

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'.

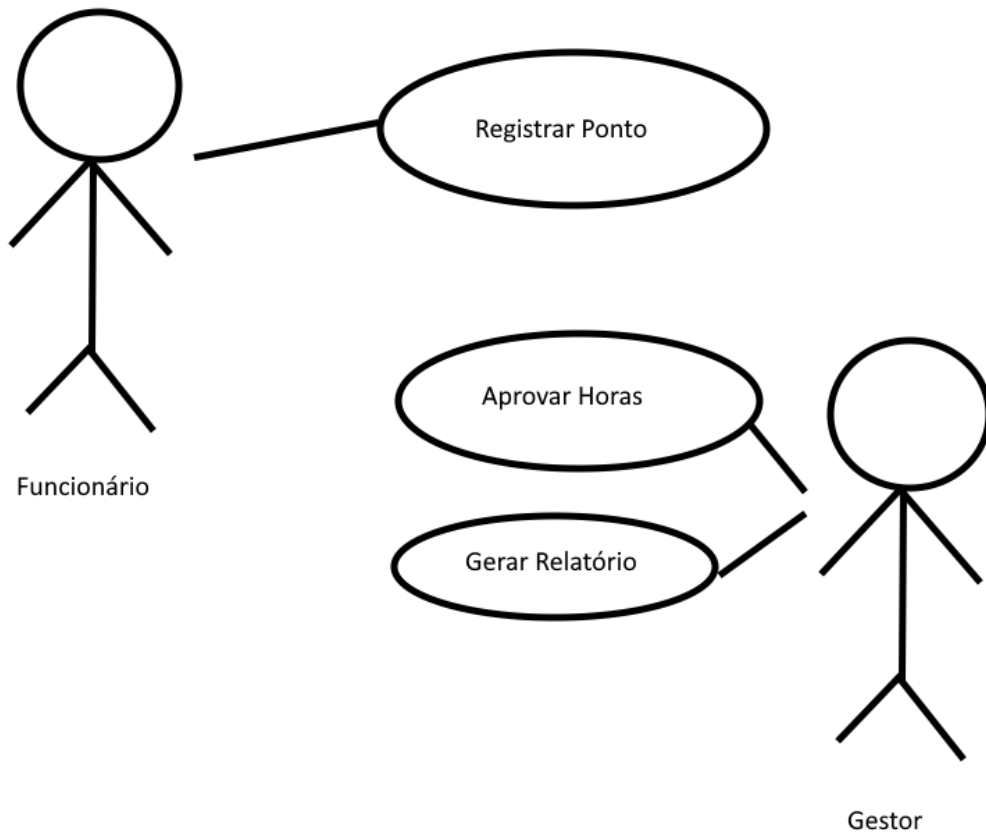
User Stories (História de usuário)

"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".

A Web Page

https://talentos.fortes.com.br

Notificação: Sucesso!

Notificação: Login e Senha incorreto.

Fortes Engenharia

Preencha todos os campos obrigatórios antes de continuar

Seu@email.com

XXXXXXXXXX

[Cadastrar](#)

[Esqueci minha senha](#)

[Termos de uso](#) | [Política de Privacidade](#)

Tela 3 - Calendário - tecnico

Técnico

Como técnico, quero ser notificado sobre o status das minhas horas extras, para saber se foram aprovadas ou não.

Heurística 3

Controle e liberdade do usuário:

Usuários devem poder desfazer e refazer ações facilmente.

Exemplo:

Botão "Y" Botão de emergência: Permite ao usuário rapidamente a 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 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 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.

Fortes Engenharia

Minhas Horas Extras

Pesquisar

Menu: Calendário Lista Relatório

Horas	Data	Horas Cumpridas	Horas restantes	Horas Extras
10 horas	10/10/2023	✓	0	0 horas
10 horas	10/10/2023	✓	0	0 horas
10 horas	10/10/2023	✗	0	0 horas

Notificação: Confirmação e aprovação?

Calendário

Calendário
<div>7</div> <div>8</div> <div>9</div> <div>10</div> <div>11</div> <div>12</div> <div>13</div> <div>14</div> <div>15</div> <div>16</div> <div>17</div> <div>18</div> <div>19</div> <div>20</div> <div>21</div> <div>22</div> <div>23</div> <div>24</div> <div>25</div> <div>26</div> <div>27</div> <div>28</div> <div>29</div> <div>30</div> <div>31</div>

Total de Horas Extras

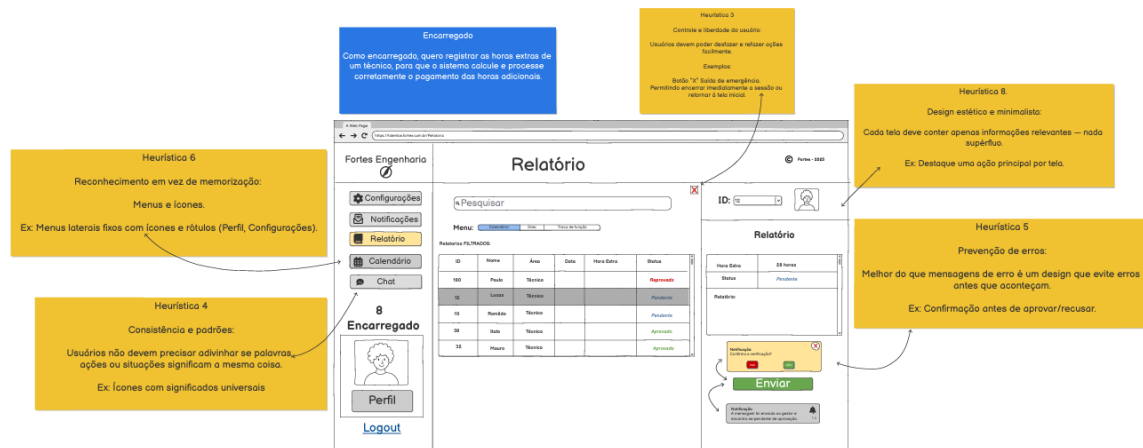
28

Total de Horas Restantes

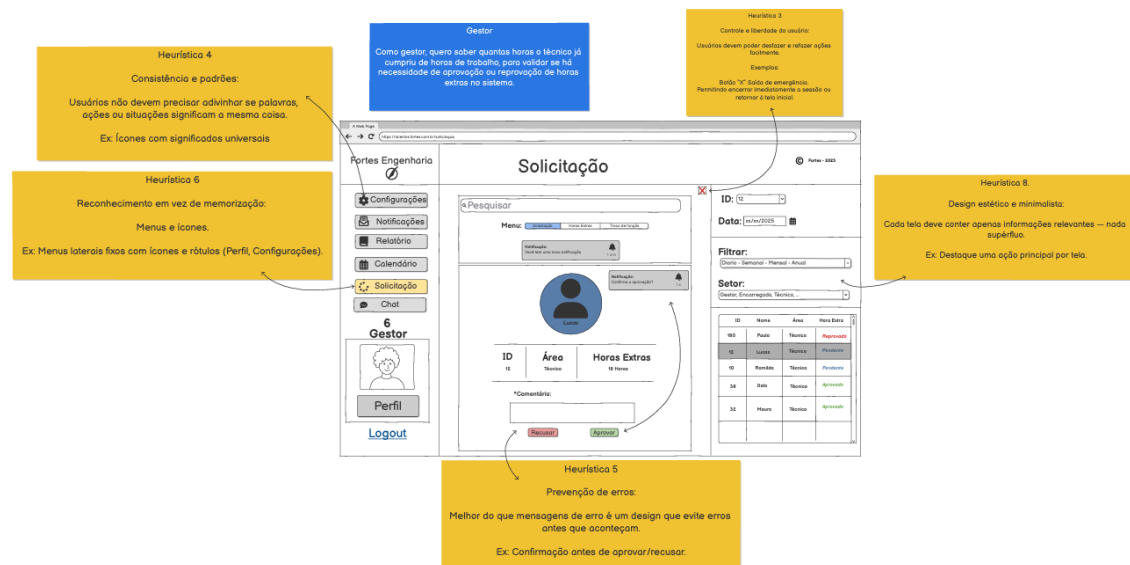
0

Logout

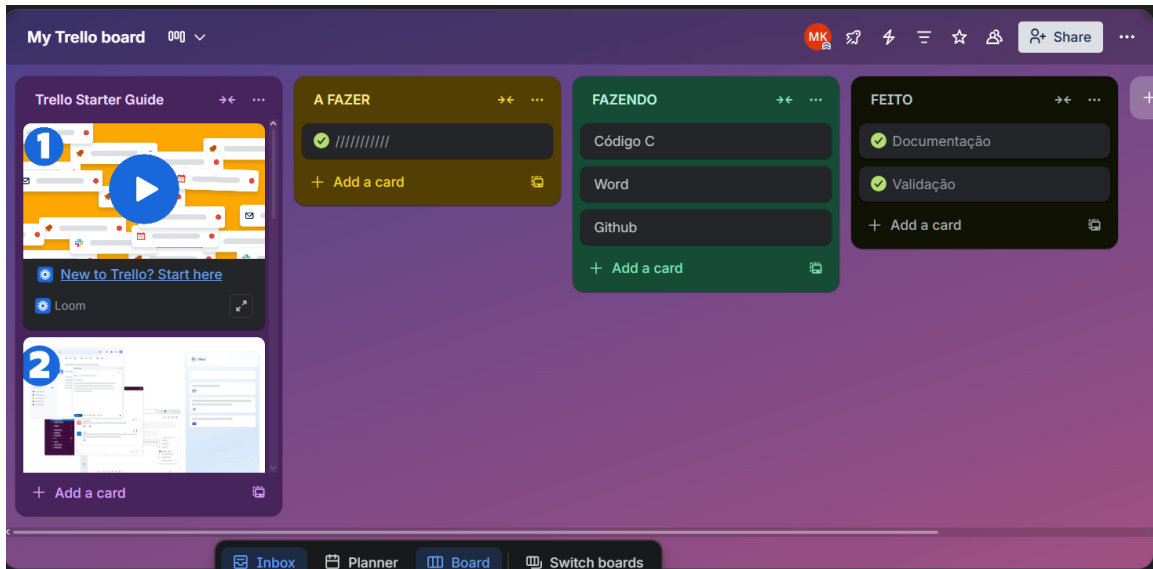
Tela 4 - Relatório



Tela 5 - Solicitação - Gestor



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.