de um projeto, visando o cumprimento do prazo dos projetos. A ideia da gestão adequada dos riscos, é sempre de reduzir probabilidade da ocorrência de problemas, e velar por um controle proativo a fim de evitar eventuais acontecimentos futuros. Com isso aumentar a probabilidade de acontecimentos de eventos positivos e reduzir os adversos. Segundo a guia da PMBOOK definir o gerenciamento de Riscos inclui: identificar os riscos, realizar a análise qualitativa dos riscos, realizar a análise quantitativa dos riscos, planejar as respostas aos riscos, monitorar e controlar os riscos.

- **Planejamento do gerenciamento de riscos:** Nessa etapa são criadas as melhores maneiras de lhe dar com os riscos nas atividades. Normalmente vem incluído no plano de Gerenciamento do Projeto estabelecido na etapa inicial do projeto. Por outras palavras é nessa etapa, que é feita o planejamento de como lhe dar, com eventuais riscos caso venham acontecer. A ideia é desenvolver tarefas ou planos de contingência curtos que podem ser postos de lado, para gerenciar riscos de maneira antecipados e rapidamente caso venham ocorrer, reduzindo assim a necessidade de gerenciar o risco por crise.
- **Identificar os riscos:** É nessa etapa em que se lista e se analisa todo o tipo e fonte, possíveis de riscos, fazendo usos de instrumentos de avaliação apropriada. Sem a identificação da quantidade de riscos, geralmente a equipa de gerenciamento excedem a capacidade de tempo da equipe do projeto para analisar e desenvolver contingências, daí a necessidade de priorizar, a resolução dos riscos com maior probabilidade de ocorrência, através de um levantamento e detalhamento dos mesmos e de suas características. No final dessa etapa espera-se obter uma lista de riscos identificados, uma lista de respostas possíveis para possíveis riscos encontrados, uma lista com causas-raiz do risco, e a categorias de risco atualizadas.
- **Análise qualitativa de riscos:** Nessa etapa basicamente é feita uma filtragem dos tipos riscos, o objetivo é fazer uma avaliação de todos os riscos, e resolve-los por ordem de

prioridade. Por outras palavras depois de identificados e listados todos os riscos**,** os com maior probabilidade e impacto são priorizados para posterior criação de um plano de respostas. Enquanto que os riscos com menor probabilidade de impacto negativo sobre o projeto, são mantidos em uma lista de observação para posterior analises e resposta.

- **Análise quantitativa de riscos:** A análise quantitativa dos riscos, nada mais é do que a análise numérica do efeito dos riscos identificados nos objetivos gerais dos projetos. É normalmente realizada sobre os riscos listados e priorizados durante a Análise qualitativa dos riscos, por serem esses que mais impactam no projeto. Normalmente faz uso técnicas como analise probabilística que fornece a equipa de gerencia informações importantes para tomada de decisões diante de incertezas ou acontecimentos de risco.
- **Planejamento de respostas a riscos:** É nessa etapa que se planeja as respostas aos riscos, ou seja, são desenvolvidas opções e ações para aumentar as oportunidades e reduzir as ameaças aos objetivos do projeto. Essas respostas aos riscos são planejadas de modos que se adequem à importância dos riscos, de modo que não sejam tão dispendiosas, assim como são planejadas de maneira que deão respostas rápidas, e realistas dentro do contexto do projeto.
- **Monitoramento e controle de riscos:** qualquer projeto é passível de Riscos, em qualquer fase. Até mesmo depois dos riscos serem mapeados e tratados de maneira de vida, ainda é possível que possam novamente. Por essa razão o processo de gerenciamento de riscos precisa ser feito de maneira continua, isso é durante todo o ciclo de vida do projeto, através de um processo de monitoramento e controle. Esse processo de monitoramento e controle normalmente inclui os planos de respostas aos riscos; monitoramento de riscos residuais; identificação de novos riscos; Avaliação da eficácia do processo de riscos durante o ciclo de vida do projeto. Para melhorar eficácia no processo de controle e monitoramento de riscos, é aconselhável que os responsáveis gerem relatórios periódico, contendo informações e respostas sobre a eficácia do plano e o registro das lições aprendidas e dos modelos de gerenciamento de riscos utilizados, que servirão de base inclusive para projetos futuros.

2.1.10 Gerenciamento das aquisições

Segundo o Guia PMBOK, o gerenciamento das aquisições do projeto é o processo inclui os processos necessários para comprar ou adquirir produtos, serviços ou resultados externos à equipe de projeto. Esse processo também contém as atividades para a administração de

contratos e pedidos de compra realizados por pessoas autorizadas pela equipe. Esse processo de gerenciamento é especialmente importante para organizações desenvolvendo projetos, que ao mesmo tempo que não podem desenvolver seus produtos, ferramentas e serviços, fazendo com que seja necessária a compra ou aquisição dos mesmos. O que se quer nesse processo é aumentar a eficiência nas aquisições dos produtos; potencializar o máximo recursos internos e externos; gerenciar da melhor maneira os custos, e reduzir os riscos relacionados uma vez que na compra sempre existem riscos grandes. Para ser eficaz, a gestão de aquisições realizada pelo gerente de projetos deve garantir: Entrega, no prazo de recursos necessários para que a equipe execute as atividades do projeto, cumprimento de prazos e requisitos dos contratos diminuindo a possibilidade de multas contratuais. O guia da PMBOOK lista 4 etapas, nesse processo de gerenciamento de projeto conforme, ilustrados na figura 2.9.

- **Planejar o Gerenciamento das Aquisições**: é nessa etapa onde o gerente documenta, todo material necessário para tomar decisões de compras do projeto. Isso é define fornecedores, especifica o que comprar, estabelece critérios de avaliação para compras etc.

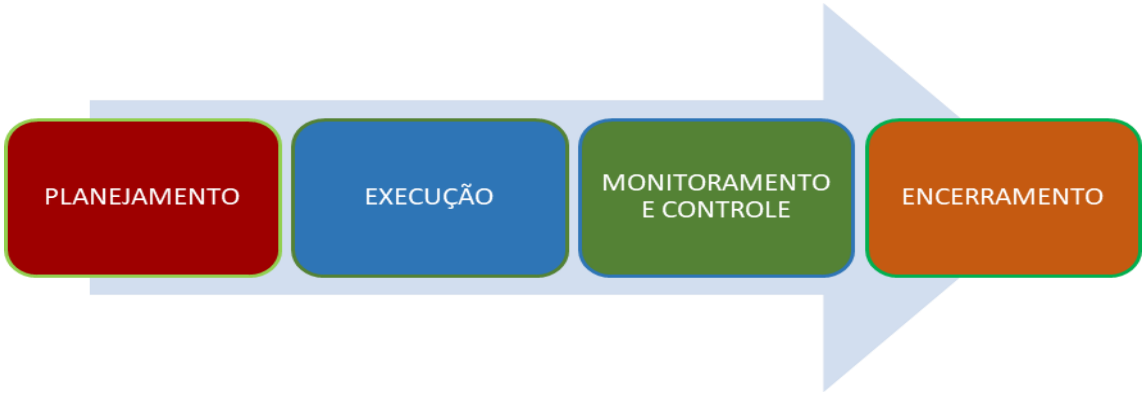


Figura 2.9 etapas do processo de gerenciamento de aquisição Fonte: adaptado, Guia da PMBOOK 5ºedição

- **Conduzir as Aquisições**: basicamente é nessa etapa em que a equipa de gerenciamento obtém as dividas respostas de fornecedores, e seleciona aquele que melhor atende as necessidades do projeto. É nessa etapa que ocorre também assinatura dos termos de fechamento de negócios.
- **Controlar as Aquisições:** é o processo de gerenciamento das relações de aquisições, monitoramento do desempenho do contrato e realizações de mudanças e correções nos contratos, conforme necessário
- **Encerrar as Aquisições**: O processo de finalizar cada uma das aquisições do projeto.

2.2 Gerente de projeto

Cada vez mais, a gestão de projeto é uma profissão que tem vindo acrescer de forma acelerada, na área de gestão organizacional, fazendo com isso que novos cargos ou seja novas tarefas organizacionais, nasçam com ela, entre elas o cargo do gerente de projeto. As regras que norteiam, guiam, os departamentos de gerenciamento de projetos, surgiram do PMI *(Project Management Institute*) através da guia PMBOOK. O PMI tem como objetivo reunir profissionais da área de gerenciamento de projetos para trocarem experiências e conhecimentos, identificar e reunir boas práticas de gerenciamento de projetos, estabelecer uma ética na profissão e certificar profissionais da área. Por si só gerir "designa a materialização de princípios, processos e funções, objetivando planejar situações futuras e seu posterior controle e avaliação da eficiência, eficácia e efetividade" (Maria de almeida de Sousa, Gestão de pessoas: uma vantagem competitiva).

Dentro de todas considerações sobre gerencia que passamos a desenvolver ao longo do nosso estudo sobre gerenciamento de projeto, baseado principalmente nos conceitos definidos pela PMI, podemos entender o gerente de projeto como um profissional no campo da gerencia de projetos que tem a função de planejar, controlar e executar projetos, em várias áreas, visando atender os requisitos de um projeto específico. Essencialmente, o gerente de projeto é responsável pelo sucesso ou fracasso de todas as ações no projeto, tendo durante a execução de um projeto, a função de definir papéis, atribuir tarefas, acompanhar e documenta o andamento da sua equipe através de ferramentas e técnicas apuradas, bem como administrar investimentos e integrar as pessoas para trabalharem juntas por um só objetivo. Basicamente o gerente de projeto atua como elo de ligação entre a estratégia da organização e sua equipa, precisando por isso de alinhar suas habilidades técnicas para gerenciar aos objetivos da organização. Gerentes de projetos trabalham visando resultados o tempo todo, para tal precisam aumentar a produtividade, aliando o tempo e dedicação da equipe. Visando uma gestão de qualidade eficaz, é interessante que gerentes estejam munidos de ferramentas e artifícios que lhes permita, otimizar e acelerar o processo de realização das tarefas do dia. Nada como mecanismos tecnológicos para melhor auxiliar gerentes nesse quesito, permitindo assim a máxima otimização das tarefas, fazendo com que se tenha um trabalho mais bem executado de maneira eficiente e passiva de menos erros.

O que seria então, o Sistema de Gerência de Projetos que nos propomos a desenvolver nesse projeto? A resposta é simples, é um software web atrelado a um aplicativo mobile, desenvolvido especificamente para auxiliar o gerente de projeto do LATITUDE, no

planejamento e controle de estudantes bolsistas, sobre sua tutela. A ideia do projeto decorre do facto de atualmente a equipe de coordenação dos projetos do LATITUDE não possuir nenhuma informatização de controle dos bolsistas, pesquisadores ou projetos, o cadastro ainda é feito manualmente e as informações documentadas sendo armazenada em pastas compartilhadas em repositório na rede LATITUDE, dificultando bastante a eficiência, nas tarefas diárias, por essa razão um software web, atrelado a um aplicativo mobile capaz de auxiliar os gerentes na coordenação de seus projetos, a automatizar essas tarefas. Foi pensando, em todos aspectos das considerações acima descrevidas que decidimos, desenvolver uma ferramenta de TI especifica (software) a fim de melhorar o processo de gerenciamento, da equipa de controle de bolsistas, do Departamento de redes nomeadamente Latitude, tentando fazer com que o trabalho do gerente atual seja menos manual e mais automatizado e por consequência mais optimizado.

É importante pontuar, que o nosso software de gerenciamento de projeto, como falaremos mais à frente, não foi projetado de modo que abranja todas as etapas do processo de gerenciamento definidas pelo Guia PMBOK, descritas ao longo dos capítulos anteriores, pois como dissemos, projetamos para uma situação especifica, no caso a gerencia de recursos humanos (estudantes bolsistas). Passamos a citar as áreas do processo de gerenciamento de projetos, que o sistema abrange:

- Gerenciamento de escopo: pois sem definição de escopo não existe projeto, definimos uma página onde o gerente pode definir todo trabalho que pretende realizar.
- Gerenciamento de tempo: uma vez que é sempre de extrema importância, de modo que se possa cumprir cada objetivo no prazo. Criamos uma página que permite ao gestor estabelecer cada projeto dentro de um tempo determinado dentro do cronograma traçado, isso é uma funcionalidade que permita registrar o início e fim de cada atividade.
- Gerenciamento de recursos humanos: O sistema em si, se baseia na gestão de recursos humanos, ou seja, uma vez o gerente tendo identificados todos os bolsistas com quem deseja trabalhar, existe funcionalidades que lhe permite, registra-los de maneira organizada, determinar funções e responsabilidades.

Com o sistema, desenvolvido o gerente funciona como um integrador, ou seja, é ele que vai fazer escolhas sobre em que pontos concentrar recursos e esforço, antecipando e tratando possíveis problemas, antes de se tornarem críticos, coordenando o trabalho visando o bem geral do projeto.

2.3 Sistemas web e aplicação móvel

O campo de Tecnologia da Informação se mostra muito dinâmico, a evolução da tecnologia ocorre no dia a dia e novos softwares se mostram mais eficientes em certas áreas, e são utilizados para isso as mais diversas ferramentas e linguagens de programação em diferentes plataformas. Os desafios que essas novas soluções trazem incluem um objetivo só que é viabilizar novas formas desenvolvimento de negócio de maneira mais simples e eficiente.
Conforme Turtschi et al. (2002) serviços web foram criados para resolver os problemas de interoperabilidade entre aplicações, de diferentes sistemas operacionais, linguagens de programação e modelos de objetos. Zhao (2010) conceitua serviços web simplesmente como uma interface programável acessível para outras aplicações através da Web. Já James F. Kurose (2006), em a Redes de Computador e internet, fala da web como sendo uma aplicação da internet que chamou atenção do público em geral, por transformar drasticamente a maneira como pessoas interagem dentro e fora de seus ambientes, de trabalho.

Atualmente, esta tecnologia vem sendo utilizada nos mais variados domínios, contudo, não foi a primeira tecnologia desenvolvida com este propósito. Anteriormente ao surgimento dos serviços web, outros padrões de comunicação já eram adotados, por exemplo, CORBA (*Common Object Request Broker Architecture*), inicialmente utilizado por sistemas UNIX e DCOM (*Distributed Component Object Model*) desenvolvido pela Microsoft.

O emprego de serviços web teve um grande impulso com o surgimento da WEB 2.0, fase na qual o conteúdo deixou de ser meramente estático para tornar-se dinâmico, dando início as aplicações web mais robustas, permitindo que os usuários passassem de expectadores a agentes ativos na construção de conteúdo. Porém, simplesmente gerar conteúdo não faz sentido se o mesmo não for compartilhado, e é justamente no compartilhamento de conteúdo entre aplicações web que o emprego de serviços web popularizou-se (FILHO, 2009).

A principal razão de optarmos por desenvolver, um software voltado para web, é devido ao facto de os sistemas web serem as melhores soluções para tornar processos manuais mais simples, rápidos e eficazes, otimizando o tempo gasto na execução de tarefas, além de que permite que se detecte falhas automaticamente, eliminando a necessidade de revisão e de erros, fora ao facto de poder ser armazenado em qualquer provedor de hospedagem, aliviando os gastos com infraestrutura. Outro fator importante que levamos bastante em consideração quando optamos pelo desenvolvimento de um sistema web, tem a ver com a segurança. Mais do que qualquer outra aplicação, devido ao fato de ser hospedado em locais especializados, e contar com certificados de segurança e senhas de acesso criptografadas o banco de dados do

sistema web fica mais seguro do que as aplicações normais. Outra razão para optarmos por desenvolver, um software voltado para web é a questão de poder ser personalizado, ou seja, ajustado as necessidades de determinado cliente ou organização, isso nos permitiu que desenvolvêssemos um software, voltado especificamente as necessidades do Departamento. E finalmente outra vantagem de se usar sistema web, é a mobilidade uma vez que são on-line, permitindo assim que seja acessado de qualquer lugar, através de Computadores *Desktop, Notebooks, Tablets ou Smartphones,* basta que o usuário esteja logrado a Internet. Resumidamente escolhemos desenvolver um sistema baseado em Web pelas seguintes razões: Mais fácil de instalar e manter seguro; mais útil para seus usuários; mais seguro; mais fácil de desenvolver; e finalmente por ser acessível de qualquer lugar.

Como o próprio nome sugere um aplicativo móvel ou aplicação móvel nada mais é do que um software desenvolvido para ser instalado em um dispositivo eletrônico móvel, como *tablets*, smartphones ou até mesmo em um Leitor de MP3. Os aplicativos moveis por norma, podem ser gratuitos ou pagos, alguns podendo vir pré-instalado nos dispositivos moveis através das configurações de fábrica, e outros podem ser baixados pelos clientes de várias plataformas de distribuição de software móvel ou aplicativos da web, basta que o dispositivo eletrônico esteja conectado à Internet. As companhias desenvolvedoras de aplicativos, normalmente disponibilizam, aplicativos para download, quer os pagos quer os gratuitos através de lojas virtuais como a *Apple Store* no caso da *Apple*, *Windows Phone Store*, loja virtual da *Microsoft* para *Windows Phone etc*. A medida que o número de usuários, de smartphone aumenta no mundo, aumenta também a quantidade de pessoas usuários de aplicativo moveis. Por essa razão, o número de empresas fabricantes aplicativos móveis está em forte crescimento. Por existir várias marcas de dispositivos no mercado, faz com que exista também, múltiplas plataformas de desenvolvimento, fazendo com isso, que exista uma variedade de aplicativos, cada um codificado para ser executado sob sua arquitetura específica. Existem várias plataformas, através dos quais, é possível, desenvolver um aplicativo. Por norma cada dispositivo móvel suporta apenas uma plataforma em particular. Por essa razão é preciso, que o desenvolvedor decida cuidadosamente sobre qual plataforma usar. Para nossa aplicação, optamos por desenvolver o aplicativo, para rodar em dispositivos que usam plataforma iOS. A *iOS* é um dos sistemas operacionais para dispositivos móveis desenvolvido e criado pela Apple Inc. essa plataforma oferece a maior quantidade de aplicativos móveis disponíveis atualmente, sendo que muitos são universais, podendo rodar em mais de um tipo de dispositivo *iOS (iPod, iPhone e iPad)*, ou em todos ao mesmo tempo. A arquitetura do iOS é formada por quatro camadas,

sendo que cada uma delas oferece um conjunto de frameworks que podem ser utilizados durante o desenvolvimento de aplicativos para os dispositivos moveis da *Apple Inc*.

Os aplicativos desenvolvidos para o iOS por norma não se comunicam diretamente com o hardware do dispositivo, ao invés disso, os aplicativos se comunicam com o hardware através de um conjunto de interfaces de sistema bem definidas que protegem seu aplicativo. As camadas da arquitetura iOS são nomeadamente, a *Cocoa Touch, Media, Core Service e Core OS,* sendo que cada uma delas oferece um conjunto de frameworks que podem ser utilizados durante o desenvolvimento. Nas camadas superiores estão as tecnologias e serviços mais sofisticados, ou seja, os frameworks que fornecem abstração orientada a objetos das camadas de níveis inferiores. Nas camadas inferiores do sistema estão os serviços fundamentais e as tecnologias dos quais todos os aplicativos dependem. Normalmente existem três possibilidades, de escolha sobre qual tecnologia optar quando queremos desenvolver um aplicativo, nomeadamente: aplicação web, aplicação híbrida ou aplicação nativa. Para o nosso projeto optamos por desenvolver via aplicação hibrida. Em seguida passaremos a conceituar as três a fim de acentuar as pequenas diferenças e melhor justificarmos a nossa escolha.

Uma aplicação nativa é uma aplicação para um determinado dispositivo móvel (*smartphone, tablet*, etc) e estas são instaladas diretamente no aparelho. O aplicativo nativo fica armazenado no dispositivo e é baixado diretamente pelas lojas, *Google Play* (Android) ou *App Store* (iOS). Pode ser desenvolvido pelas linguagens nativas destas plataformas e podem utilizar as funcionalidades específicas do dispositivo, como câmera, GPS, contatos, etc, além de possuir uma integração diretamente com as bibliotecas de cada um deles. Enquanto isso quando discorremos sobre aplicações *web* móveis, fazemos referência a aplicações com acesso à internet que têm funcionalidades específicas para dispositivos móveis. Estas são acedidas por meio do navegador *web* e não precisam ser instaladas no aparelho. Diferente do aplicativo nativo e do híbrido, A aplicação web não utiliza as funcionalidades do dispositivo, como câmera e GPS, além disso não ocupa espaço na memória do aparelho, a visualização é por um navegador e só funciona se estiver conectado à internet. O design é de uma aplicação web com a interface para dispositivo móvel é o custo que tem melhor custo benefício comparado a aplicação nativa, por exemplo. Enquanto isso o aplicativo híbrido, diferente do nativo, não é desenvolvido completamente na linguagem específica de cada sistema operacional, ao invés disso o *app* utiliza várias linguagens e engloba dois formatos: é metade nativo e metade web *app*. Assim como os nativos, também pode ser baixado pelas lojas e utilizar as funcionalidades do dispositivo, mas a diferença é que também permite o acesso através da aplicação *Web*,

inserindo uma página *web*, por exemplo, onde as informações são integradas do site para o *app*. Portanto apesar do aplicativo em si ter característica totalmente nativa, o nosso projeto como um todo, é concessionado para funcionar de maneira "Hibrida ", uma vez que além de desenvolvermos o aplicativo de maneira totalmente nativa, atrelamos a ele um sistema web, isso é, uma parte das metas estabelecidas nos objetivos, como o *CRUD* por exemplo é implementada apenas na parte web, onde só o gestor tem acesso e consegue adicionar Bolsistas, Projetos e Atividades. Sendo que pelo *app*, o bolsista apenas visualizar seus dados e não consegue editar, o que seria função apenas do administrador.

2.4 Linguagem de Programação e Softwares

2.4.1 Java script

De maneira geral, cada plataforma de desenvolvimento de aplicativos móveis (ex.: iOS, Android) requer o seu próprio processo de desenvolvimento. Isso faz com que cada plataforma de desenvolvimento de aplicações móveis tenha a sua própria linguagem de programação nativa. Para o desenvolvimento da nossa aplicação utilizaremos o *React Native* um software desenvolvido pelo *Facebook* que usa como linguagem de programação o Java script.

Quando desenvolvemos páginas web, utilizamos como base a linguagem de marcação de hipertexto (*HTML*), que forma toda a estrutura do documento a partir de tags, trechos de código que são identificados e interpretados pelo navegador para montar algum elemento. Aliando essa estrutura às CSS (*Cascading Style Sheets*), é possível estilizar os elementos da página, garantindo uma formatação visual mais completa e de fácil manutenção. Temos assim uma página bem estruturada, com vários elementos visuais (e não visuais) como formulários e tabelas bem formatados, com bordas, cores de plano de fundo, imagens, etc., porém, não vamos muito além disso, utilizando apenas *HTML e CSS* se obtemos apenas documentos estáticos, sem interação com o usuário, como caixas de diálogo e validação de entrada de dados, onde é introduzido o *Java Script*.

O *Java Script* (às vezes chamado apenas de JS) é uma linguagem de programação, leve, interpretada, orientada a objetos, baseada em protótipos e em *first-class functions* (funções de primeira classe), mais conhecida como a linguagem de *script* da *Web*. *O Java Script* nos permite implementar itens complexos em páginas *web*, mostrando conteúdo que se atualiza em um intervalo de tempo, mapas interativos ou gráficos 2D/3D animados, etc. Ela foi criada basicamente para dar vida (do lado do cliente – *front-end*) em seu navegador. Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do

lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido. O núcleo da linguagem Java Script consiste em alguns benefícios comuns da programação que nos permite fazer coisas como:

Armazenar conteúdo útil em variáveis. Operações com pedaços de texto (conhecidos como "*strings*" em programação). Executar o código em resposta a determinados eventos que ocorrem em uma página da Web. Outra funcionalidade empolgante na linguagem Java Script construída no topo do núcleo são as API's (*Application Programming Interfaces* - Interface de Programação de Aplicativos) que nos provêm várias funcionalidade adicionais . Elas geralmente se dividem em duas categorias:

- **APIs de navegadores**: aquelas que já vem implementadas no navegador, e são capazes de expor dados do ambiente do computador, ou fazer coisas complexas e úteis.

Exemplo: a API DOM (*Document Object Model*) que nos permite manipular HTML e CSS, criando, removendo e mudando HTML, aplicando dinamicamente novos estilos para página, como é o caso toda vez uma janela pop up aparece em uma página. Outro exemplo seria a API de Geolocalização que recupera informações geográficas, é através dessa API que o Google Maps consegue encontrar nossa localização e colocar em um mapa.

- **APIs de terceiros**: que são aquelas que não estão implementados no navegador automaticamente. Para serem implementadas geralmente pega-se no código informações em algum lugar da rede. Podemos pegar como exemplo a API do *Twitter* que permite exibir os últimos *tweets* em um *website*.

Por fim de maneira geral podemos dizer que o *Java Script* é atualmente a principal linguagem para programação *client-side* em navegadores *web*, porem começa também a ser bastante utilizada do lado do servidor através de ambientes como o Node.js.

2.4.2 Node.Js

O Node.js é uma plataforma construída sobre o motor Java Script do Google Chrome para facilmente construir aplicações de rede rápidas e escaláveis. Node.js usa um modelo de *Input/Output* direcionada a evento não bloqueia e que o torna mais leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos. O *Nodejs* foi construído em cima da *engine V8* que interpreta Java Script, criado pela Google e usado em seu navegador Chrome, significando que utilizará a linguagem pelo lado do servidor também, e não só pelo browser, como normalmente ocorre. Além disso, ele usa uma arquitetura voltada a eventos, o que se integra muito bem com Java Script (*callbacks*!). Usando um *loop*

de eventos, o Node interpreta em uma única *thread*, as requisições de forma assíncrona em vez de sequenciais, e não permitindo bloqueios. Isso o torna incrivelmente rápido, perfeito para lidar com um número muito alto de requisições. O *loop* de eventos do Node recebe várias requisições, como por exemplo, buscar alguma informação no banco, ler um arquivo no servidor, etc. Depois de executar uma das ações, em vez de esperar a resposta (como esperar o banco responder), ele passa a processar a próxima requisição. Quando a resposta de uma delas é retornada, é disparado um evento e o *callback* correspondente da requisição, ou seja, a função que deve ser executada como resultado da resposta, é posta na fila para ser executada assim que possível. Devido sua natureza Node.js é ótimo para realizar diversos tipos de projetos, como: APIs; Aplicações web *real-time* como servidores de chat ou aplicações colaborativas entre múltiplos usuários como o *Firebase* que utilizamos neste projeto; Aplicações que demandam alta escalabilidade; Servidores de streaming de dados e etc.

2.4.3 Diagrama UML

O UML (*Unified Modelling Language*), em português linguagem de modelagem unificada é uma linguagem diagramática, utilizável para especificação, visualização e documentação de sistemas de software (Alberto Manuel e Carlos Alberto UML, Metodologias e Ferramentas CASE, 2001). A UML foi criada para estabelecer uma linguagem para arquitetura, design e implementação de sistemas de software complexos, tanto estruturalmente quanto para comportamentos. A UML é promovida pela OMG, e segundo a mesma com o propósito de fornecer a arquitetos de sistemas, engenheiros de software e desenvolvedores de software ferramentas de análise, design e implementação para sistemas baseados em software, por um lado, e por outro lado desenvolver as condições gerais da indústria ao permitir a interoperabilidade de ferramentas de modelagem visual de objetos. Por norma existem vários tipos de diagramas UML usados para documentar e modelar diversos aspectos dos sistemas, podendo ser divididos em Estruturais e Comportamentais.

Na categoria dos diagramas estruturais temos: diagrama de classes; diagrama de componentes; diagrama de estrutura composta; diagrama de implementação; diagrama de objetos; diagrama de pacotes.

Na categoria de diagramas UML comportamentais tem-se: diagramas de atividade; diagrama de comunicação; diagrama da visão geral da interação diagrama de sequência; diagrama de máquina de estados; diagrama de tempo; diagrama de caso de uso.

Para o nosso projeto estamos interessados apenas em dois tipos de diagramas UML, o diagrama de classe da categoria de diagramas estruturais, e o diagrama de caso de uso na categoria de diagramas comportamentais.

- **Diagrama de Classes:** Ilustra o conjunto de classes, do sistema com seus atributos e métodos e forma como as classes estão relacionados entre si, permitindo assim que se especifique uma estrutura estática de um sistema segundo a abordagem orientada por objetos. Normalmente um diagrama, de classe, contém um nome para classe, um ou vários atributos, com o referido tipo de dados, um ou vários métodos com suas funções e parâmetros, e finalmente associação, que é forma como as classes estão interligadas.
- **Diagrama de casos de usos:** ilustra a funcionalidade do sistema, mas vista do ponto de vista do usuário. Ou seja, ilustram uma situação pratica de uso do sistema, casos de utilização, que representam a visão do sistema na perspectiva do seu utilizador. Diagramas de Casos de Uso são compostos basicamente por quatro partes:
    1. Cenário: Sequência de eventos que acontecem quando um usuário interage com o sistema.
    2. Ator: Usuário do sistema, ou melhor, um tipo de usuário.
    3. Use case: É uma tarefa ou uma funcionalidade realizada pelo ator (usuário).
    4. Comunicação: é o que liga um ator com um caso de uso.

2.5 Software

2.5.1 React Native:

Para o desenvolvimento da nossa aplicação utilizamos o *React Native*, um novo módulo introduzido na biblioteca principal (ReactJS) lançado pelo Facebook para o desenvolvimento de aplicativos móveis múltiplataforma. O *React Native* é um projeto baseado em Java Script que consiste em uma série de ferramentas que viabilizam a criação de aplicações móveis nativos, utilizando o que há de mais moderno no desenvolvimento *front-end* mirando no futuro. Apesar das desconfianças por se tratar de uma biblioteca nova o *React Native* é considerado por muitos desenvolvedores eficiente e recomendável, principalmente por duas razões:

Com o *React Native* a aplicação, bem como toda a lógica interna a ela, é toda escrita e executada em Java Script, o que nos possibilita ter um UI (*User Interface*) totalmente nativo. Logo, não é necessário que o desenvolvedor assuma nenhum compromisso tipicamente associado ao UI da HTML5.

O *React* introduz uma abordagem nova e altamente funcional para a construção de interfaces de usuário: a interface do usuário é simplesmente expressa como uma nova função do estado atual do aplicativo. O *React Native* faz uso do Node.js para efetuar os builds de código *Java Script.*

O *React Native* surge como evolução do React, "uma biblioteca Java Script declarativa, eficiente e flexível para a criação de interfaces de usuário (UI) "com funcionalidades relacionadas que podem ser chamadas pelo desenvolvedor para resolver problemas específicos, como a criação de interfaces de usuário reaproveitáveis. Porem apesar de que os código e meios de uso do *React Native* e do React serem bastantes semelhantes com declarações Java Script relativas às UIs, por detrás de tudo as interfaces do mesmo tem suporte em *controllers* específicos da biblioteca, em vez de em elementos do objeto DOM como é o caso do React. O ponto chave do *React Native* é que ele pretende trazer principalmente o poder do modelo de programação do React para o desenvolvimento de aplicativos móveis. O maior objetivo dele é implantar o conceito de uma plataforma *learn-once run-anywhere* (aprenda uma vez e execute em qualquer lugar), isto é, uma vez aprendidos os conceitos do React, os mesmos podem ser transcritos para qualquer linguagem de programação, tal como funciona com a lógica de programação.

2.5.2 Native Base

NativeBase é uma biblioteca de componentes de interface livre e de código-fonte aberto para o *React Native*, utilizada para criar aplicativos móveis nativos para plataformas iOS e Android. *NativeBase* também suporta web a partir da versão 2.4.1, e possui uma estrutura de *front-end* dinâmica. Nossa escolha por ele tem a ver com o facto de nos permitir usar componentes de plataforma cruzada compartilhados da interface do usuário, o que aumenta drasticamente nossa produtividade. Ao usar o *NativeBase,* poderíamos usar qualquer biblioteca de terceiros nativa fora da caixa. O *NativeBase* é feito de blocos de construção eficazes chamados de componentes. Os componentes são construídos em pura plataforma *React Native,* juntamente com algumas funcionalidades *Java Script* com um rico conjunto de propriedades personalizáveis. Esses componentes permitem que criemos com mais facilidades a melhor interface.

2.5.3 Firebase

2.5.3.1 O que é o Firebase

O Firebase é uma plataforma da google que provê várias ferramentas necessárias para desenvolver aplicações *mobile* (Android e iOS) e aplicações web. Ele é o que chamamos de BaaS, o que quer dizer que o software é utilizado como um serviço. BaaS permite ao desenvolvedor focar no *frontend*, pois provê uma infraestrutura completa de *backend*. O Firebase é um *backend* completo pois oferece serviços ilimitados de analise, uma base de dados em tempo real, autenticação de usuário, relatórios de erros, envio de notificações e ainda mais serviços.

2.5.3.2 BaaS

"*Backend as a Service*" (BaaS) é um termo que deriva de "*Software as a Service"* (SaaS) um software como serviço, onde você paga para poder utilizar. Um conceito que pode ser estendido para o BaaS que pode ser entendido como mais que um software. Além de fornecer uma infraestrutura de *backend* completa, ele é uma plataforma que providencia tudo que vá além da camada de apresentação de uma aplicação. Com ele se adquire não só parte da aplicação, mas também a parte de segurança e infraestrutura da aplicação

2.5.3.3 Principais recursos do Firebase:

- **Em tempo real:**

  O Firebase permite que você guarde e sincronize dados em tempo real, o que torna seu acesso muito mais simples de qualquer lugar, *web* ou móvel. Sempre que os dados são alterados, todos os dispositivos conectados recebem essa atualização em milissegundos.
- **Off-line:**

Os aplicativos que utilizam o *Firebase* como um serviço de *backend* permanecem responsivos mesmo off-line, pois o SDK do *Firebase* mantém seus dados em disco. Quando a conectividade é restabelecida, o dispositivo cliente recebe as alterações perdidas e faz a sincronização com o estado atual do servidor.

- **Acessível em dispositivos clientes:**

A plataforma pode ser acessada diretamente de um dispositivo móvel ou navegador, sem um servidor de aplicativos. A segurança e a validação de dados estão disponíveis por meio

de regras de segurança baseadas em expressão da própria plataforma, executadas quando os dados são lidos ou gravados.

- **Escalonar entre vários bancos de dados:**

Oferece suporte em grande escala às necessidades de dados do app. Para isso é preciso dividir os dados entre várias instâncias de banco de dados no mesmo projeto do *Firebase*. Simplificando assim a autenticação no projeto com o *Firebase.*

- **Princípio de funcionamento:**

É possível criar aplicativos avançados e colaborativos, ao conceder acesso seguro ao banco de dados diretamente do código do cliente. Os dados são mantidos em disco e, mesmo off-line, os eventos em tempo real continuam sendo acionados, proporcionando uma experiência responsiva ao usuário final. Quando o dispositivo recupera a conexão, o serviço de banco de dados em tempo real sincroniza as alterações feitas nos dados locais com as atualizações remotas que ocorreram enquanto o cliente estava *off-line*, mesclando qualquer conflito automaticamente. Além de uma linguagem de regras flexíveis baseadas em expressão, denominadas regras de segurança, para definir como os dados são estruturados e quando podem ser lidos e gravados. Por meio da integração com o Firebase *Authentication*, os desenvolvedores podem definir quem tem acesso, a quais dados e como esses dados podem ser acessados. O banco de dados utilizado pelo Firebase não é o comum conhecido por outras aplicações. Eles utilizam o *NoSQL*, que tem otimizações e funcionalidades diferentes de um banco de dados relacional, a principal delas é a escalabilidade. A API que o Firebase utiliza foi desenvolvida para autorizar apenas operações que possam ser executadas com rapidez. Isso possibilita uma ótima experiência em tempo real que atende a milhões de usuários sem comprometer a capacidade de resposta.

2.5.4 Visual Code

Para editar o nosso código usamos o Visual code ou simplesmente VScode um editor de código totalmente gratuito e *open source* que foi lançado ano de 2015 pela equipa da Microsoft e é destinado ao desenvolvimento de aplicações web. O *VScode* trata-se de uma ferramenta leve e multiplataforma que está disponível tanto para Windows, quanto para Mac OS e Linux e atende a uma gama enorme de projetos, não apenas ASP.NET, como também Node.js. Adicionalmente, o editor possui suporte à sintaxe de diversas linguagens como Python, Ruby e C++.

Alguns pontos interessantes ao se trabalhar com o *VSCode* é que com ele é possível abrir tanto um único arquivo como uma pasta completa e pode utilizar os arquivos abertos como base para oferecer opções no *IntelliSense* dinamicamente, como é o caso dos arquivos JSON de configuração dos projetos *ASP.NET 5*. O *VScode* possui um layout simplificado e intuitivo, o objetivo é maximizar a área do editor, buscando deixar mais espaço para a navegação e acesso completo ao contexto da pasta ou do projeto. A interface de usuário (UI) é dividida em quatro partes principais: Editor; *Side Bar, Status Bar*: e *View Bar*.

A baixo passamos a listar algumas das funcionalidades mais importantes trazidas pelo *VScode*:

- *Intellisense: o VScode* traz um sistema inteligente de auto complete para variáveis e métodos, além de mostrar várias informações daquele método/variável.
- *Debugging*: Com *Vscode* é possível depurar aplicação, adicionar breakpoints, identificar valores dentro da *call stack etc,* diretamente do editor.
- *Code Navigation e Refactor: Tras o Go to* definitivo, que nos permite saber o que um determinado método faz, além de permitir mudar uma variável/método em todas as referências de todos os arquivos a um clique.

Integração com *o Git:* facilita e permite visualizar as diferenças de códigos compartilhados na núvem de forma mais clara. Além de possuir vários *plugins* incríveis, que nos permite transformar um editor simples numa ferramenta de trabalho extremamente poderosa, como é o caso por exemplo.

2.5.5 EXPO

De modo geral para desenvolvermos e testarmos aplicações móvel com *React Native* não precisamos instalar a SDK do Android ou o XCode seus respectivos emuladores, ao invés disso, existe uma aplicação denominada Expo que possui um aplicativo móvel instalável pelas lojas do Android/iOS que contém todo código nativo necessário pelo *React Native* para iniciar uma aplicação. A vantagem é que nesse formato o desenvolvedor inicia muito rápido. Em menos tempo conseguimos ser mais produtivos e eficientes.

O Expo é uma ferramenta utilizada no desenvolvimento mobile com React Native que permite o fácil acesso às API's nativas do dispositivo sem precisar instalar qualquer dependência ou alterar código nativo. Optamos por usar o Expo pela questão do tempo, pois normalmente quando iniciamos no desenvolvimento mobile percebemos que o número de API's e recursos nativos que podemos controlar através da nossa aplicação é gigante, e muitas

vezes não nos recordamos de todas opções que temos disponíveis. O Expo nos oferece grande parte desses recursos de forma nativa e integrada.

3 METODOLOGIA E IMPLEMENTAÇÃO

3.1 Delimitação do tema

O desenvolvimento deste projeto teve como base a proposta de desenvolvimento de um sistema web para realizar algumas tarefas essenciais para o laboratório Latitude, situado na Universidade de Brasília. A ideia veio de fazer um aplicativo móvel que pudesse ao mesmo tempo se conectar com o sistema web de forma distribuída e trazer as mesmas informações, sem muitas dificuldades. A execução do projeto foi dividida em cinco etapas, para assim facilitar seu desenvolvimento, sendo distribuídas da seguinte forma:

- Etapa 1 – Coletar, analisar e definir as características e necessidades da equipe de coordenação de projetos do LATITUDE.
- Etapa 2 – Busca de referências sobre as ferramentas a serem utilizadas, a linguagem de programação, o SGBD, o framework a ser usado de base e as demais ferramentas necessárias.
- Etapa 3 – Preparar o ambiente de trabalho, ou seja, fazer as instalações de software necessárias.
- Etapa 4 – Desenvolvimento do código.
- Etapa 5 – Testes do sistema web/aplicativo móvel.

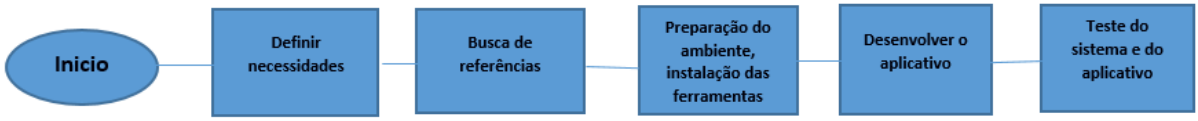


Figura 3.1 Etapas do Projeto

A metodologia de criação do projeto foi baseada em um termo que designa quatro operações básicas em Banco de Dados. CRUD (*Create, Read, Update* e *Delete*) é uma sigla dita para mostrar o que uma aplicação web pode fazer: Inserir, consultar, alterar e deletar.

3.1.1 Coleta, analise e definição das necessidades

Atualmente, o laboratório LATITUDE possui uma questão de praticidade e otimização do processo de controle de projetos realizados por eles. Ao reunir com o responsável pelo gerenciamento dos bolsistas e pesquisadores, coletamos alguns dados a fim de definir as necessidades de praticidade para o gerenciamento de projetos e bolsistas. Desvendando o funcionamento do processo e o registro da documentação envolvida no mesmo, foi visto que a

pesquisa de bolsistas não é eficaz e que os dados armazenados ficam em um repositório e não em uma base de dados. Essas e outras questões podem ser aperfeiçoadas com a criação de um sistema para automatizar parte desse processo, uma vez que isso gera um desgaste a mais e que pode ser evitado com o desenvolvimento de uma ferramenta *web*.

Nesse processo, a maior parte da documentação dos bolsistas é impressa e feita manualmente, gerando um certo desgaste e lentidão, dado que o gerenciamento é feito de forma manual. Assim como o gestor para encontrar os bolsistas também necessita de uma pesquisa manual, feita através de planilhas. A ideia principal do projeto, é melhor a eficácia de pesquisa e diminuir a necessidade do cadastro manual.

Observando essas necessidades do LATITUDE chegamos às funcionalidades básicas que precisam ser implementadas nesse sistema, essas funcionalidades estão listadas abaixo:

- **Automatização do processo de cadastro dos bolsistas:** O processo de cadastro dos bolsistas é feito manualmente, e existe uma dificuldade para a pesquisa de bolsistas e a filtragem desses dados. Na aplicação, esse processo encontra-se parcialmente automatizado, visto que é necessário que o bolsista entre no sistema se cadastre e atualize seus dados. Com um aplicativo isso seria possível de maneira rápida, o administrador ou o próprio consegue cadastrar os dados necessários para registro no sistema. O resultado do cadastro fica gravado na base de dados e pode ser acessado por qualquer usuário que tenha *login* e acesso ao sistema. Assim, o sistema web comete com que o processo, que atualmente é feito totalmente de forma manual com a utilização de formulários impressos pela equipe de coordenação, seja efetuado de forma mais rápida e prática.
- **Pesquisa de bolsistas:** Existe sempre a dúvida de algum dado do bolsista, ou a necessidade de entrar em contato com o bolsista, ou até mesmo saber os vínculos do bolsista com algum projeto específico. O projeto apresentou alguns desafios, de como saber se algum bolsista alterou algum dado com o sistema já em funcionamento, e se os registros dos planos deveriam ser digitalizados e salvos no sistema. Quanto a mudança de dados, torna-se possível pelos recursos oferecidos pelo *Realtime Database* do Firebase, olhar a movimentação de usuários no sistema e daí verificar se houveram ou não alterações de cadastro. Quanto as demais funções de registro serão implementadas em trabalhos futuros com o sistema.

3.1.2 Busca de referências

A busca para encontrar as ferramentas que seriam utilizadas no projeto deu-se da seguinte forma - feita a análise de necessidades do projeto e coletadas as informações

necessárias, discutiu-se baseado nas limitações de conhecimento de programação da equipe qual era o melhor caminho a seguir e buscou-se inspiração em projetos prontos sobre gerenciamento de projetos. A maioria dos projetos observados era em linguagens de programação desconhecidas pela equipe, assim como a maior parte das ferramentas utilizadas nesses projetos. Aprender uma nova linguagem de programação e desenvolver um sistema demanda muito tempo e, para cumprir os objetivos dentro do prazo dado para o projeto, foi necessário uma série de testes de ferramentas até descobrirmos um caminho ideal.

O primeiro passo foi descobrir uma linguagem de programação simples e que pudesse ser aprendida em um curto tempo, que obtivesse um *framework* e que juntos fossem práticos o suficiente para se desenvolver o aplicativo com pouco conhecimento específico. Após diversos testes, escolhemos o *React Native* e o *Firebase* para desenvolver nossa aplicação.

O *React,* por ser uma biblioteca em *Javascript*, traz recursos que permitem um desenvolvimento mais simples. A declaração, automatização e separação são 3 recursos fundamentais para a escolha do *React* como ferramenta de desenvolvimento. A declaração no *React* é uma maneira simplificada de declarar telas e seus vínculos, e a automatização é a atualização automática da tela vinculada aos dados que foram mudados. A separação é ter cada elemento visual reaproveitado em diversas partes da aplicação sem muito trabalho.

O *Firebase* por ser um *BaaS,* como já foi explicado, é um serviço que roda online e foi escolhido por ter uma grande dimensão de recursos para a aplicação. Ele traz inúmeros prós para o desenvolvimento de um aplicativo. O primeiro é a economia de tempo, visto que o prazo era curto para desenvolver toda uma aplicação, além de uma API que necessitaríamos para desenvolver um sistema web e um aplicativo que seus dados estivessem sincronizados, recursos que são oferecidos pelo *Firebase*. Ele traz uma certa segurança, considerando que os dados que estarão na nuvem da Google. Por estar na nuvem também pode-se considerar a redução de custos, visto que você não terá que se preocupar com servidores e por assim dizer disponibilidade e escalabilidade.

3.1.3 Preparar o ambiente de trabalho

Para reparar o ambiente de trabalho para o desenvolvimento do aplicativo, foi necessário considerar que o sistema operacional utilizado por nós seria o Windows. Então foi essencial instalar diversos softwares em nossas máquinas. A maioria dos softwares instalados eram executáveis do Windows, com exceção de alguns que foram instalados via CMD.

Durante a instalação foram feitas as verificações necessárias para o assegurar a correta instalação dos softwares.

- **Node JS:** Considerando o sistema operacional mencionado, os passos de instalação foram simples, primeiro descarregou-se o executável diretamente do site[www.nodejs.org] e depois executou-se, após isso verificamos pela figura 3.2 o software já instalado.

Figura 3.2 Print de tela do prompt do Nodejs depois da instalação

- **Visual Studio Code:** Segue o mesmo esquema de instalação anterior. Essa é uma das ferramentas mais importantes para o desenvolvimento e testes feitos no aplicativo. No próprio Visual Studio Code, é possível ter uma tela do CMD acoplado a área de trabalho do software, e isso possibilita emular o aplicativo para qualquer dispositivo. Abaixo segue a tela do software em funcionamento.

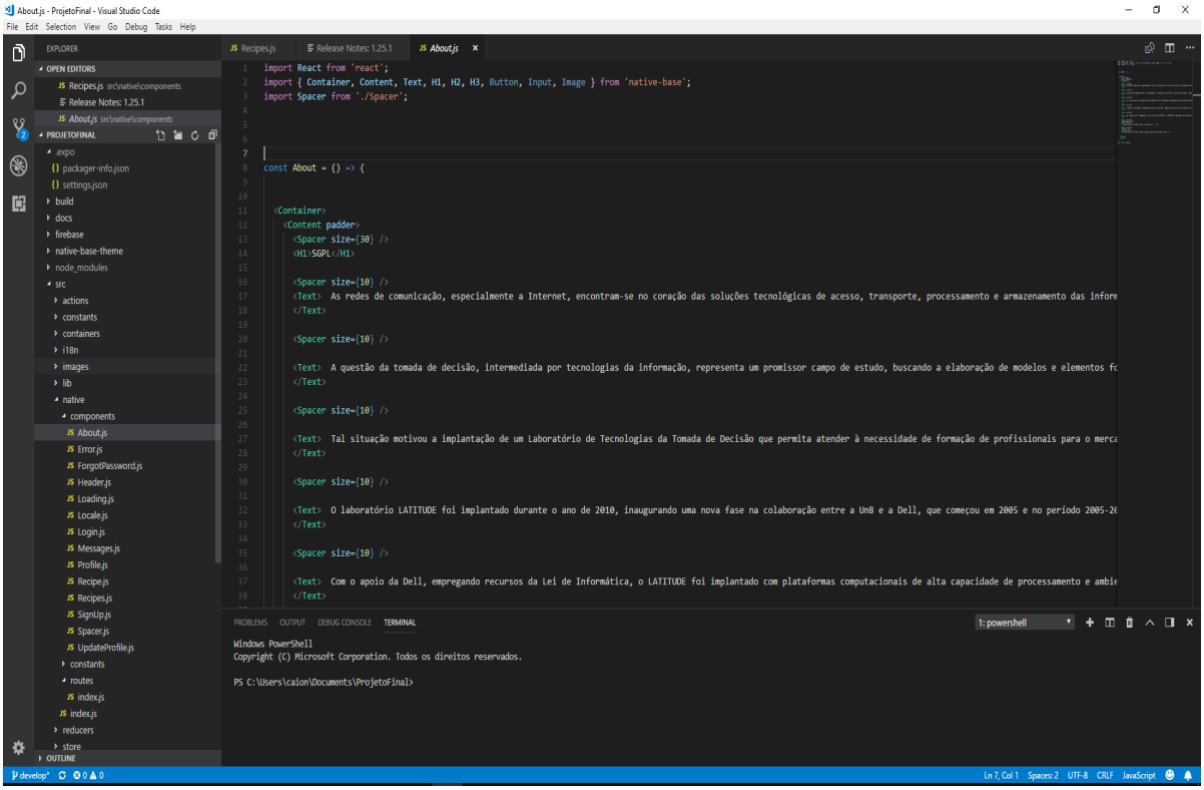


Figura 3.3 Print de Tela do Visual Studio Code

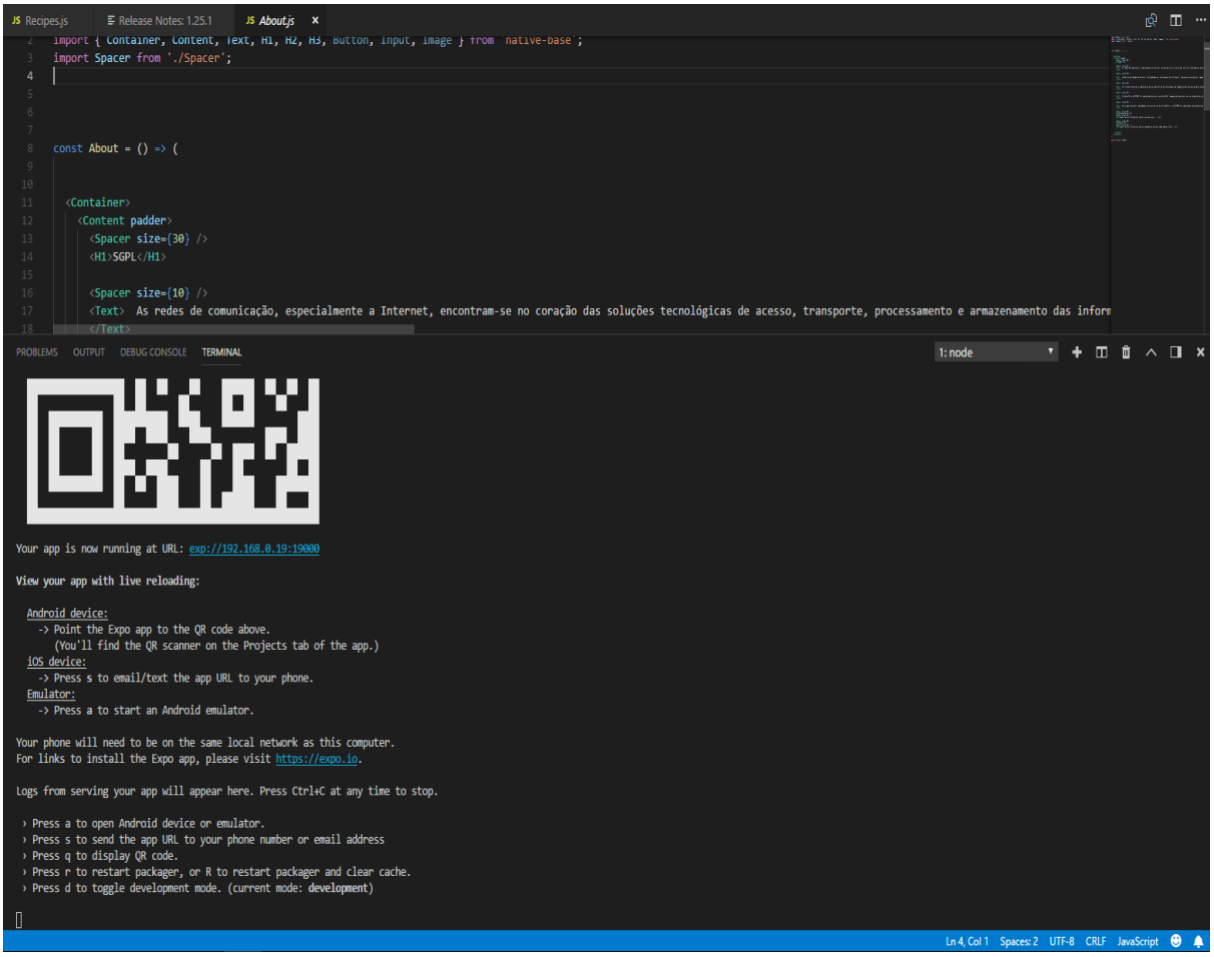


Figura 3.4 Print de tela do Visual Studio Code depois do npm start.

- **Expo XDE:** Segue o mesmo esquema de instalação anterior. Uma ferramenta para o ambiente de trabalho imprescindível, pois com ela é possível simular o aplicativo em qualquer dispositivo, utilizando um link local, SMS ou até mesmo QR Code.

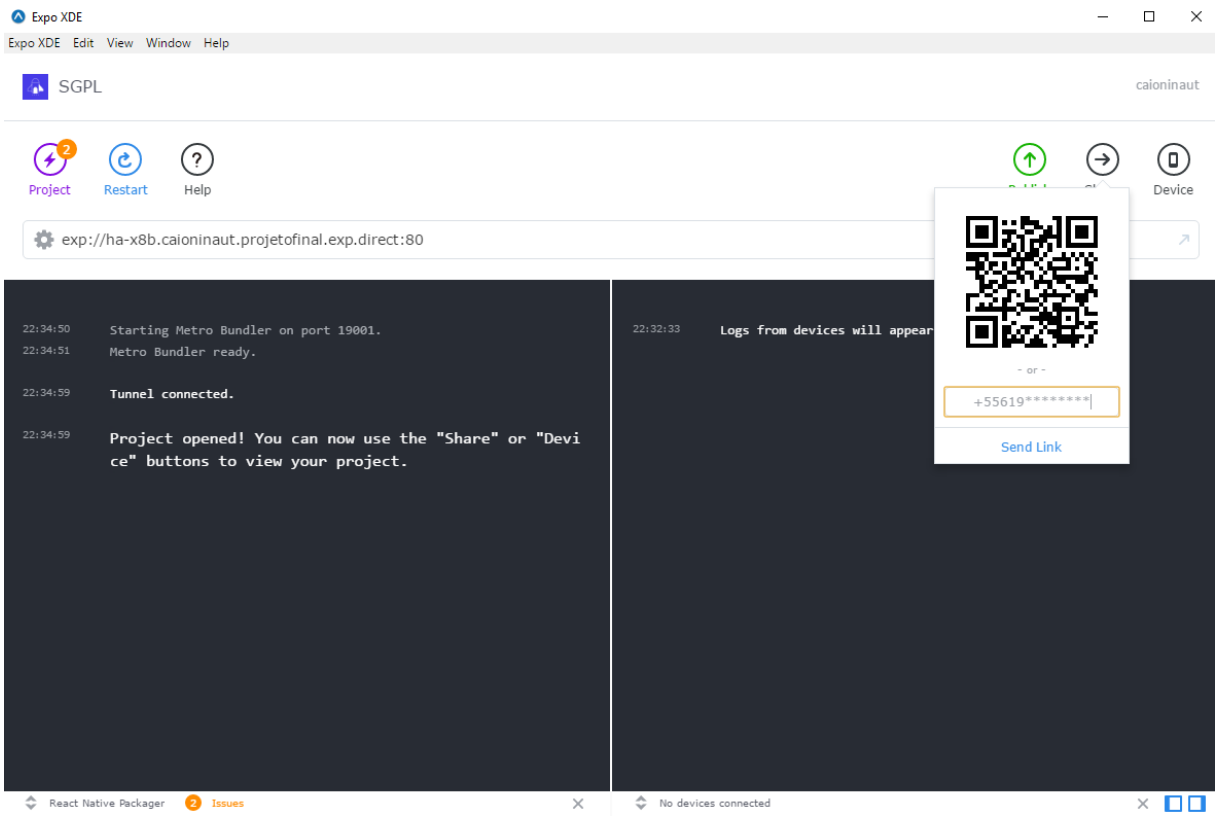


Figura 3.5 Print de Tela do software Expo

- **React Native:** Para instalar o React Native e começar um projeto seguimos alguns passos que se encontram no Apêndice A1. Visto que os outros softwares acima já estão instalados.
- **Firebase:** O Firebase foi escolhido como plataforma para gerenciar parte do *backend* do aplicativo. O banco de dados e a autenticação de login são feitas por ele. Outros recursos também ficam disponíveis na plataforma podendo ou não serem utilizados pelo administrador. Na figura 3.6 é possível ver o código do objeto em JavaScript necessário para adicionar o Firebase ao projeto.

```
<script src="https://www.gstatic.com/firebasejs/5.3.0/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyCW3sXKdSGCoX2zeYhvAdSQjbTT_zhwdKw",
    authDomain: "brutkasten7.firebaseapp.com",
    databaseURL: "https://brutkasten7.firebaseio.com",
    projectId: "brutkasten7",
    storageBucket: "brutkasten7.appspot.com",
    messagingSenderId: "376439044806"
  };
  firebase.initializeApp(config);
</script>
```

Figura 3.6 Print de tela do código para utilização do Firebase

3.1.4 Desenvolvimento do código

O desenvolvimento do código seguiu um fluxo de escolhas e pesquisa que foram necessários para o andamento do projeto. A escolha do React para desenvolvimento do sistema web e do aplicativo foi pela facilidade de desenvolver para diferentes plataformas, tanto como Android quanto para IOS, pois a maioria das ferramentas utilizadas para o desenvolvimento de um código são compartilhadas, com a exceção de algumas pequenas características que devem ser levadas em conta na hora de escolher desenvolver um aplicativo híbrido.

O primeiro passo para se desenvolver o projeto foi definir a estrutura do banco de dados a ser utilizados e como seus dados se relacionam, visto que é através dessa estrutura que nossa aplicação funcionará. Definimos 3 estruturas principais na qual toda aplicação se baseou, lembrando que os dados armazenados são em formato JSON, são elas:

- Users (id, nome, sobre, foto, projetos, atividades)
- Projetos (id, nome, dt_inicio, dt_fim)
- Atividades (users_id, dt_inicio, dt_fim)

A forma de pensar sobre a estrutura é diferente neste projeto, já que o estamos usando o Firebase. E isso nos leva ao próximo passo que é definir a estrutura do sistema web e do aplicativo, considerando que o mesmo banco de dados serve para os dois. Outro fator importante a se levar em consideração é o fato de o sistema web e o aplicativo utilizam diferentes tipos de layout, para o primeiro utilizamos o *ReactStrap* e para o segundo utilizamos o NativeBase. As duas aplicações estão relacionadas devido a chamada do banco de dados que são as mesmas, possuem pequenas diferenças nos formatos de layout.

Uma vez que nosso projeto é principalmente voltado para tarefas do administrador, a parte do CRUD foi implementada apenas para o sistema web, onde apenas o gestor tem acesso, conseguindo com isso realizar tarefas como, a de adicionar Bolsistas, cadastrar Projetos, bem como criar atividades e vincula-las a determinado bolsista. Pelo *app* o bolsista apenas consegue se cadastrar, e visualizar seus dados deixando as tarefas como as de editar perfil, excluir cadastro para o ADM utilizando a parte *web* da aplicação desenvolvida.

No passo seguinte programamos as telas que foram codificadas com pequenas diferenças entre o sistema *web* e o aplicativo. Vale ressaltar que como ambas aplicações compartilham do mesmo banco de dados as mudanças realizadas no sistema web e vice-versa são imediatamente refletidas no outro sistema.

Para o sistema web seguimos o seguinte fluxo de desenvolvimento das telas:

Tela de início: contém os Relatórios (Total de projetos, Total de Usuários e Total de Atividades);

Tela de Bolsistas: lista todos os bolsistas e tem opção de adicionar um bolsista;

Tela de Projetos: cria um projeto e permite o administrador adicionar uma atividade nova ao projeto;

Tela de Detalhe de Projeto: é onde aparece o nome e atividade do bolsista;

Tela de Perfil: contém o nome, e-mail e senha do usuário e Tela de Login.

Já para o App, temos algumas pequenas diferenças, as telas que aparecem no app são:

Tela de Login: permite o acesso ao app

Tela de Sobre: apresenta uma breve descrição sobre o sistema

Tela de Bolsistas: Lista todos os bolsistas cadastrados no sistema

Tela de Perfil: permite que o administrador atualize o perfil dos bolsistas cadastrado.

3.2 Marco do projeto

Com a chegada do projeto, foi definido um marco para cada parte do processo de criação do aplicativo móvel e feita as presciências para elaboração do mesmo. Abaixo é possível observar uma tabela com as datas indicadas previstas para a elaboração do aplicativo, que tiveram duração de um semestre para sua implementação.

TABELA 1: PREVISÃO DE DATAS DO PROJETO

Marco	Previsão
Coleta, analise e definição das necessidades	20/07/2018
Pesquisa das ferramentas de software a serem utilizadas	20/08/2018
Testes das ferramentas	20/09/2018
Desenvolver os códigos de funcionalidade e de *front-end* para aplicação	20/10/2018
Testar a aplicação	25/11/2018
Revisar e aprimorar funcionalidades	26/11/2018
Entrega da primeira versão da aplicação	27/11/2018

3.3 Restrições

A estrutura do projeto foi elaborada com o objetivo de implementar um aplicativo móvel e um sistema web que compartilhem o mesmo banco de dados e que tivessem as mesmas funcionalidades, porém de acordo com o andamento do projeto, não foi possível implementar todas as funcionalidades dentro de ambas as plataformas devido ao nível de conhecimento da equipe e ao prazo.

3.4 Banco de dados

Um sistema de Bancos de Dados nada mais é do que um sistema computadorizado de manutenção de registros. O Banco de Dados, por si só, pode ser considerado como o equivalente, eletrônico de um armário de arquivamento, ou seja, ele é um repositório ou recipiente para uma coleção de arquivos de dados computadorizados. Para entendermos o conceito do banco de dados *FireBase* é importante entender antes o conceito por de trás de um banco de dado *NoSql*. Banco de dados *NoSQL* são banco de dados não relacionais, otimizados para performance escalável e modelos de dados sem esquema. O Banco de dados NoSql são amplamente reconhecidos por sua facilidade de desenvolvimento, baixa latência e resiliência. Esse banco de dados usa diversos modelos de dados, incluindo dados colunares, documentos, gráficos e armazenamento de pares chave-valor na memória. Normalmente são otimizados para

aplicativos que exigem modelos de dados de grande volume de dados, baixa latência e flexibilidade, ao mesmo tempo que são ideais para muitos aplicativos de big data, mobilidade e web, que exigem maior escalabilidade e capacidade de resposta que as oferecidas pelos bancos de dados relacionais tradicionais. O banco de dados NoSQL possuem estruturas de dados mais simples e escalabilidade horizontal, e respondem geralmente com maior velocidade e facilidade com relação ao banco de dados relacionais. Os sistemas de gerenciamento de banco de dados relacionais e os bancos de dados não relacionais (NoSQL) oferecem vantagens e desvantagens diferentes. Nos Bancos de dados relacionais por exemplo as consultas de dados são flexíveis, mas têm um custo relativamente alto e não escalam com facilidade em situações de grande volume de tráfego. Em um banco de dados NoSQL, há formas limitadas de consultar dados com eficiência. As demais formas de consulta podem apresentar alto custo e baixa performance. Passamos a baixo a listar uma tabela de diferenças entre os bancos de dados relacionais e não relacionais a fim de conceituarmos principais diferenças vantagens e desvantagens.

Modelos comparativos	Banco de Dados NoSQL	Banco de dados relacionais
Modelos de dados	Não aplicam um esquema. Geralmente, uma chave de partição é usada para recuperar valores, conjuntos de colunas ou documentos semiestruturados JSON, XML.	O modelo relacional normaliza dados em tabelas, compostas por linhas e colunas. Um esquema define estritamente tabelas, colunas, índices, e relações entre tabelas
Ferramentas	Os bancos de dados NoSQL normalmente oferecem ferramentas para gerenciar clusters e escalabilidade. As aplicações são a interface principal com os dados subjacentes.	Os bancos de dados SQL normalmente oferecem um rico conjunto de ferramentas para simplificar o desenvolvimento de aplicações orientadas ao banco de dados.
Performance	Desempenho geralmente é uma função do tamanho do cluster do hardware subjacente, da latência de rede e da aplicação que faz a chamada.	O desempenho normalmente depende do subsistema do disco. A otimização de consultas, índices e estrutura de tabela é necessária para alcançar máximo desempenho.
Escala	Projetado para escalabilidade horizontal usando clusters distribuídos de hardware de baixo custo para aumentar o throughput sem aumentar a latência.	Escalabilidade vertical mais fácil com hardware mais rápido. Outros investimentos são necessários para que tabelas relacionais abranjam um sistema distribuído.
APIS	As APIs baseadas em objetos permitem que desenvolvedores de aplicativos armazenem e restaurem facilmente estruturas de dados na memória..	As solicitações para armazenar e recuperar dados são comunicadas usando consultas compatíveis com uma Structured Query Language (SQL – Linguagem de consultas estruturadas).

TABELA 2 – TABELA COMPARATIVA BANCO DE DADOS NÃO RELACIONAIS X RELACIONAIS

O banco de dados precisará ser conectado a um servidor fazendo uma atualização diária do cadastro dos bolsistas novos, inserindo no banco de dados, ou apagando bolsistas excluídos do sistema. E este link com as nuvens pode ser verificado como um backup, tornando o serviço mais seguro e confiável contra eventuais problemas que possam vim a acontecer e comprometer os dados do banco de dados.

Diante dessas informações podemos falar sobre como foi pensado e estruturado o banco de dados utilizado neste projeto, dado que utilizamos o Firebase, que utiliza um sistema NoSQL. Como já citado foram criadas 3 estruturas principais, onde os dados armazenados no formato JSON, são enviados para o servidor de banco de dados do Firebase que os interpreta e armazena-os, e como não há uma relação direta no servidor do Firebase, esses dados são relacionados após um planejamento de estrutura. Abaixo podemos ver um exemplo de como é uma das estruturas que utilizamos. A tabela "Users" nela é possível ver os campos que mostram na tela de bolsistas e lista os projetos e atividades que ele faz parte, que serão mostrados no capitulo 4.

Figura 3.7 Código JSON da tabela 'users'

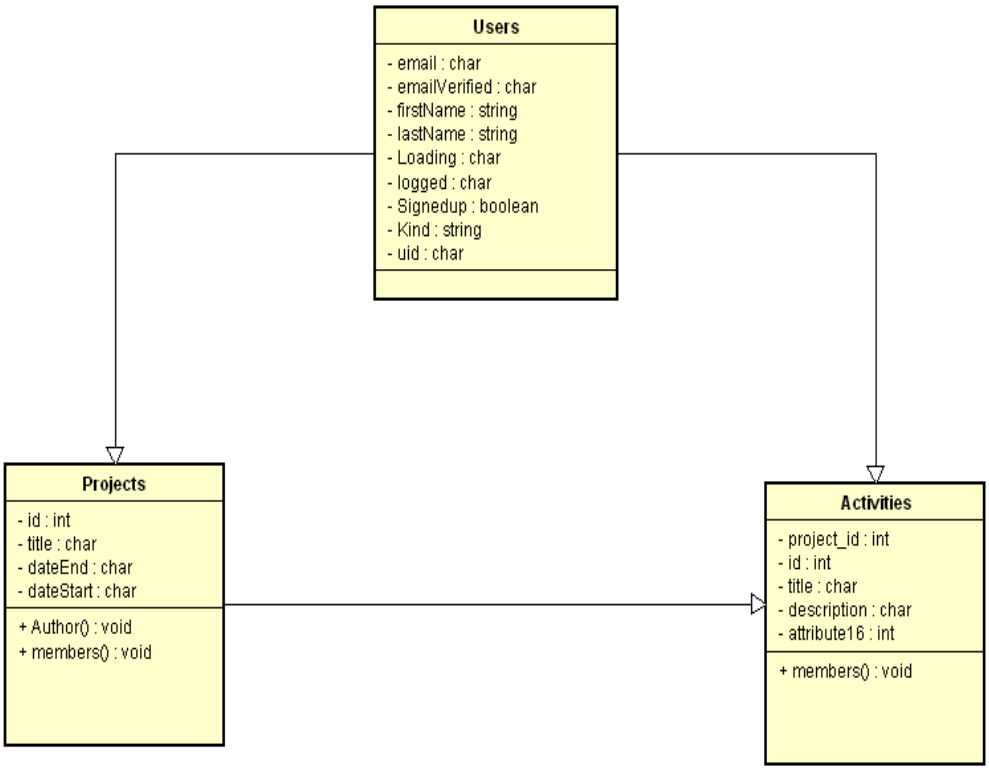


Figura 3.8 Modelo de Entidade e Relacionamento

Através da figura 3.8 vemos como é a estrutura utilizada no sistema e como o banco de dados que não é relacional, ou seja, o relacionamento das tabelas não é direto, através do código de back-end que relacionamos a configuração dos dados. Como já dito, o sistema se baseia em 3 tabelas principais que contém todos os campos necessários no sistema na configuração atual. A configuração do banco é mostrada acima.

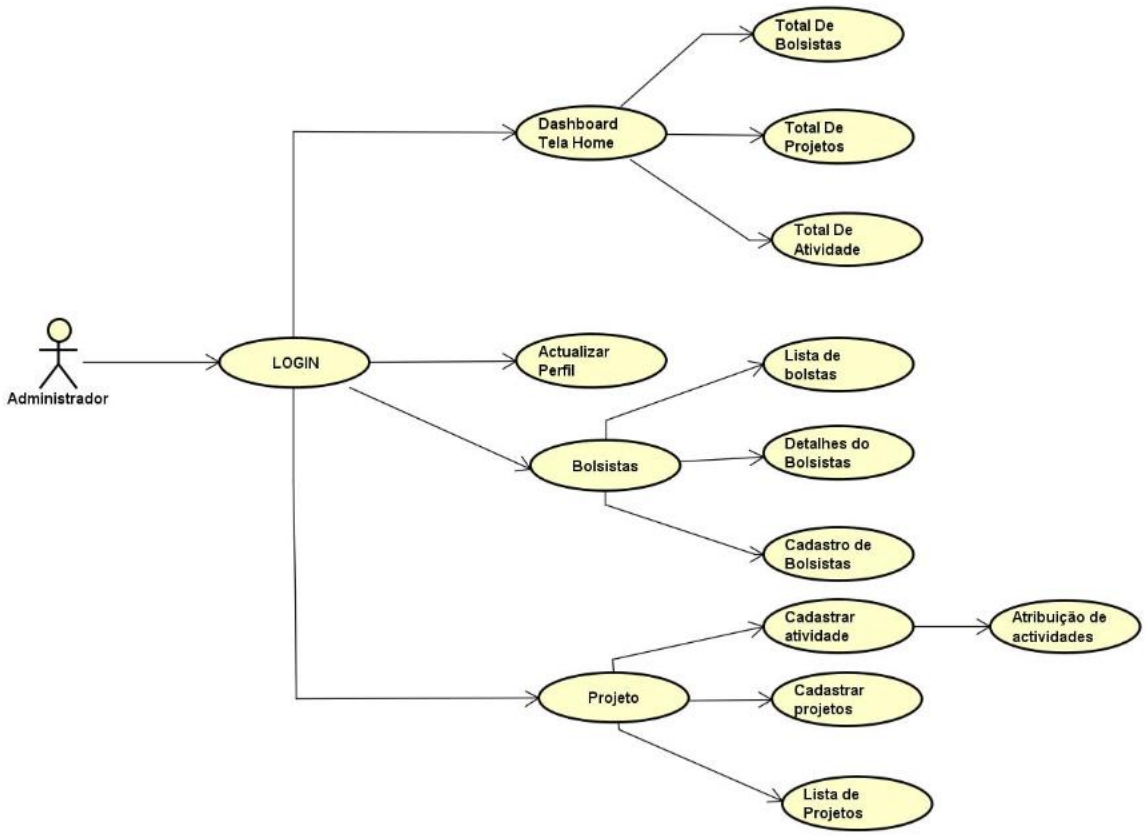


Figura 3.9 Diagrama De Caso De Uso Sistema Web Para Usuário/Administrador

Como visto ao longo do texto, o ponto central do nosso projeto é o Gerente, que na figura 3.9 é descrito como administrador, é ele o responsável pela manutenção e manuseamento do sistema web, na figura 3.9, ilustramos um diagrama de caso de uso do sistema web, nele estão listados o conjunto de Atividade possíveis de serem realizados pelo administrador do sistema. Cada elipse corresponde há uma tela do sistema e em seguida passamos a explicar detalhadamente, o que cada uma dela faz.

Ao passar pela tela de Login, o Administrador é encaminhado para uma tela onde contem quatro opções:

- Dashboard(Tela de Login): Essa tela de cara apresenta, três opções , nomeadamente, o número total de bolsistas cadastrados no sistema, o número total de projetos cadastrados no sistema, e o úmero total de atividades cadastradas no sistema.
- Atualizar perfil: passando pelo login o Administrador, tem a opção de atualizar seus dados cadastrais, como e-mail, senha, nome data de nascimento.
- Bolsista: passando pela tela de login, o administrador tem a opção de consultar o menu bolsista. É através desse menu, que o administrador consegue ter acesso a lista com todos os bolsistas, cadastrados no sistema web, assim como, adicionar novos bolsistas. A partir do menu bolsista o administrador pode também, cadastrar novos bolsistas, verificar detalhes de cada bolsista como, em qual projeto está vinculado, assim como qual atividade lhe foi atribuída.
- Projetos: Acessando o menu do sistema na opção projeto, o administrador tem acesso a três funcionalidades, nomeadamente, pode cadastrar os projetos, pode cadastrar as

atividades, e listar todas os projetos cadastrado. É dentro do menu cadastrar atividade, no menu projeto que determinada atividade é atribuída a determinado Bolsista.

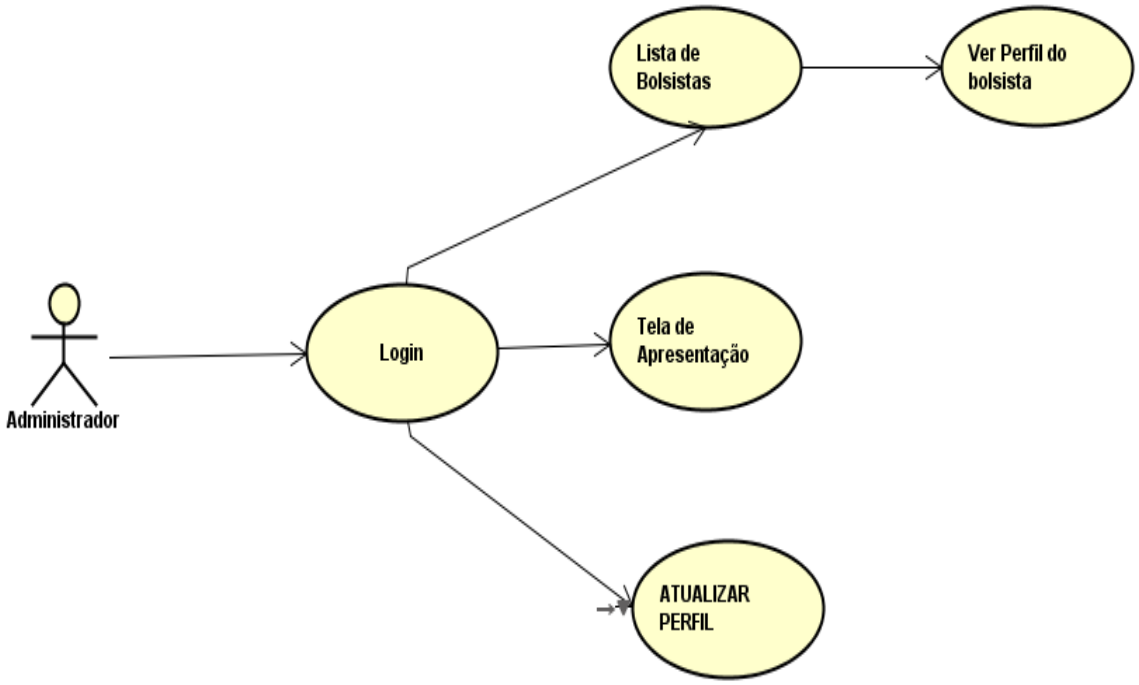


Figura 3.10 Diagrama De Caso De Uso Apps Para Usuario Administrador.

Na figura 3.10, ilustramos o diagrama caso de uso, executado pelo administrador, olhando através do aplicativo móvel. Como observamos, pelo aplicativo não é possível executar o mesmo número de funcionalidades como é possível no sistema web. Com isso uma vez efetuado o login, o administrador tem acesso as seguintes funcionalidades:

- Tela de apresentação: essa tela, mostra uma pequena ilustração teórica, do funcionamento do sistema, ou qualquer outra informação que o administrador opte por colocar.
- Lista de bolsistas: Essa tela mostra uma lista de todos os bolsistas cadastrados, através dessa tela é possível acessar a opção ver perfil, onde o administrador pode acessar os dados de todos os bolsistas cadastrados, podendo atualizar ou não.
- Atualizar perfil: essa tela, oferece ao administrador a opção de atualização dos seus dados cadastrais, como nome, senha ou e-mail.

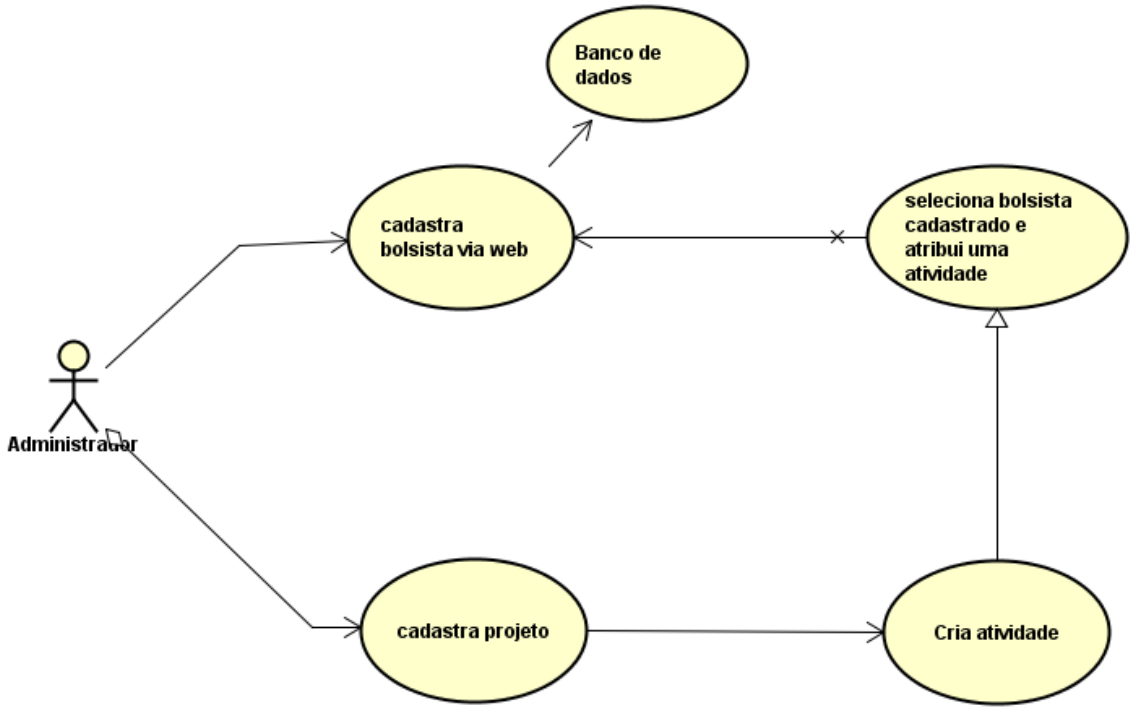


Figura 3.11 Diagrama de caso de uso para a atribuição de bolsista a uma atividade

Na figura 3.11 ilustramos um dos muitos casos de uso possível no sistema. Nele temos o caso em que o administrador vincula ao bolsista uma determinada atividade, para tal o primeiro passo, é o cadastro do bolsista pelo administrador no sistema, em segundo o Administrador cadastra um projeto, e cria uma determinada atividade dentro do projeto, posteriormente basta selecionar o bolsista e atribui-lo atividade pretendida.

4 RESULTADOS

Ao final do projeto passamos a listar cada tela do sistema web e do aplicativo e suas respetivas funcionalidades:

Sistema WEB:

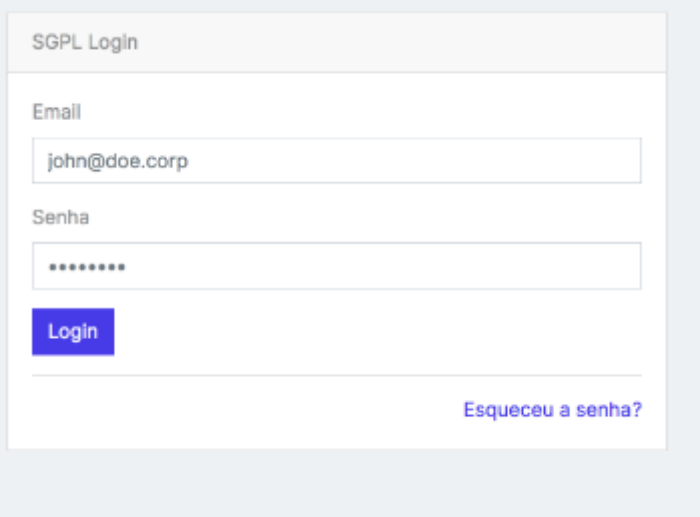


Figura 4.1 Tela De Login Do Sistema Web

Através da figura 4.1temos tela de login, através da qual é possível que o Administrador/Gestor tenha acesso ao sistema.

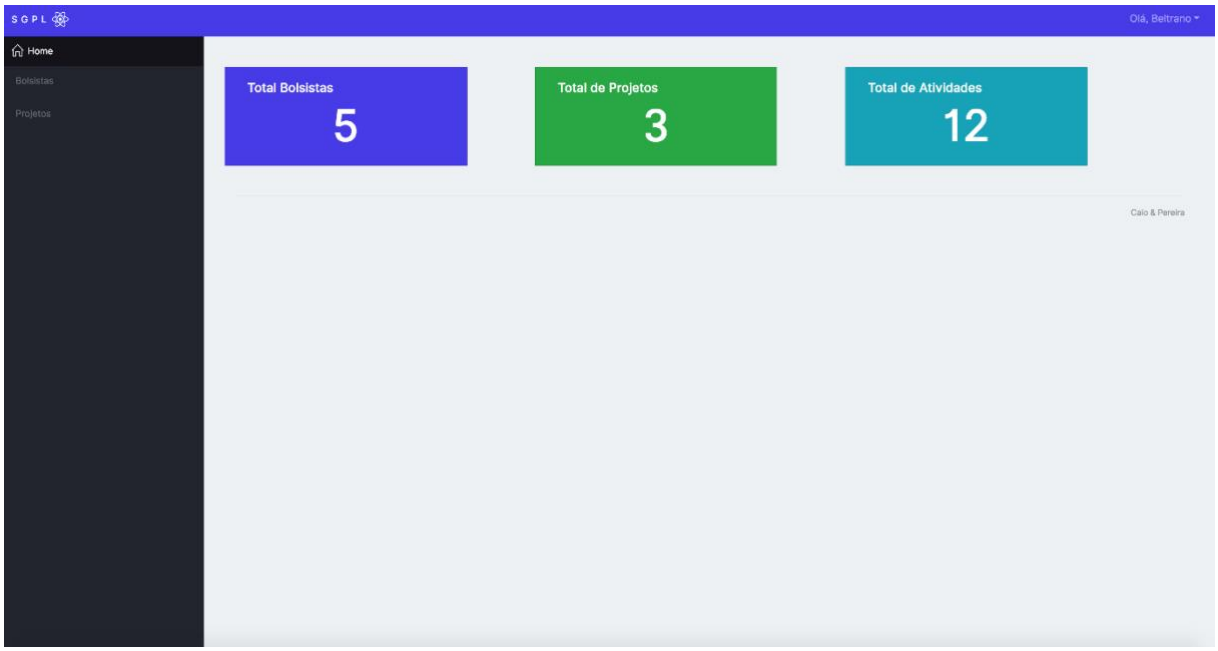


Figura 4.2 Dashboard Tela Home

A segunda tela, ou seja, a primeira tela depois que o Administrador logra no Sistema via web é a tela home, mostrada na figura 4.2, ela mostra o número total de bolsista cadastrados, o número total de projetos em curso assim como o número total de atividades.

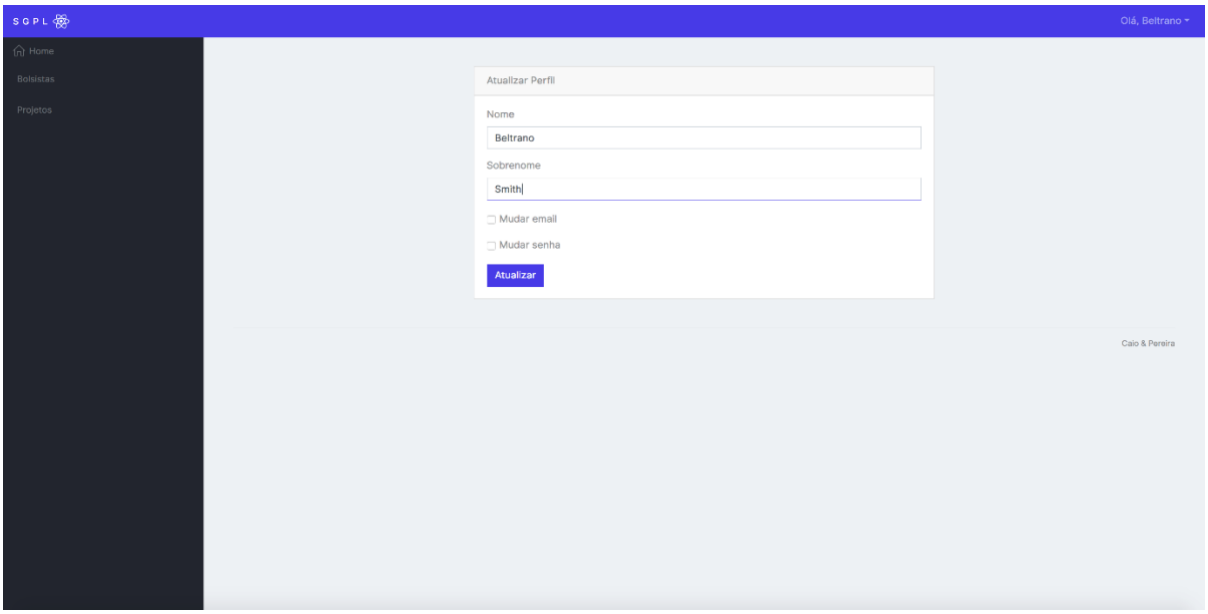


Figura 4.3 tela para atualização de cadastro via sistema web.

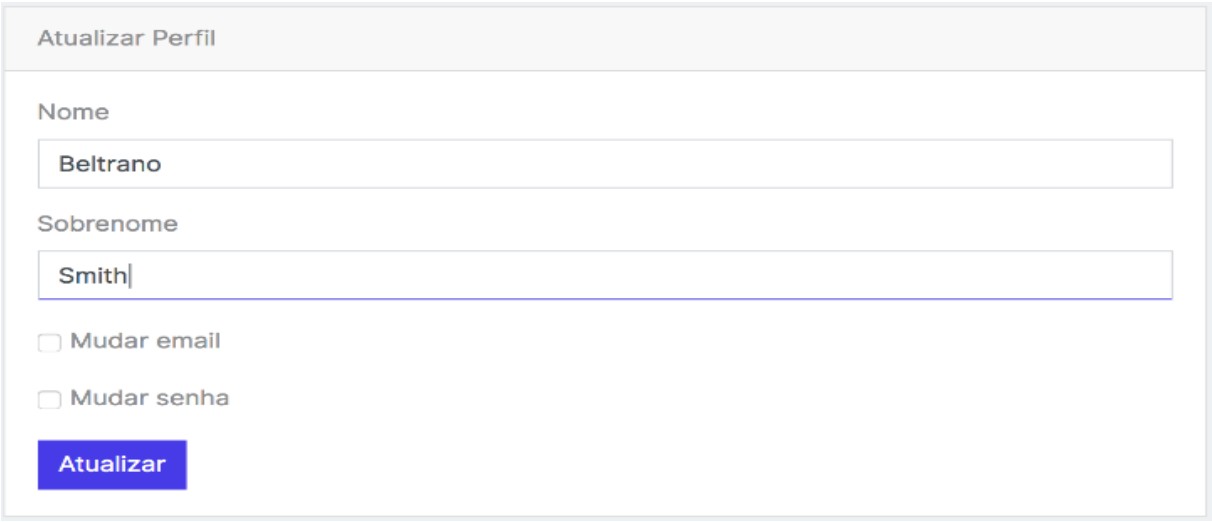


figura 4.4 tela de atualização de cadastro via web.

A figura 4.4 é uma ampliação para mostrar os campos da figura 4.3. É através dessas telas que o administrador consegue atualizar dados cadastrais, mudar e-mail ou redefinir sua senha.

Figura 4.5 Tela de menu do sistema web

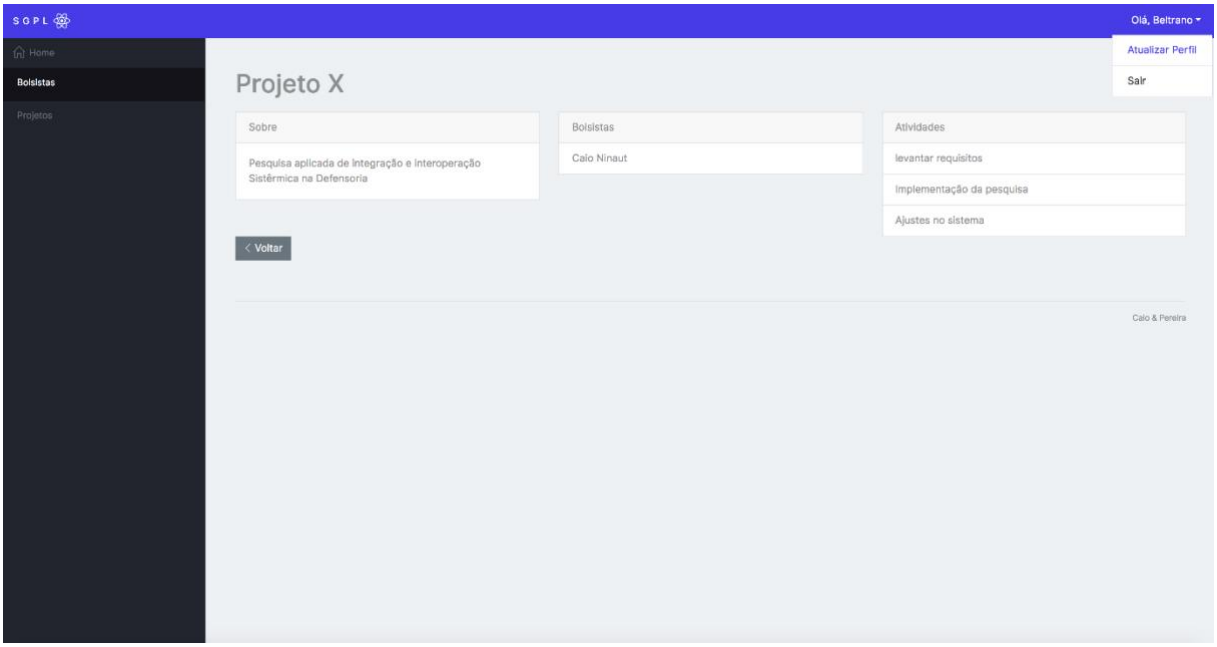


Figura 4.6 Tela De Menu Do Sistema Via Web

A figura 4.5 é o menu do sistema ela oferece as opções de sair do sistema e leva também a tela de atualização de perfil do usuário logado, momentaneamente (Administrador) no sistema.

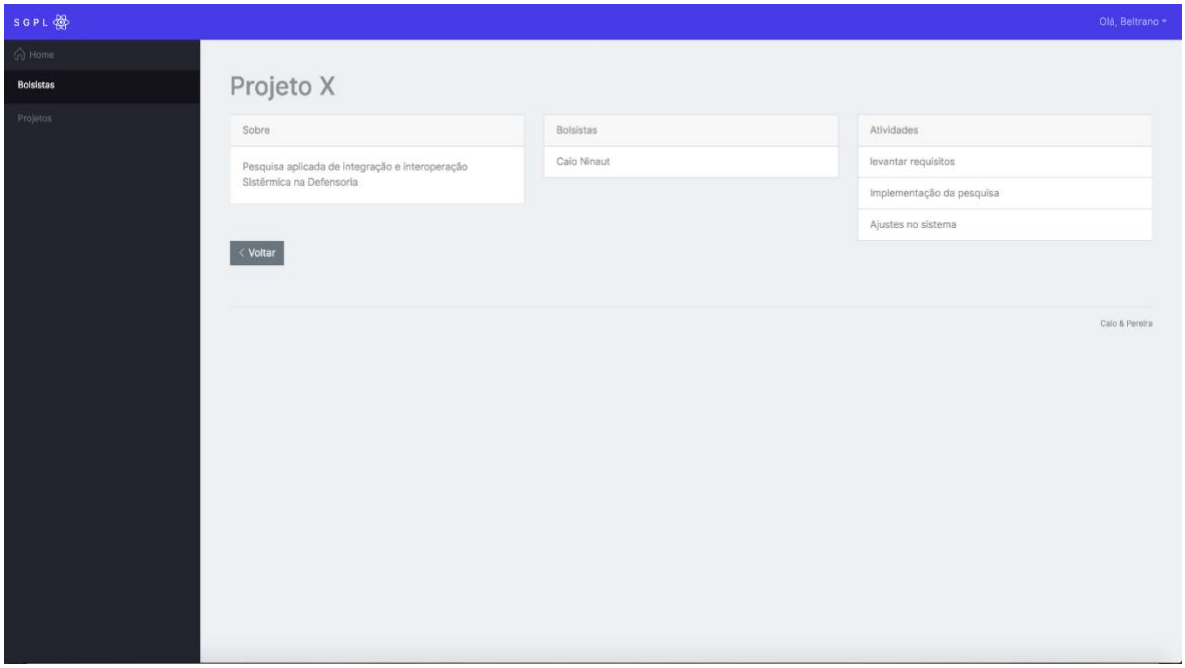


Figura 4.7 Tela de detalhes do projeto

No ato de cadastro de cada bolsista é atribuído a ele um determinado projeto. A tela acima além de detalhar o projeto no campo sobre, mostra quais bolsistas estão cadastrados nesse projeto, no campo bolsista através do nome, e mostra no campo atividades as atividades vinculadas a esse bolsista dentro do projeto em que está inserido.

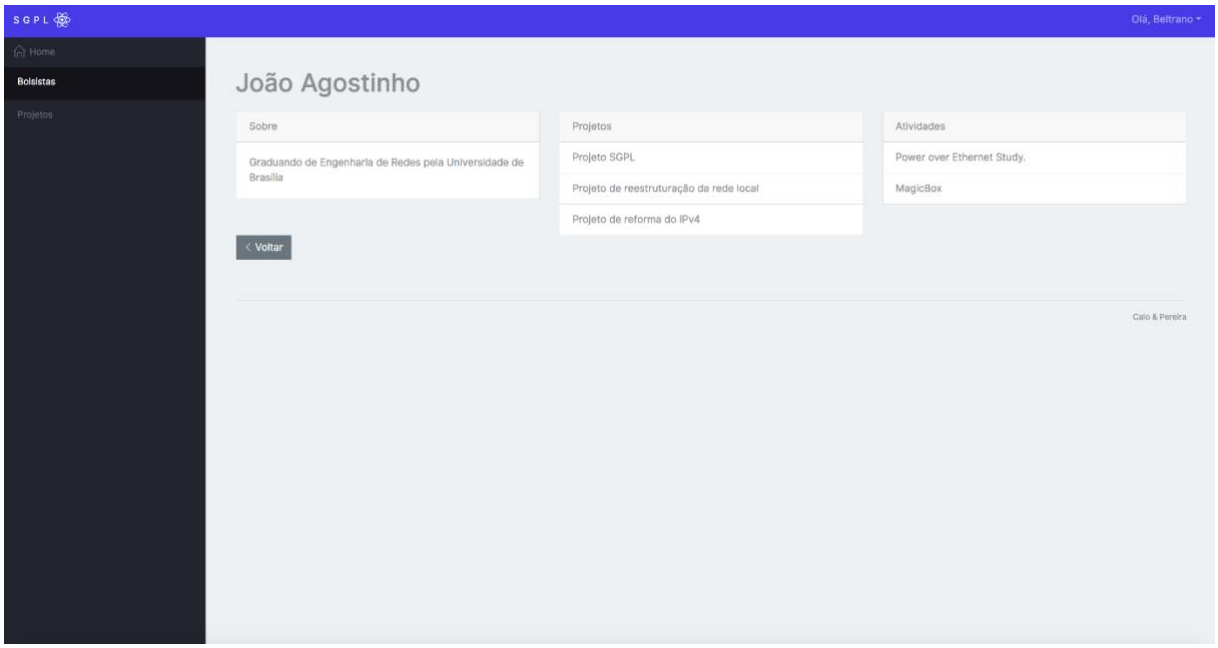


Figura 4.8 Tela De Detalhes Do Bolsista

A tela da figura 4.8 mostra detalhes do bolsista, ou seja, permite que se acessada, o administrador saiba mais sobre o bolsista no campo sobre, sobre quais projetos está inserido no campo projetos e quais atividade dentro do projeto está vinculado a ele especificamente.

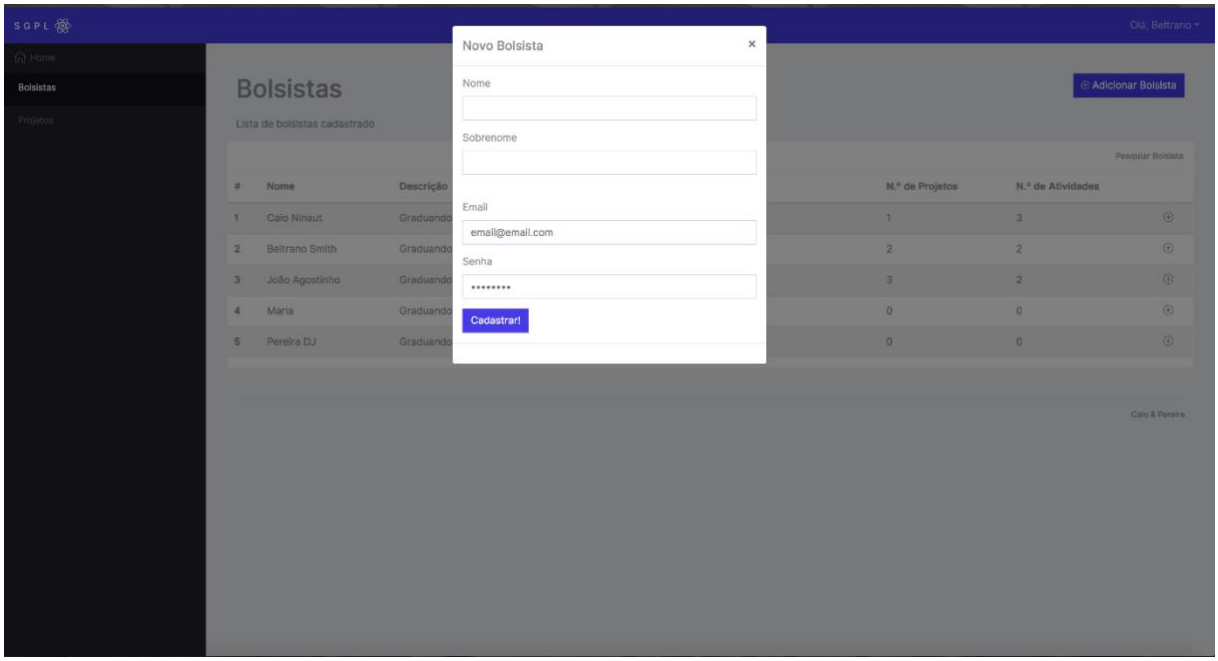


Figura 4.9 Tela De Cadastros De Bolsistas Via Web

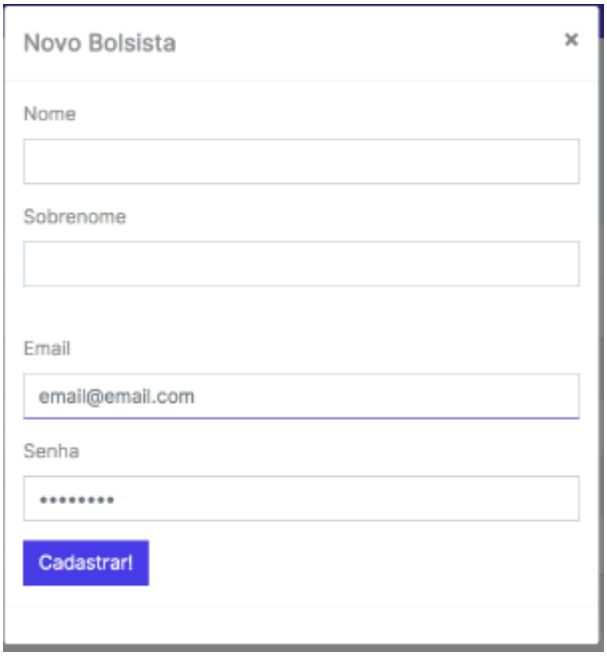


Figura 4.10 Tela De Campos De Cadastros De Bolsistas

A figura 4.10 é uma ampliação da tela mostrada na figura 4.9, é através dessas telas que o administrador cadastra determinado bolsista novo. Na figura 4.11, apresenta uma lista com todos bolsistas cadastrados no sistema, pelo nome através do campo nome, apresenta uma breve descrição sobre o bolsista em questão, através do campo descrição, apresenta também o número de projeto, com o qual cada bolsista está vinculado, e dentro dos projetos quantas atividades lhe foi atribuído.

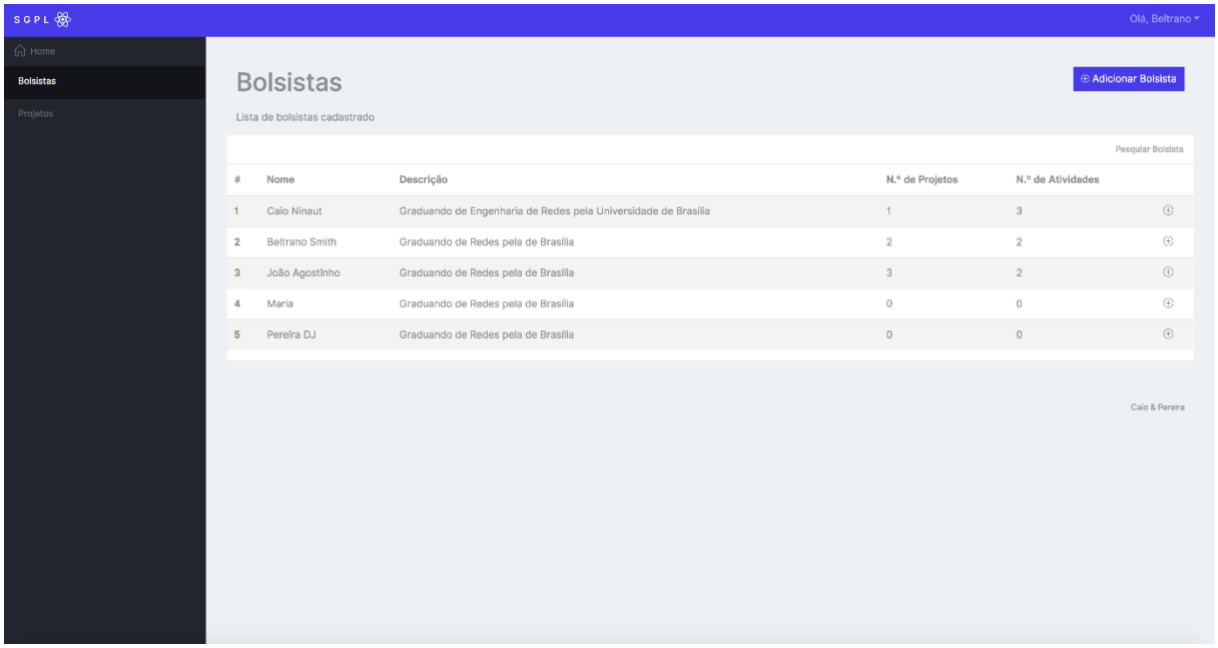


figura 4.11 tela de lista de bolsistas cadastrados

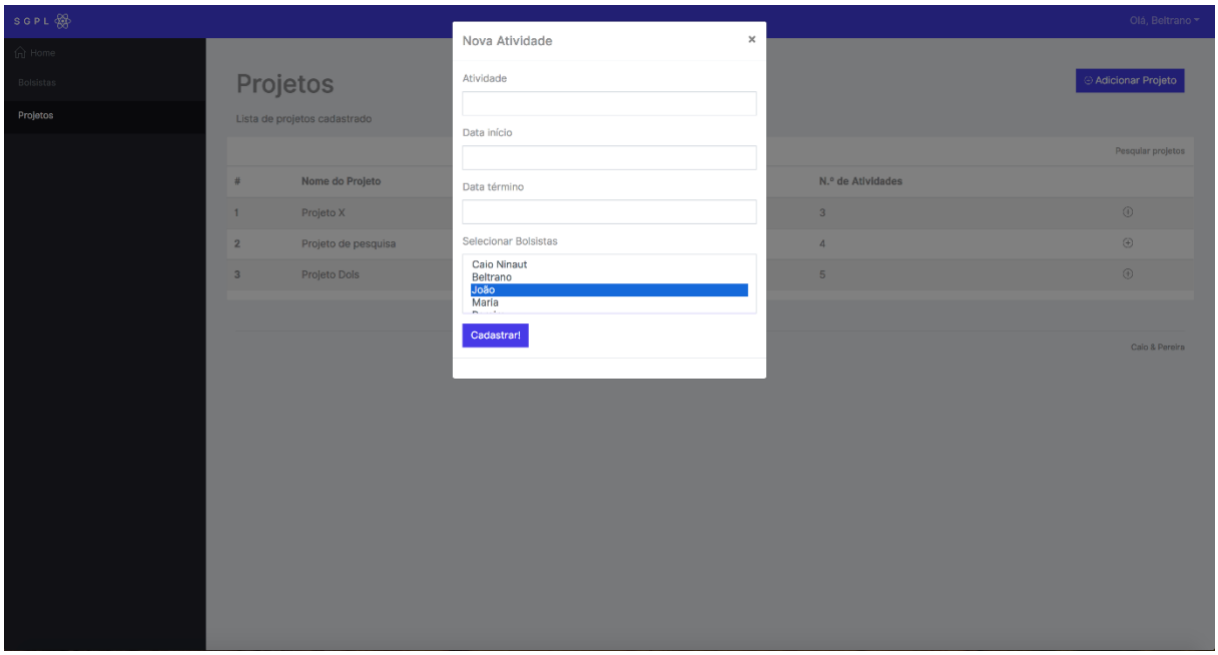


Figura 4.12 Tela De Atribuição De Atividades

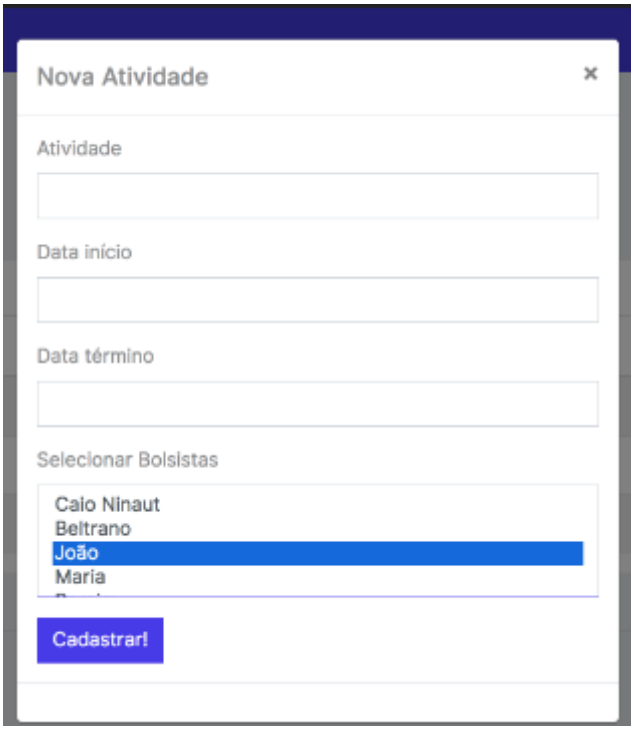


Figura 4.13 Tela De Campos De Atribuição De Atividades

As figuras 4.12 e 4.13 ilustram as telas de atribuição de atividade. É através dessas telas que o administrador vincula determinado bolsista a determinada atividade através do campo selecionar bolsista. O administrador determina uma data de início da atividade e uma data para o termino da atividade, através dos campos data início e data termino, e a descrição da atividade é registrada através do campo atividade.

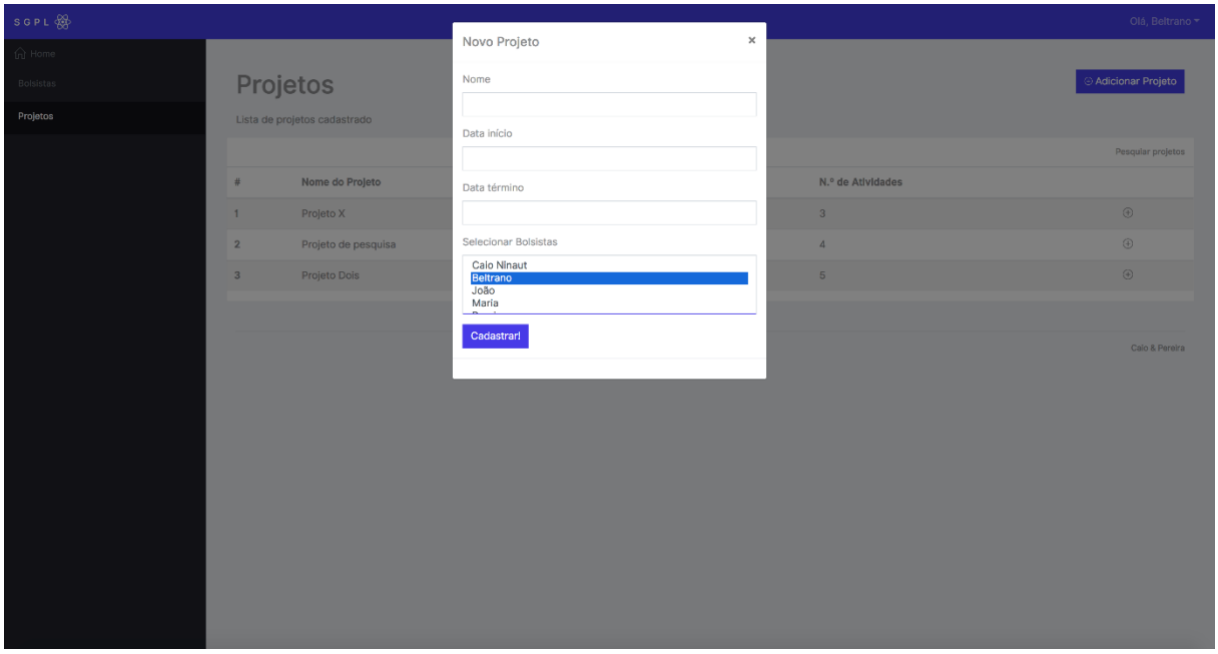


Figura 4.14 Tela De Atribuição De Projetos A Bolsistas

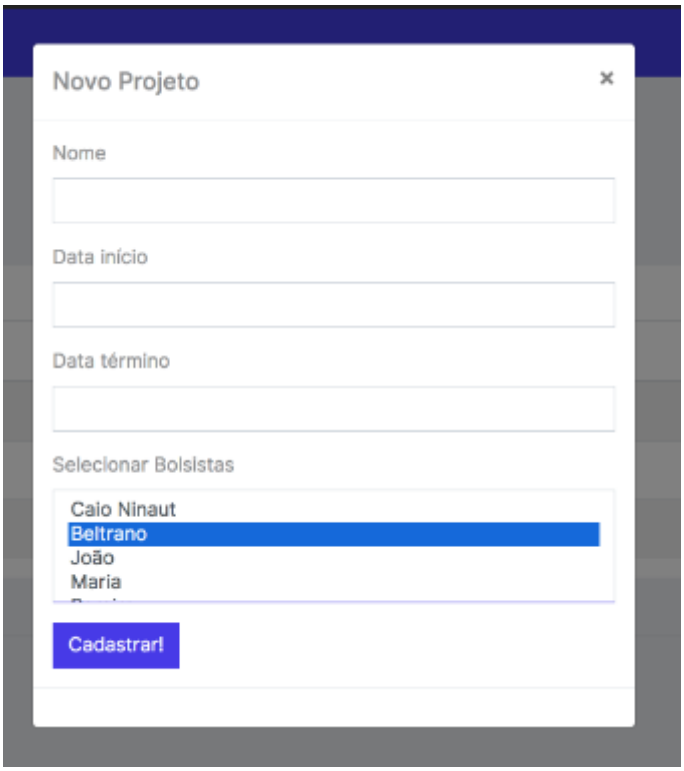


Figura 4.15 Tela De Campos De Atribuição De Projetos A Bolsistas

As figuras 4.14 e 4.15 ilustram as telas de atribuição de Projetos. É através dessas telas que o administrador atribui determinado projeto a determinado bolsista através do campo selecionar bolsista. O administrador determina uma data de início para o projeto e uma data para o termino do projeto, através dos campos data início e data termino, e a descrição (tópico) do projeto em que o bolsista será vinculado é registrada através do campo nome.

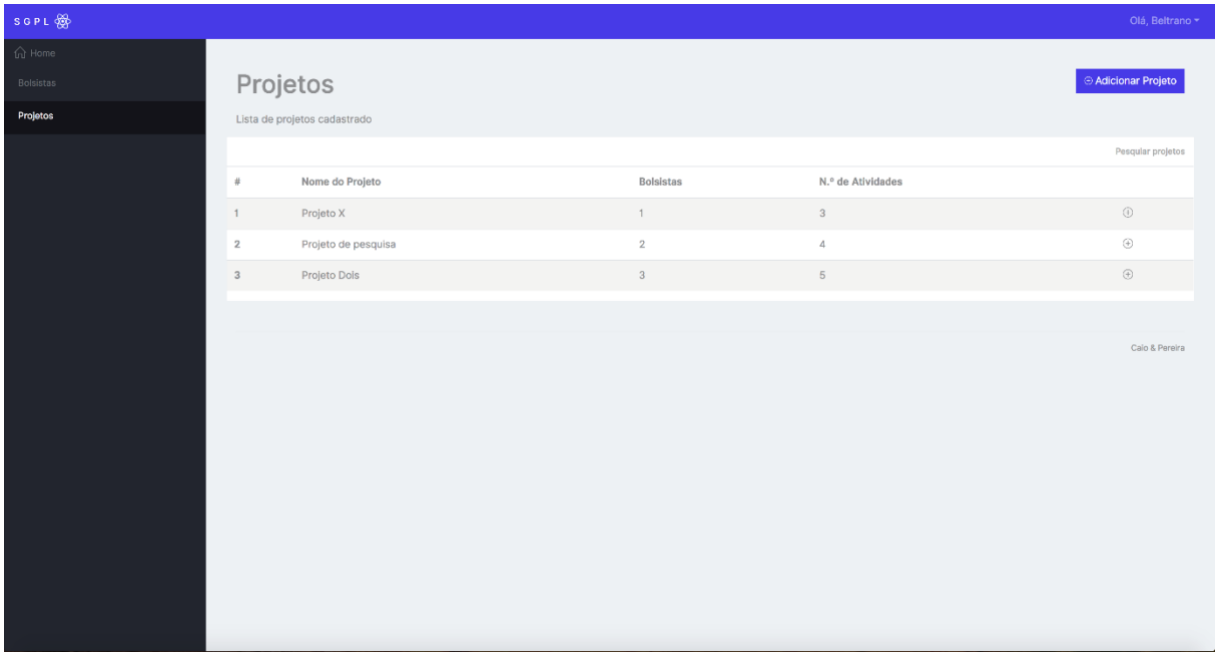


Figura 4.16 Tela De Lista De Todos Projetos Cadastrados.

A tela acima lista todos projetos cadastrados no sistema pelo tema, através do campo nome do projeto. Uma vez que em um mesmo projeto podem estar atribuídos, mas do que um bolsista, é possível ver o número de bolsista atribuído a cada projeto através do campo Bolsistas, e finalmente é possível ver dentro dos projetos qual o número de atividade atribuída a cada bolsista através do campo número de atividade.

4.1 TELAS APP:

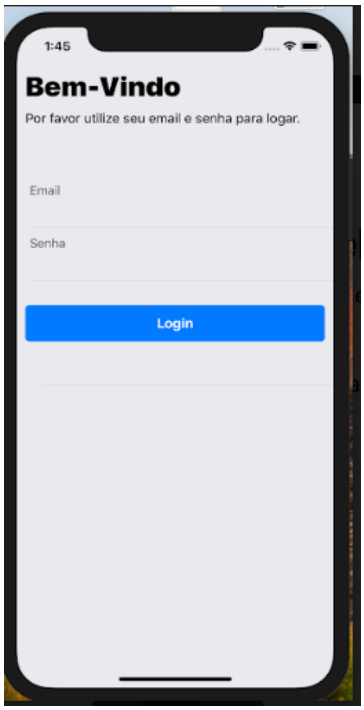


Figura 4.17 Tela De Login Do App

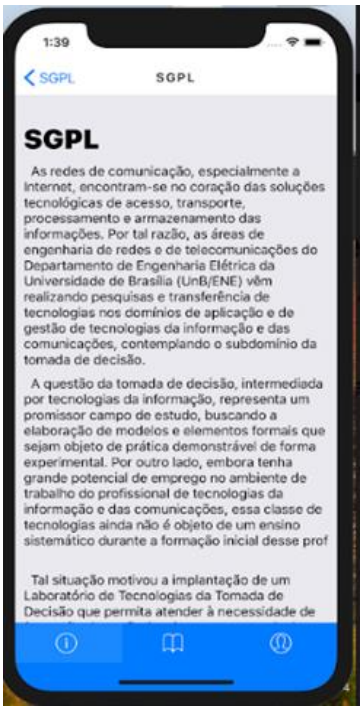


Figura 4.18 Tela De Apresentação Do App

A figura4.17, ilustra a tela de login do aplicativo móvel. É através dessa tela que o Administrador acessa o aplicativo, o mesmo acontece com o bolsista sendo que sua senha de acesso é previamente criada e dada pelo administrador do sistema. A figura 4.18 logo ao lado ilustra a tela de apresentação do aplicativo, contém um texto que pode ser alterado de acordo com aquilo que administração do sistema desejar. Tela de apresentação onde é exibido o conteúdo sobre o aplicativo e sobre o laboratório que será introduzido o sistema e o aplicativo para gestão de projetos do LATITUDE.

A figura 4.19 ilustra a tela de bolsistas cadastrados no sistema. Através dela o Administrador tem acesso aos dados de todos bolsistas cadastrados, e através do campo ver perfil tem acesso ao perfil do bolsista podendo com isso editar o perfil dos bolsistas. É importante frisar que os Bolsistas que aparecem, nessa tela são os mesmos bolsistas que são cadastrados via sistema web, e qualquer alteração no sistema web é prontamente refletida no app mobile.

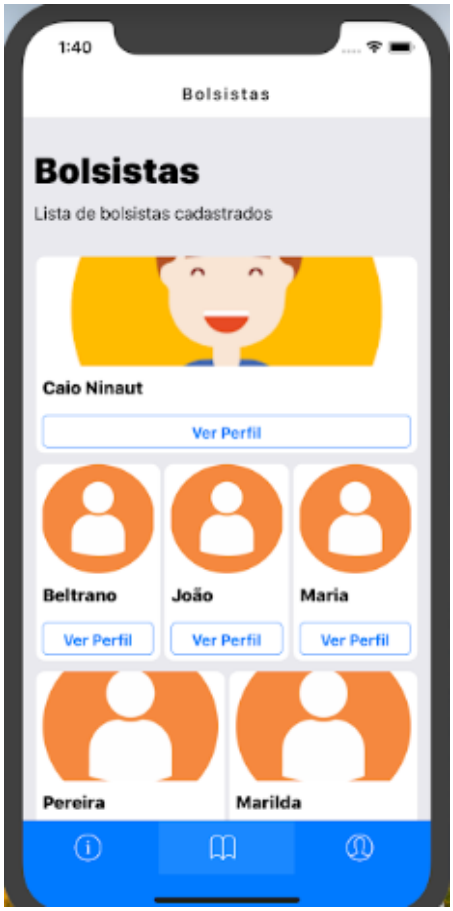


Figura 4.19 Tela de Bolsistas cadastrados

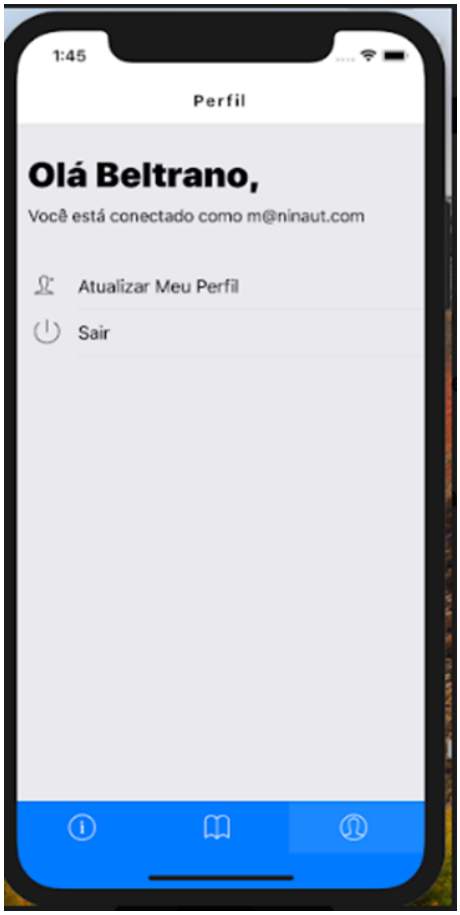


Figura 4.20 Tela De Perfil Via App

A figura 4.20, ilustra a tela de perfil de determinado bolsista. Através dela o administrador tem a opção de sair do aplicativo, ou então ir para tela de atualização de perfil de bolsista, ilustrada na figura 4.21, onde o administrador poderia editar o perfil de determinado bolsista, e redefinir e-mail ou senha.

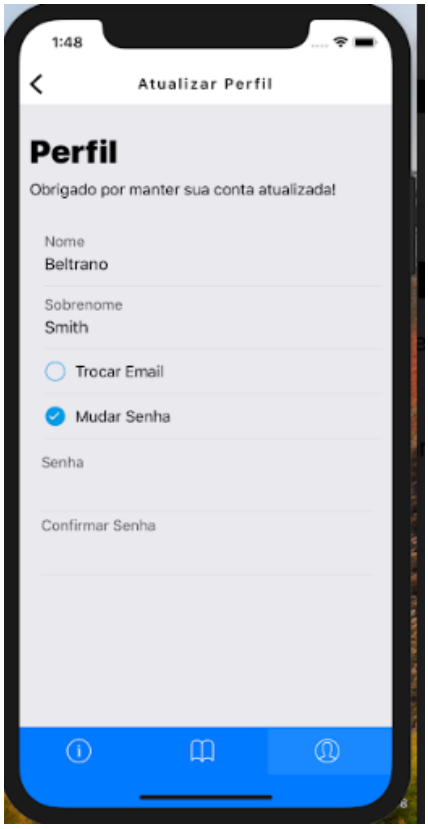


Figura 4.21 Tela Que Permite Atualizar Perfil Via App

5 CONCLUSÃO E TRABALHOS FUTUROS

O objetivo geral do presente trabalho era o de desenvolver uma arquitetura de sistema web/aplicativo mobile, que auxiliasse o (s) gerente (s) de projetos do Latitude, nas tarefas de planejar, monitorar e controlar estudantes, bem como os projetos vinculados a eles a qualquer hora e de qualquer lugar. No final do projeto, já temos o sistema funcionando como é possível ver ao longo do texto, porém é um sistema em desenvolvimento passível de ajustes como falaremos no próximo parágrafo. Apesar disso arquitetura do sistema, está montada, e questões como cadastro de bolsistas, cadastro de projetos, inclusão de atividades e atribuição da mesma aos estudantes cadastrados já podem ser feitas. Um ponto que é importante destacar, é que essa arquitetura foi projetada, para atender a demanda do departamento de gerenciamento de projetos do Laboratório de Tecnologias da Tomada de Decisão gerenciamento de projetos, podendo porem ser expandido, para mais áreas que atuam na mesma esfera de gerencia.

Atualmente, toda tarefa da plataforma é realizada através do sistema web, sendo o aplicativo usado apena para, fins de consulta, o que nos remete as questões que podem ser implementadas futuramente:

- A ideia é que o aplicativo móvel, também realize todas funções, que o sistema web realiza.
- Expandir aplicação móvel, para que os bolsistas também tenham acesso, a ela, para, tarefas como por exemplo consultar seus históricos, eventuais atividades que estejam inseridos, consultem suas declarações, termos e planos de pesquisas, consigam fazer seus próprios cadastros, ou seja, fazer com que aplicativo, não se limite apenas na tarefa de auxílio ao gestor, mas também no auxílio aos próprios estudantes bolsistas.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] NodeJS. Disponível em: www.nodejs.org. Acesso em: 24/07/2018

[2] Expo. Disponível em https://expo.io/learn. Acesso em: 24/07/2018

[3] Documentação sobre o Expo. Disponível em: https://docs.expo.io. Acesso em: 24/07/2018

[4]Firebase. Disponível em: https://firebase.google.com/docs/database/web/structure-data?hl=pt-br. Acesso em: 24/07/2018

[5] NativeBase. Disponível em: https://docs.nativebase.io/. Acesso em: 24/07/2018

[6] Sobre NoSQL. Disponível em https://aws.amazon.com/pt/nosql/. Acesso em: 24/07/2018

[7]Visual Studio Code. Disponível em: https://code.visualstudio.com/. Acesso em: 24/07/2018

[8] Começando com React. Disponível em: https://reactjs.org/docs/getting-started.html. Acesso em: 24/07/2018

[9] Começando com React Native. Disponível em: https://facebook.github.io/react-native/docs/getting-started.html. Acesso em: 24/07/2018

[10] Introdução ao sistema de BANCOS DE DADOS C.J date

[11] TANENBAUM, A. S. – Redes de Computadores – 4ª Ed., Editora Campus (Elsevier), 2003

[12] KUROSE,Jim, Redes de Computadores e a Internet - Uma Abordagem Top-Down - 3ª Ed, Editora Pearson, 2006.

[13] Alberto Manuel,Carlos Alberto Escaleira UML, Metodologias e Ferramentas CASE, editora Centro Atlântico Portugal/2001

[14] Candido, Roberto. Gerenciamento de projetos, Curitiba: Aymará, 2012.

[15] Zelia, Maria.Gestão de pessoas uma vantagem competitiva, FGV 2016

[16] ZHAO, W. Building Highly dependable Wireless Web Services. Journal of Electronic Commerce in Organizations, p. 1-16, 10 a 12 2010.

[17] TURTSCHI, A. et al. Chapter 11 - Web Services. C#.NET Web Developer's Guide, Burlington, p. 575-668, 2002.

[18] FILHO, O. F. F. Serviços Semânticos: Uma abordagem RESTful. Escola Politécnica da Universidade de São Paulo. São Paulo, p. 103. 2009.

[19] https://firebase.google.com/docs/?hl=pt-br

[20] Guia da PMBOK® 5 EDIÇÃO5ª edição.2012.

7 APÊNDICE

7.1.1 Instalação do React Native e Criação de um projeto

A instalação do React Native se deu pelo comando:

```
npm install –g create-react-native-app
```

A criação do projeto se deu pelo comando abaixo, onde “sgplapp” é o nome do projeto criado.

```
create-react-native-app sgplapp
```

Assim as instalações foram feitas em sua maioria, após isso iniciou-se o desenvolvimento do projeto.

7.1.2 Código fonte do programa

```
export default {
  projects: [
    {
        id: 1,
        title: '---- ------ ',
        author:[{
            email: '---- ------ ',
            emailVerified: '---- ------ ',
            firstName: '---- ------ ',
            lastName: '---- ------ ',
            loading: '---- ------ ',
            logged: '---- ------ ',
            signedUp: '---- ------ ',
            uid: "KVpcOYOAQpeUZ0FfnJKJ0QhqnF92",
            }],
        dateEnd: '---- ------ ',
        dateStart: '---- ------ ',
        members: [{
                uid: 'user Id',
                uid: 'user Id',
                uid: 'user Id',
            }]
        },
  ],
  activities: [
    {
      project_id: 1,
      id: 0,
      title: '---- --- -- ------',
      description: '---- --- -- ------ ---- --- -- ------ ---- --- -- ------ ---- --- -- ------',
      members: [{
          uid: 'user Id',
          uid: 'user Id',
          uid: 'user Id',
      }],
    }
],
  users: [
      {
      email: '---- ------ ',
      emailVerified: '---- ------ ',
      firstName: '---- ------ ',
      lastName: '---- ------ ',
      loading: '---- ------ ',
      logged: '---- ------ ',
      signedUp: '---- ------ ',
      kind: 'admin',
      uid: "KVpcOYOAQpeUZ0FfnJKJ0QhqnF92",
    }
    ],
};
```

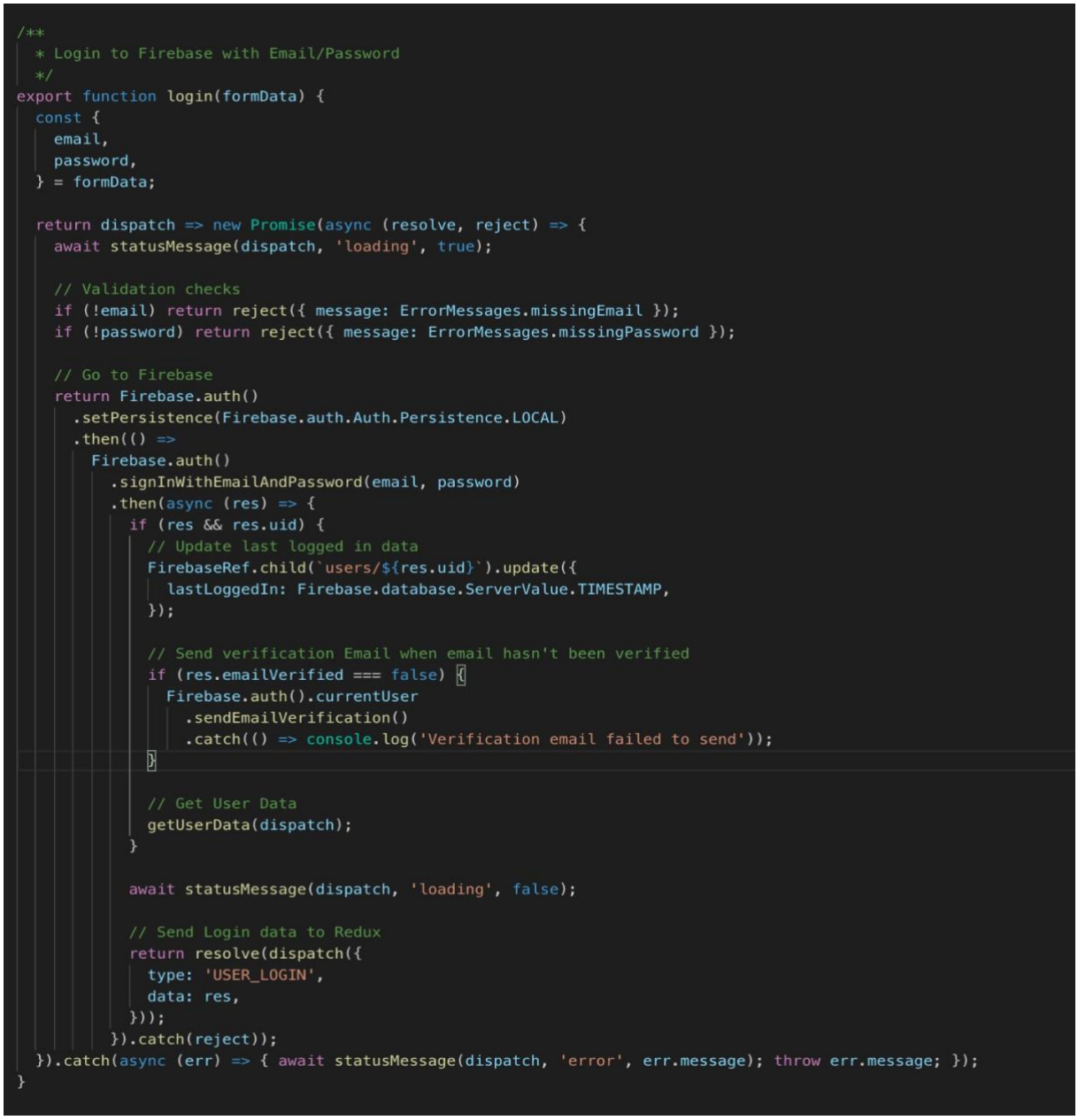
Figura 7.1 Estrutura do banco de dados em js

```
export function newProject(formData) {
  console.log(formData, 'NOVO');
  const {
    projectAuthor,
    projectDateStart,
    projectDateEnd,
    projectSelectUsers,
    projectName,
  } = formData;

  return dispatch => new Promise(async (resolve, reject) => {
    // Validation checks
    if (!projectName) return reject({ message: ErrorMessages.missingFirstName });
    // if (!ProjectDateStart) return reject({ message: ErrorMessages.missingLastName });
    // if (!ProjectDateEnd) return reject({ message: ErrorMessages.missingEmail });
    // if (!ProjectselectUsers) return reject({ message: ErrorMessages.missingPassword });

    await statusMessage(dispatch, 'loading', true);


    return FirebaseRef.child('projects').push({
      title: projectName,
      author: projectAuthor,
      dateStart: projectDateStart,
      dateEnd: projectDateEnd,
      members: projectSelectUsers,
      id: new Date().valueOf(),
    }).then(() => statusMessage(dispatch, 'loading', false).then(resolve));
  }).catch(async (err) => { await statusMessage(dispatch, 'error', err.message); throw err.message; });
}


/**
 * Get Projects
 */
export function getProjects() {
  if (Firebase === null) return () => new Promise(resolve => resolve());

  return dispatch => new Promise(resolve => FirebaseRef.child('projects')
    .on('value', (snapshot) => {
      const get_projects = snapshot.val() || {};
      const projects = Object.keys(get_projects).map(key => get_projects[key]);
      console.log(projects, 'opa');
      return resolve(dispatch({
        type: 'PROJECTS_REPLACE',
        data: projects,
      }));
    })).catch(e => console.log(e));
}
```

Figura 7.2 Script de chamada do Firebase no React Native

```
/**
 * Get this User's Details
 */
function getUserData(dispatch) {
  const UID = (
    FirebaseRef
    && Firebase
    && Firebase.auth()
    && Firebase.auth().currentUser
    && Firebase.auth().currentUser.uid
  ) ? Firebase.auth().currentUser.uid : null;

  if (!UID) return false;

  const ref = FirebaseRef.child(`users/${UID}`);

  return ref.on('value', (snapshot) => {
    const userData = snapshot.val() || [];

    return dispatch({
      type: 'USER_DETAILS_UPDATE',
      data: userData,
    });
  });
}

export function getMemberData() {
  if (Firebase === null) return () => new Promise(resolve => resolve());

  // Ensure token is up to date
  return dispatch => new Promise((resolve) => {
    Firebase.auth().onAuthStateChanged((loggedIn) => {
      if (loggedIn) {
        return resolve(getUserData(dispatch));
      }

      return () => new Promise(() => resolve());
    });
  });
}
```

Figura 7.3 Código de chamada de usuários no banco de dados (firebase)



```
/**
 * Login to Firebase with Email/Password
 */
export function login(formData) {
  const {
    email,
    password,
  } = formData;

  return dispatch => new Promise(async (resolve, reject) => {
    await statusMessage(dispatch, 'loading', true);

    // Validation checks
    if (!email) return reject({ message: ErrorMessages.missingEmail });
    if (!password) return reject({ message: ErrorMessages.missingPassword });

    // Go to Firebase
    return Firebase.auth()
      .setPersistence(Firebase.auth.Auth.Persistence.LOCAL)
      .then(() =>
        Firebase.auth()
          .signInWithEmailAndPassword(email, password)
          .then(async (res) => {
            if (res && res.uid) {
              // Update last logged in data
              FirebaseRef.child(`users/${res.uid}`).update({
                lastLoggedIn: Firebase.database.ServerValue.TIMESTAMP,
              });

              // Send verification Email when email hasn't been verified
              if (res.emailVerified === false) {
                Firebase.auth().currentUser
                  .sendEmailVerification()
                  .catch(() => console.log('Verification email failed to send'));
              }

              // Get User Data
              getUserData(dispatch);
            }

            await statusMessage(dispatch, 'loading', false);

            // Send Login data to Redux
            return resolve(dispatch({
              type: 'USER_LOGIN',
              data: res,
            }));
          }).catch(reject));
  }).catch(async (err) => { await statusMessage(dispatch, 'error', err.message); throw err.message; });
}
```

Figura 7.4 Código de login

```
/**
 * Reset Password
 */
export function resetPassword(formData) {
  const { email } = formData;

  return dispatch => new Promise(async (resolve, reject) => {
    // Validation checks
    if (!email) return reject({ message: ErrorMessages.missingEmail });

    await statusMessage(dispatch, 'loading', true);

    // Go to Firebase
    return Firebase.auth()
      .sendPasswordResetEmail(email)
      .then(() => statusMessage(dispatch, 'loading', false).then(resolve(dispatch({ type: 'USER_RESET' }))))
      .catch(reject);
  }).catch(async (err) => { await statusMessage(dispatch, 'error', err.message); throw err.message; });
}
```

Figura 7.5 Código para modificar a senha

```
/**
 * Sign Up to Firebase
 */
export function signUp(formData) {
  const {
    email,
    password,
    password2,
    firstName,
    lastName,
  } = formData;

  return dispatch => new Promise(async (resolve, reject) => {
    // Validation checks
    if (!firstName) return reject({ message: ErrorMessages.missingFirstName });
    if (!lastName) return reject({ message: ErrorMessages.missingLastName });
    if (!email) return reject({ message: ErrorMessages.missingEmail });
    if (!password) return reject({ message: ErrorMessages.missingPassword });
    if (!password2) return reject({ message: ErrorMessages.missingPassword });
    if (password !== password2) return reject({ message: ErrorMessages.passwordsDontMatch });

    await statusMessage(dispatch, 'loading', true);

    // Go to Firebase
    return Firebase.auth()
      .createUserWithEmailAndPassword(email, password)
      .then((res) => {
        // Send user details to Firebase database
        if (res && res.uid) {
          FirebaseRef.child(`users/${res.uid}`).set({
            firstName,
            lastName,
            signedUp: Firebase.database.ServerValue.TIMESTAMP,
            lastLoggedIn: Firebase.database.ServerValue.TIMESTAMP,
          }).then(() => statusMessage(dispatch, 'loading', false).then(resolve));
        }
      }).catch(reject);
  }).catch(async (err) => { await statusMessage(dispatch, 'error', err.message); throw err.message; });
}
```

Figura 7.6 Código de autenticação de usuários no firebase