

Documentação - Sistema Digital de Controle de Horas Extras

Contexto do Problema

O problema central é o processo manual de controle de horas extras, que atualmente depende de formulários físicos e validações manuais. Isso gera lentidão, erros de cálculo e dificuldade de fechamento da folha nos horários exigidos (13h ou 14h), sobrecarregando os gestores e encarregados.

Objetivo do Sistema

Desenvolver um sistema digital prototipado em Linguagem C para automatizar o registro, cálculo e validação de horas extras. O foco é garantir a consistência lógica dos dados, aplicar as regras de negócio de aprovação e preparar a base para futuras interfaces web/mobile.

Stakeholders

Gestores/Encarregados: Aprovam as horas e geram relatórios.

Funcionários/Técnicos: Registram os horários de entrada e saída.

Startup Base27: Parceira interessada na solução do problema.

Requisitos Funcionais

RF01 - Login: O sistema deve diferenciar o acesso entre Funcionário e Gestor.

RF02 - Registro de Horas: Permitir inserção de data, horas trabalhadas e tipo de dia (útil/fim de semana).

RF03 - Validação de Horário: Bloquear aprovações de horas extras após o horário de corte (13h ou 14h).

RF04 - Cálculo Financeiro: Calcular o valor da hora extra com base no salário e adicionais (50%, 100%, noturno).

RF05 - Relatório: Gerar um extrato final consolidado com status (Aprovado/Reprovado).

Requisitos Não Funcionais

RNF01 - Tecnologia: O protótipo deve ser desenvolvido estritamente em Linguagem C.

RNF02 - Interface: Utilização de interface via console (CLI) para validação lógica.

RNF03 - Portabilidade: O código deve ser compilável em GCC padrão.

Regras de Negócio

RN01 - Corte de Aprovação: O gestor só pode aprovar horas até as 14h do dia de fechamento.

RN02 - Cálculo de Adicional: Dias de semana = 50% (ou conforme CLT), Domingos/Feriados = 100%.

RN03 - Hierarquia: Apenas usuários marcados como 'Gestor' podem alterar o status de uma hora extra para 'Aprovado'.

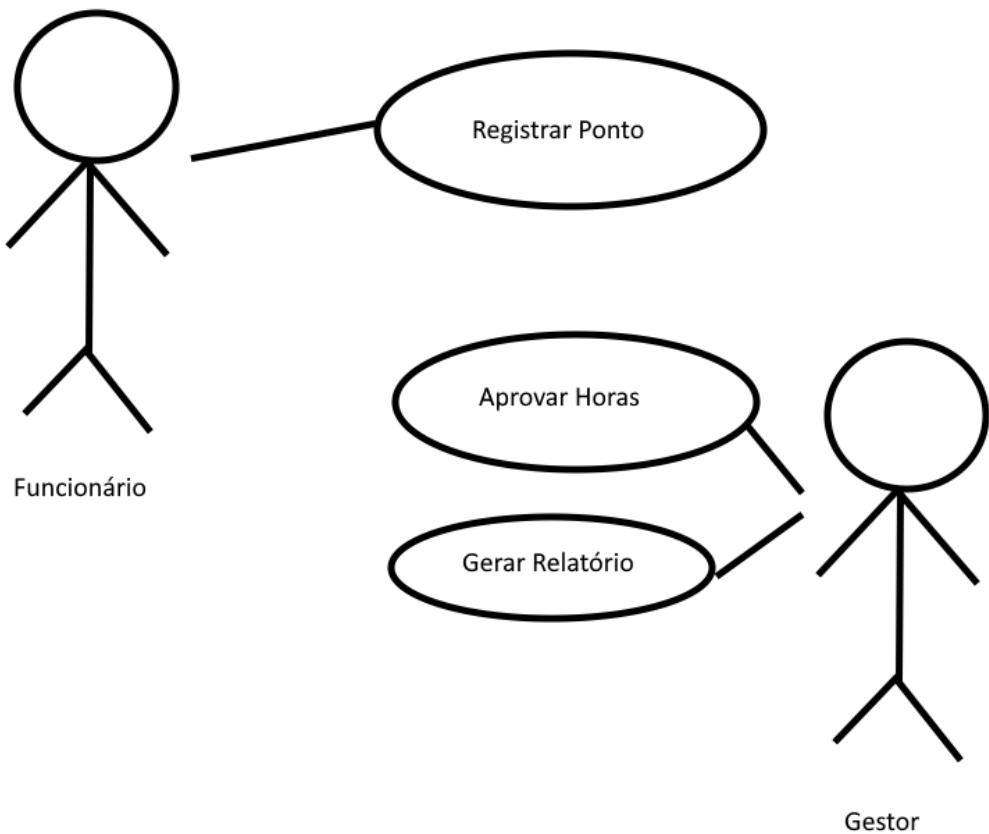
User Stories (História de usuário)

"Como **Funcionário**, quero registrar minhas horas extras digitalmente para não precisar preencher papéis."

"Como **Gestor**, quero visualizar todas as solicitações pendentes para aprovar ou reprovar rapidamente antes das 14h."

"Como **Gestor**, quero que o sistema calcule os valores automaticamente para evitar erros de pagamento."

Escolher entre: Diagrama, Fluxograma ou caso de uso



Protótipo da interface

Tela 1 - Login

Heurística 1
Visibilidade do status do sistema:
Mostra carregamento e mensagem de login incorreto.
Ex: Exiba mensagens de sucesso ("sucesso") ou de erro ("Senha incorreta").

Heurística 5
Prevenção de erros:
Validação de campos obrigatórios.
Ex: Desabilite o botão "Entrar na conta" até que todos os campos obrigatórios estejam preenchidos.

Heurística 2.
Correspondência entre o sistema e o mundo real:
A interface deve falar a linguagem do usuário, usando conceitos familiares e uma ordem lógica.
Ex: Em vez de "Autenticação", use "Entrar na conta".

Tela 3 - Calendário - técnico

Heurística 6
Reconhecimento em vez de memorização:
Menus e ícones.
Ex: Menus laterais fixos com ícones e rótulos (Perfil, Configurações)

Heurística 4
Consistência e padrões:
Usuários não devem precisar adivinhar se palavras, ações ou situações significam a mesma coisa.
Ex: Ícones com significados universais

Heurística 3
Controle e liberdade do usuário:
Usuários devem poder desfazer e refazer opções facilmente.
Exemplo:
Botão "X" Símbolo de emergência.
Permitindo encerrar imediatamente a sessão ou retornar ao menu inicial.

Heurística 5
Prevenção de erros:
Melhor do que mensagens de erro é um design que evite erros antes que aconteçam.
Ex: Confirmação antes de aprovar/recusar.

Heurística 6.
Design estético e minimalista:
Cada tela deve conter apenas informações relevantes — nada superfluo.
Ex: Destaque uma opção principal por tela.

Tela 4 - Relatório

Encarregado
Como encarregado, quero registrar as horas extras de um técnico, para que o sistema calcule e processe corretamente o pagamento das horas adicionais.

Heurística 6
Reconhecimento em vez de memorização:
Menus e ícones:
Ex: Menus laterais fixos com ícones e rótulos (Perfil, Configurações).

Heurística 4
Consistência e padrões:
Usuários não devem precisar adivinhar se palavras, ações ou situações significam a mesma coisa.
Ex: Ícones com significados universais

Heurística 3
Controle e liberdade do usuário:
Usuários devem poder desfazer e refazer suas ações.
Exemplo:
Botão "X" sólo de emergência:
Permitindo encerrar o sistema, reiniciar o sessão ou retornar à tela inicial.

Heurística 8
Design estético e minimalista:
Cada tela deve conter apenas informações relevantes — nada supérfluo.
Ex: Destaque uma ação principal por tela.

Heurística 5
Prevenção de erros:
Melhor do que mensagens de erro é um design que evite erros antes que aconteçam.
Ex: Confirmação antes de aprovar/recusar.

Tela 5 - Solicitação - Gestor

Gestor
Como gestor, quero saber quantas horas e técnico já cumpriu de horas extras para validar se há necessidade de convocação ou recuperação de horas extras no sistema.

Heurística 4
Consistência e padrões:
Usuários não devem precisar adivinhar se palavras, ações ou situações significam a mesma coisa.
Ex: Ícones com significados universais

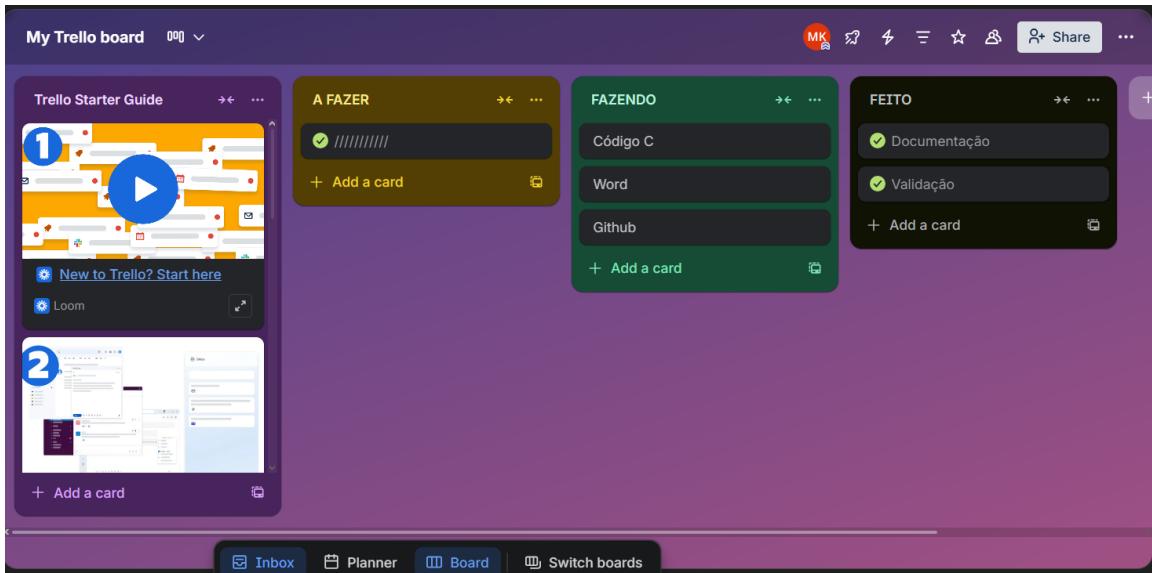
Heurística 6
Reconhecimento em vez de memorização:
Menus e ícones:
Ex: Menus laterais fixos com ícones e rótulos (Perfil, Configurações).

Heurística 3
Controle e liberdade do usuário:
Usuários devem poder desfazer e refazer suas ações.
Exemplo:
Botão "X" sólo de emergência:
Permitindo encerrar o sistema, reiniciar o sessão ou retornar à tela inicial.

Heurística 5
Prevenção de erros:
Melhor do que mensagens de erro é um design que evite erros antes que aconteçam.
Ex: Confirmação antes de aprovar/recusar.

Heurística 8
Design estético e minimalista:
Cada tela deve conter apenas informações relevantes — nada supérfluo.
Ex: Destaque uma ação principal por tela.

Acompanhamento da Equipe (Trello ou Jira)



Matriz de Rastreabilidade

ID	Descrição do Requisito	Onde está o Código	Onde está o Design
RF01	Sistema deve diferenciar Login de Gestor e Funcionário.	struct Funcionario (campo ehGestor)	Tela 1
RF02	Registrar entrada e saída de horas extras.	Função cadastrarHoraExtra()	Tela 4
RF03	Validar horário de corte para aprovação (13h/14h).	Função verificarHorario() if lógico	Tela 5
RF04	Calcular valor financeiro da hora extra (50%/100%).	Função calcularValorExtra()	N/A
RF05	Gerar relatório final consolidado.	Função mostrarResultados() printf	Tela 5

Conclusão

O projeto atingiu o objetivo de prototipar a lógica de negócios em C, garantindo o entendimento dos fluxos de dados e requisitos para a futura implementação web.