**UNISENAI**
**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**
**PROJETO APLICADO 2**
**ENTREGA 1: PREPARAÇÃO DO PROJETO**

**Evandro dos Reis Ferreira Alves**
**Gabriel Oliveira da Silva**
**Jonatas Fred Rossetto**
**Lucas Jose de Paula**

# Demanda: Soluções para o gerenciamento simplificado de hospedagens

## Visão geral

O projeto escolhido é uma solicitação da Pousada Quinta do Ypuã, (https://www.quintadoypua.com/) para o desenvolvimento de um sistema web que centralize de forma simples a gerência de hóspedes e reservas da pousada. Atualmente, este controle é realizado de forma manual levando a uma série de problemas como erros de agendamento, duplicação de reservas ou perda de informações, tendo impacto negativo sobre a operação do negócio.

Em contato com os administradores da pousada, se esclareceu que o sistema solicitado é para uso interno da pousada e deve realizar a gerência de hóspedes, acomodações e reservas, deve ser simples de utilizar e deve operar na web facilitando o acesso à plataforma, além disso o sistema deve ter baixo custo.

A pousada espera diminuir o tempo atualmente gasto com processos manuais e espera melhorar a eficácia da sua operação evitando erros por inconsistência de informações.

## Resultados da primeira entrega

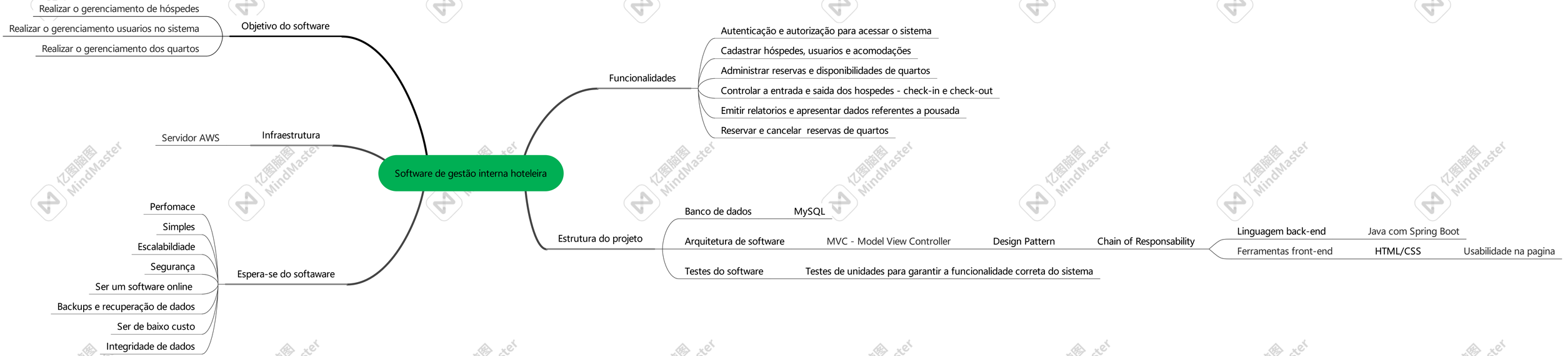
Neste momento estamos apresentando 3 ferramentas importantes para aumentar a nossa compreensão da solicitação do cliente de modo a facilitar o processo de execução e implementação de um MVP dentro do prazo estipulado pela disciplina.

Nosso mapa mental procura expor diversos aspectos relacionados ao problema, incluindo as solicitações do cliente, alguns questionamentos da equipe em relação ao sistema bem como possíveis aspectos técnicos relacionados com a implementação da solução.

Apresentamos também uma lista de requisitos funcionais e não funcionais, procurando tornar mais visível e mais detalhado as funcionalidades que o sistema a ser desenvolvido deverá possuir. Este documento é fundamental para orientar as atividades necessárias para o desenvolvimento do sistema.

Por fim apresentamos uma visão preliminar do cronograma, sob a forma de gráfico de Gantt, das atividades para execução do projeto.

Estes são documentos iniciais para dar um rumo ao projeto, mas sendo atentos ao Manifesto Ágil, estamos prontos para promover rápidas correções de rumo durante o desenvolvimento do produto solicitado pelo cliente.


Software de gestão interna hoteleira
Objetivo do software
Realizar o gerenciamento de hóspedes
Realizar o gerenciamento usuarios no sistema
Realizar o gerenciamento dos quartos
Funcionalidades
Autenticação e autorização para acessar o sistema
Cadastrar hóspedes, usuarios e acomodações
Administrar reservas e disponibilidades de quartos
Controlar a entrada e saida dos hospedes - check-in e check-out
Emitir relatorios e apresentar dados referentes a pousada
Reservar e cancelar reservas de quartos
Infraestrutura
Servidor AWS
Estrutura do projeto
Banco de dados
MySQL
Arquitetura de software
MVC - Model View Controller
Design Pattern
Chain of Responsability
Testes do software
Testes de unidades para garantir a funcionalidade correta do sistema
Linguagem back-end
Java com Spring Boot
Ferramentas front-end
HTML/CSS
Usabilidade na pagina
Espera-se do softaware
Perfomace
Simples
Escalabildiade
Segurança
Ser um software online
Backups e recuperação de dados
Ser de baixo custo
Integridade de dados

# Requisitos Funcionais

**Sistema deve cadastrar novos usuários**

Lista de dados para cadastro dos usuários:

- **Id_usuario** – inteiro, incremental, gerado automaticamente pelo banco de dados, chave primária
- **CPF** - inteiro, com regex para verificar quantidade de números e lógica para verificar se o CPF é válido
- **Nome** - String, com no mínimo 3 caracteres
- **Telefone** - inteiro, com regex para validar quantidade de números
- **E-mail** - String, verificar se possui informações pertinentes a um e-mail
- **Senha** - String, de até 10 caracteres
- **Ativo** - Boolean, será cadastrado e mantido como true, mas será possível colocar como false, com isso o usuário perde os acessos ao sistema
- **Administrador** - Boolean, sempre cadastrado como false, só pode ser alterado por um usuário administrador
- **Data de cadastro** - Local Date, formato (dd-MM-yyyy) deverá ser o dia atual do sistema (preenchido automaticamente pelo sistema)

Implementar, através de triggers no SGBD, uma tabela de auditoria para guardar informações sobre atualizações na tabela de usuários

**Sistema deve atualizar usuários cadastrados**

Lista de dados do usuário que podem ser atualizados:

- **Nome**
- **Telefone**
- **E-mail**
- **Senha**
- **Ativo**
- **Administrador** - só pode ser atualizado por um usuário administrador

Implementar, através de triggers no SGBD, uma tabela de auditoria para guardar informações sobre atualizações na tabela de usuários

**Sistema deve remover/inativar usuários cadastrados**

Deverá ser possível inativar/remover usuários cadastrados no sistema caso seja necessário, esta ação somente poderá ser realizada por usuários cadastrados com administrador.

Implementar, através de triggers no SGBD, uma tabela de auditoria para guardar informações sobre atualizações na tabela de usuários

**Sistema deve listar/detalhar os registros dos usuários cadastrados**

Será possível listar todos os dados dos usuários cadastrados ou apenas um específico através do CPF/ID para verificar as informações

**Sistema deve cadastrar novos hóspedes**

Dados obrigatórios para cadastro dos hóspedes:

- **Id_hospede** – Inteiro, incremental, gerado automaticamente pelo banco de dados
- **Número documento** – String (validade dos dados deve ser verificado pelo tipo de documento)
- **Tipo de documento** – String (rg, cpf, passaporte)
- **Nome** - String com no mínimo 3 caracteres
- **Telefone** – Inteiro, com regex para validar quantidade de números, deve aceitar número de telefone com código do país
- **E-mail** - String e verificar se possui informações pertinentes a um e-mail
- **Endereço** - String com máximo de 255 caracteres
- **Data Nascimento** - Local Date formato (dd-MM-yyyy) com restrição de ano mínimo sendo 1900 e ano máximo ano atual
- **Data de cadastro** - Local Date, formato (dd-MM-yyyy) deverá ser o dia atual do sistema (preenchido automaticamente pelo sistema)
- **Usuário_id** – id do cadastro do usuário que realizou a entrada dos dados do hóspede (preenchido automaticamente pelo sistema)

Implementar, através de triggers no SGBD, uma tabela de auditoria para guardar informações sobre atualizações na tabela de hóspedes.

**Sistema deve atualizar hóspedes cadastrados**

Lista de dados que podem ser atualizados:

- **Nome**
- **Telefone**
- **E-mail**
- **Endereço**

Implementar, através de triggers no SGBD, uma tabela de auditoria para guardar informações sobre atualizações na tabela de hóspedes.

**Sistema deve remover hóspedes cadastrados**

Deverá ser possível remover todos os dados dos hóspedes cadastrados no sistema caso seja necessário.

Implementar, através de triggers no SGBD, uma tabela de auditoria para guardar informações sobre atualizações na tabela de hóspedes.

**Sistema deve listar/detalhar hóspedes cadastrados**

Será possível listar/detalhar todos os dados dos hóspedes cadastrados ou apenas um específico através do nome para verificar as informações.

**Sistema deve cadastrar novas acomodações**

Dados obrigatórios para cadastro de acomodações:

- **Id_quarto –** Inteiro, incremental, gerado automaticamente pelo banco de dados
- **Número do quarto** - Number, validação para receber apenas números positivos e inteiros
- **Quantidade de camas** - Number, validação para receber apenas números positivos e inteiros
- **Capacidade do quarto** – Number, quantidade de pessoas que o quarto pode atender
- **Valor da diária** - Big Decimal, validação para receber apenas números positivos

**Sistema deve atualizar acomodações cadastradas**

Dados que podem ser atualizados:

- **Quantidade de camas**
- **Valor**
- **Número do quarto**
- **Capacidade do quarto**

**Sistema deve remover acomodações cadastradas**

Deverá ser possível remover todos os dados das acomodações cadastradas no sistema caso seja necessário

**Sistema deve listar/detalhar as acomodações cadastradas**

Será possível listar todos os dados das acomodações cadastradas ou apenas um específico para verificar as informações

**Sistema deve reservar acomodações para hóspedes**

Deve realizar o cadastro do hóspede e vincular ele com 1 quarto disponível na data desejada. Dados obrigatórios para reserva:

- **Id_reserva** - Int, incremental, gerado automaticamente pelo banco de dados
- **Hospede_id** – chave estrangeira relacionando com a tabela de hóspedes
- **Acomodação_Id** – chave estrangeira relacionando com a tabela de acomodações
- **Data entrada** - Local Date formato (dd-MM-yyyy) deverá ser no mínimo o dia atual
- **Data saída** - Local Date formato (dd-MM-yyyy) deverá ser no mínimo a data da entrada + 1 dia
- **Desconto** - Number, validação para receber apenas números positivos e inteiros
- **Forma de pagamento** - Seletor, terá as opções (A vista, Pix, Débito e Crédito)
- **Número cartão de crédito** - Int, será validado somente números inteiros e positivos (somente se pagamento for cartão de crédito)
- **Validade cartão de crédito** - Int, será validado somente números inteiros e positivos (somente se pagamento for cartão de crédito)
- **Nome cartão de crédito** - String, (somente se pagamento for cartão de crédito)
- **Quantidade de pessoas na reserva** - Int, para verificar se o quarto acomoda todas as pessoas da reserva
- **Observações adicionais** - campo texto para incluir solicitações ou observações adicionais sobre a reserva

Dados preenchidos automaticamente:

- **Hora entrada** - Time, formato (HH-mm) valor default 14:00 (no momento do check-in)
- **Hora saída** - Time, formato (HH-mm) valor default 11:00 (no momento do check-out)
- **Valor diária** - Big Decimal, valor da acomodação escolhida (no momento da reserva)
- **Valor total** - Big Decimal, valor da diária x quantidade de dias da hospedagem -desconto (no momento da reserva)

**Sistema deve atualizar reservas**

Dados que podem ser atualizados:

- **Acomodação Id** - Número do quarto
- **Data entrada** - Local Date, formato (dd-MM-yyyy) deverá ser no mínimo o dia atual
- **Data saída** - Local Date, formato (dd-MM-yyyy) deverá ser no mínimo a data da entrada + 1 dia
- **Desconto** Number, validação para receber apenas números positivos e inteiros

**Sistema deve cancelar reservas de hóspedes**

Deve ser possível realizar o cancelamento da reserva do hóspede contanto que a data de saída não tenha ocorrido

**Sistema deve listar reservas**

Será possível listar todas as reservas realizadas, por data de entrada, por **Id_reserva** ou por ID do cliente.

**Sistema deve ser web para acesso em qualquer dispositivo**

Deve ser possível acessar o sistema em qualquer dispositivo que tenha conexão com a internet

- Para dispositivos com telas de 5 polegadas teremos um layout simples e com apenas 1 coluna e os botões para acesso
- Para dispositivos com telas de 8 polegadas o layout terá 2 colunas e os botões de acesso
- Para telas de 14 polegadas ou acima o layout será com 4 colunas com a melhor experiência de usabilidade

**Tela Login no sistema**

Permitir entrar como login e senha para acessar o sistema

**Tela principal com diversas opções de ação**

- Administração de usuários
- Administração de hóspedes
- Administração de reservas
- Administração de acomodações
- Sair do sistema

**Tela de administração de usuários**

- Tela cadastrar novo usuário
- Tela atualizar usuário existente
- Tela relatório de usuários
- Tela de ativar/inativar usuários

**Tela de administração de hóspedes**

- Tela cadastro de hóspede
- Tela atualizar hóspede existente
- Tela relatório de hóspede
- Tela de remover hóspedes

**Tela de administração de reservas**

- Tela cadastro de reserva
- Tela check-in / check-out acredito que esta incluso no cadastro da reserva
- Tela atualizar reserva
- Tela relatório de reservas
- Tela de cancelar reservas

**Tela de administração de** acomodações

- Tela cadastro de acomodações
- Tela atualizar acomodações
- Tela relatório de acomodações
- Tela de remover acomodações

# Requisitos não funcionais

**Sistema deve conter um banco de dados relacional (SQL)**

No banco de dados serão armazenadas todas as tabelas e informações do sistema

- Tabela de hóspedes
- Tabela de usuários
- Tabelas de acomodações
- Tabela de reservas

**Sistema deve ter backup e recuperação de dados**

O sistema fará backup dos dados frequentemente para segurança e recuperação de dados quando for necessário.

- Será feito um backup completo do banco 1 vez por mês, todo dia 1
- Será feito backup diferencial a cada 3 dias
- Backup será armazenado na store da nuvem (S3 ou similar)

Em caso de restauração de dados será analisado de acordo com a situação

- Falha catastrófica, ou seja, todos os dados foram perdidos, então usaremos o último backup completo que temos da nuvem.
- Falha não catastrófica, perdemos alguns dados ou tabelas, usaremos o backup mais recente que teremos diferencial.

**Integridade dos dados**

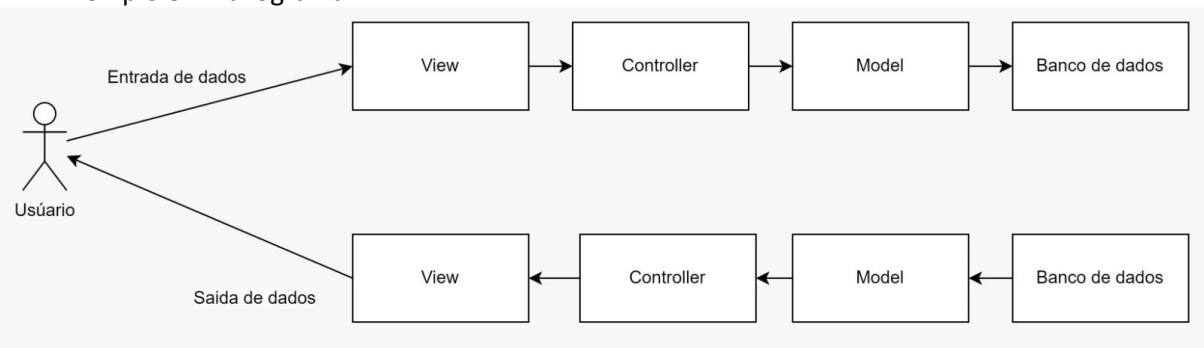
O Sistema terá diversas validações na entrada dos dados para garantir que os dados cheguem íntegros no banco, mas no banco também terá constrains como segurança, ou seja todos os dados armazenados terão a integridade garantida.

**Arquitetura do sistema**

Será feito na arquitetura de camadas MVC (Model View Controller), e cada camada será responsável por passar informações para outra camada e esta por sua vez processará o que compete a ela. Desta forma organizamos o código e visivelmente se torna fácil e de dar manutenção, além disso seguimos um padrão de arquitetura que qualquer programador futuramente entenderá o código.

Exemplo: a view será usada por usuários credenciados para qualquer tipo de ação, a view acionará o controller, se for uma ação de cadastrar, por exemplo, o controller direcionará para outra camada de serviço o model e este aplicará regras de negócio definidas e por sua vez chamará a camada de banco de dados e está realizará o cadastro na base. E por fim uma resposta padronizada será trafegada de volta do banco para a model depois para o controller e por fim para a view.

Exemplo em fluxograma:



**Observabilidade do sistema**

O sistema terá em suas linhas de códigos os logs de “info e erro” onde serão padronizados mensagens e payload que serão informados nestes trechos e desta forma teremos a observabilidade de todo o funcionamento e performance do sistema.

Estes logs serão expostos em ferramentas como o Splunk e similares para podermos acompanhar o sistema quando ele estiver em produção.

**Sistema deve ser responsivo**

O sistema será possível de usar em dispositivos móveis com a tela menor em caso de necessidade.

**Sistema deve ter usabilidade**

Deve ser de fácil manuseio e com interface agradável para os usuários.

**Sistema deve ter segurança**

Deve conter segurança para garantir sigilo das informações no banco de dados.

Somente usuários autenticados podem acessar as informações, além disso o sistema terá sanitização de dados para minimizar possíveis ataques.

**Performance**

Deve ser performático para melhor experiência do usuário e funcionamento como um todo.

**Disponibilidade**

Deve ter a capacidade de sempre estar disponível para acesso, por isso será hospedado em nuvem (AWS ou similar) será 24/7.

# Planejador de Projetos

*Selecione um período para realçar à direita. A seguir há uma legenda que descreve o gráfico.*

**Realce do Período: #**

Duração do Plano | Início Real | % concluída | Real (além do plano) | % Concluída (além do planejado

| ATIVIDADE | INÍCIO DO PLANO | DURAÇÃO DO PLANO | INÍCIO REAL | DURAÇÃO REAL | PORCENTAGEM CONCLUÍDA |
|---|---|---|---|---|---|
| **Montar Grupo** | 1 | 1 | 1 | 1 | 100% |
| **Escolher o tema** | 2 | 1 | 2 | 1 | 100% |
| **Lista de requisitos** | 3 | 2 | 3 | 2 | 100% |
| Requisitos Funcionais - Evandro e Lucas | 3 | 2 | 3 | 2 | 100% |
| Requisitos não funcionais - Jonatas e Gabriel | 3 | 2 | 3 | 2 | 100% |
| **Dimensar recursos humanos** | 3 | 1 | 3 | 2 | 100% |
| Listar tarefas em ordem cronológica | 3 | 1 | 3 | 2 | 100% |
| Distribuir tarefas | 3 | 1 | 3 | 2 | 100% |
| **Mapa Mental** | 4 | 1 | 4 | 1 | 100% |
| Mapa mental - Evandro e Lucas | 4 | 1 | 4 | 1 | 100% |
| Mapa mental - Jonatas e Gabriel | 4 | 1 | 4 | 1 | 100% |
| **Criar diagrama UML** | 5 | 3 | | | 0% |
| Criar diagrama entidade e relacionamento - Evandro e Jonatas | 5 | 1 | | | 0% |
| Criar diagrama de classes - Gabriel e Lucas | 5 | 1 | | | 0% |
| **Desenvolver Front End** | 8 | 3 | | | 0% |
| Desenvolver tela de login | 8 | 3 | | | 0% |
| Desenvolver tela de cadastro | 8 | 3 | | | 0% |
| Desenvolver tela de reserva | 8 | 3 | | | 0% |
| Desenvolver tela de relatório | 8 | 3 | | | 0% |
| Desenvolver tela de deleção | 8 | 3 | | | 0% |
| Conectar telas com o Back-End | 11 | 1 | | | 0% |
| **Desenvolver Back End** | 8 | 5 | | | 0% |
| Criar a estrutura do projeto | 8 | 1 | | | 0% |
| Criar camada de controller | 8 | 1 | | | 0% |
| Criar camada de serviço | 8 | 1 | | | 0% |
| Criar validações e regras de negócios | 8 | 4 | | | 0% |
| Criar camada de banco de dados | 8 | 1 | | | 0% |
| Testes locais | 8 | 5 | | | 0% |
| Criar estrutura de logs | 9 | 1 | | | 0% |
| Conectar Back-End com as telas | 11 | 1 | | | 0% |
| Conectar Back-End com Banco | 12 | 1 | | | 0% |
| Criar testes de unidade | 13 | 1 | | | 0% |
| **Desenvolver Banco de dados** | 11 | 3 | | | 0% |
| Criar Database | 11 | 1 | | | 0% |
| Criar usuários e polices de acessos | 11 | 1 | | | 0% |
| Criar tabela hóspede | 11 | 1 | | | 0% |
| Criar tabela usuário | 11 | 1 | | | 0% |
| Criar tabela acomodação | 11 | 1 | | | 0% |
| Criar tabela reservas | 11 | 1 | | | 0% |
| Criar plano de backup | 12 | 1 | | | 0% |
| Criar triggers | 13 | 1 | | | 0% |
| **Gravar Pitch** | 14 | 2 | | | 0% |
| Elaborar enredo | 14 | 1 | | | 0% |
| Ensaio de gravação | 14 | 1 | | | 0% |
| Gravar Pitch | 15 | 1 | | | 0% |