Projeto final

Registro de Ponto

Aluno: Raphael Rodrigues Silva

Agosto, 2023

Objetivos

Esse projeto tem por objetivo a criação de uma aplicação para Windows capaz de gerenciar o registro de ponto de funcionários, utilizando C#, e os frameworks WPF ou UWP ou WinUI3.0 e XAML.

Requisitos

- Preferencialmente fazer uso do banco de dados SQLite para persistência de dados, quando aplicável.
- Preferencialmente utilizar um padrão de arquitetura em camadas, por exemplo MVC ou MVVM.
- Implementar uma interface gráfica capaz de:
    - Cadastrar novos funcionários (nome, foto, ID único, cargo/área).
    - Fazer busca por funcionários (por nome e ID) e exibir detalhes.
    - Registrar horários de chegada/saída de cada funcionário.
    - Exibir avisos/notificações quando algum funcionário tiver mais de 5 faltas por mês.
    - Opcional: gerar um relatório mensal listando os funcionários por ordem decrescente de faltas.
    - Opcional: registrar entradas e saídas durante o expediente.
    - Opcional: registrar abono por dispensa médica.
- Opcional: implementar testes de unidade.

Experiência do Usuário

A jornada começou com a técnica de Brainstorming, na qual a equipe se reuniu para gerar ideias sobre o software. Isso permitiu identificar os principais stakeholders, os atores que interagem com o sistema, e dividir os usuários em dois grupos: Trabalhadores e Gestores. Os Trabalhadores são os funcionários que registrarão seus horários, enquanto os Gestores são responsáveis pelos ajustes na folha de ponto. Essa segmentação proporcionou uma abordagem mais direcionada para atender às necessidades específicas de cada grupo.

Para aprofundar a compreensão das necessidades dos usuários, realizamos entrevistas com uma Assistente Administrativa, que possui uma perspectiva valiosa sobre a rotina de ajustes na folha de ponto e sobre a expectativa de registro do próprio ponto como funcionária. Essa entrevista ajudou a identificar as seguintes necessidades e oportunidades:

Necessidades dos Trabalhadores:

- Realizar o registro de ponto de maneira eficiente.
- Ter um comprovante do registro para referência.
- Acessar informações sobre marcações, faltas e abonos.
- Receber notificações/alertas relacionados ao registro de ponto.
- Submeter atestados médicos.
- Registrar o ponto em situações especiais (técnicos externos).
- Acompanhar horas extras e descontos.

Necessidades dos Gestores:

- Acesso seguro através de login e senha.
- Gerenciar licenças de usuários.
- Realizar ajustes manuais na folha de ponto.
- Lidar com qualquer lentidão ou problemas de performance.
- Facilitar o contato com os funcionários para esclarecimentos.
- Receber notificações/alertas sobre questões relevantes.
- Garantir a assinatura da folha de ponto.

A partir dessas necessidades, foram exploradas alternativas de design, tanto aquelas semelhantes a soluções existentes quanto ideias inovadoras. Com base nisso, foram desenvolvidos protótipos de baixa e alta fidelidade. Os protótipos de baixa fidelidade permitiram visualizar a estrutura e fluxo da aplicação, enquanto os de alta fidelidade trouxeram uma representação visual mais realista do software, incorporando elementos visuais, layout e interações.

A abordagem centrada no usuário, incluindo a identificação das necessidades, a realização de entrevistas e a criação de protótipos iterativos, contribuiu para o desenvolvimento de um software mais alinhado com as expectativas dos usuários e mais intuitivo de usar. Dessa forma, garantimos que a experiência do usuário seja uma prioridade, resultando em uma aplicação que atenda efetivamente às necessidades dos Trabalhadores e Gestores, proporcionando uma experiência de usuário mais positiva e satisfatória.

Gerenciamento de Projeto

A equipe adotou abordagens eficazes para garantir um desenvolvimento organizado e eficiente do software de Registro de Ponto. A metodologia ágil foi escolhida para guiar o processo, com o objetivo de permitir adaptação contínua, priorização das necessidades dos usuários e entregas incrementais.

Para iniciar, estabeleceu-se uma estrutura de trabalho no Azure DevOps, onde foi criado um Projeto para centralizar todas as etapas do processo. Isso proporcionou um ambiente para o planejamento, rastreamento e execução do projeto.

Um aspecto fundamental foi a definição do Product Backlog, para garantir uma visão clara do que estava por vir. O próximo passo envolveu a criação de User Stories, que descreviam as necessidades dos usuários em um formato claro e conciso, sendo a base para a priorização e o planejamento das iterações.

As iterações foram organizadas de acordo com o backlog, dividindo o desenvolvimento em ciclos menores e mais gerenciáveis. No entanto, é importante destacar que, devido à equipe ser composta por uma única pessoa, a divisão tradicional de responsabilidades ágeis entre membros da equipe não se aplicou diretamente. Mesmo assim, os princípios ágeis continuaram a guiar o projeto, permitindo a flexibilidade necessária para adaptações.

Embora a estrutura ágil tradicional de divisão de responsabilidades possa não ter sido aplicada completamente devido ao tamanho da equipe, a criação do Product Backlog e das User Stories ainda desempenhou um papel fundamental. Elas proporcionaram clareza nas metas, ajudaram a priorizar o desenvolvimento e auxiliaram na definição dos marcos.

Em resumo, a metodologia ágil, embora com adaptações devido ao tamanho da equipe, ofereceu um roteiro para o desenvolvimento do software de Registro de Ponto. A abordagem centrada no Azure DevOps, desde a criação do Product Backlog até a execução das iterações, facilitou o planejamento e a execução das tarefas, permitindo que a equipe entregasse um produto que atendesse às necessidades dos usuários.

Versionamento de código

O projeto adotou práticas eficazes para garantir a organização e o controle das alterações realizadas no código-fonte. O sistema de controle de versão GIT foi a ferramenta escolhida para gerenciar as iterações e o histórico de desenvolvimento.

Para manter uma estrutura organizada e rastreável, a equipe adotou um padrão de nomenclatura para branches e mensagens de commits. Cada tela era associada a uma nova branch. Essa abordagem permitiu que cada desenvolvimento fosse isolado e testado separadamente antes de ser integrado à versão principal do código.

O padrão utilizado para as mensagens de commits foi claro e descritivo, seguindo as práticas do desenvolvimento colaborativo. As mensagens eram concisas e comunicavam de forma precisa a alteração realizada, facilitando a compreensão das mudanças para a equipe.

A abordagem referente aos commits, foi a de que os commits eram realizados somente quando o código estava funcional para as funções implementadas. Isso garantiu que cada commit fosse uma contribuição válida e testada, reduzindo a possibilidade de introduzir erros na base de código principal.

A estratégia de branches também contribuiu para um fluxo de trabalho mais organizado. Cada branch era responsável por uma tarefa específica, o que facilitou o gerenciamento de versões e a identificação de áreas de desenvolvimento. Assim que uma tarefa era concluída e testada, o merge na branch principal ("main") era realizado, consolidando as mudanças no projeto como um todo.

Em relação ao README do repositório, ele desempenha o papel fundamental perfeitamente para compreensão e na execução da solução de software por pessoas não ligadas ao grupo de desenvolvimento. O README contém informações detalhadas sobre a configuração do ambiente de desenvolvimento, as dependências necessárias, as etapas de compilação e execução, bem como os recursos e funcionalidades oferecidos pelo software. Esse guia completo permite que qualquer pessoa interessada em executar o software pudesse fazê-lo de maneira suave, mesmo sem estar envolvida no processo de desenvolvimento.

Interface do Usuário

Quanto a Interface do Usuário (UI), para o desenvolvimento da interface gráfica, foi adotado o framework Windows Presentation Foundation (WPF), que possibilita a criação de interfaces ricas e interativas.

A escolha do WPF como framework ofereceu diversas vantagens, incluindo a capacidade de criar interfaces atraentes e responsivas. O WPF também facilitou a separação das preocupações entre a lógica de apresentação e os dados subjacentes, o que foi fundamental para adotar um design orientado a padrões.

O padrão de arquitetura MVVM (Model-View-ViewModel) foi adotado como base para o desenvolvimento da interface do usuário. Essa abordagem promove a separação clara entre a lógica de negócios, a apresentação visual e a manipulação de dados, resultando em um código mais organizado e de fácil manutenção.

O motivo da escolha do framework WPF e do pattern MVVM no desenvolvimento desse projeto ocorreu devido ambos terem sido apresentados na disciplina de Interface do Usuário do curso.

Desenvolvimento

Este projeto empregou uma combinação de tecnologias, padrões e ferramentas para criar uma solução coesa, eficiente e de alta qualidade.

Para a construção da lógica de apresentação e interação com o usuário, foi adotado o framework Windows Presentation Foundation (WPF). O WPF permitiu a criação de interfaces gráficas ricas e interativas, possibilitando uma experiência de usuário aprimorada. Além disso, o WPF se integrou perfeitamente ao padrão arquitetural adotado, o MVVM (Model-View-ViewModel).

O desenvolvimento do código foi realizado principalmente por meio do VS Code, devido à familiaridade da equipe com a IDE e sua simplicidade e eficiência para tarefas mais enxutas.

Para a persistência de dados, foi adotado o SQLite. No banco de dados, foram criadas duas tabelas principais: "Funcionarios" e "RegistroPonto". A tabela "Funcionarios" armazena os detalhes dos funcionários cadastrados, enquanto a tabela "RegistroPonto" registra as batidas de ponto dos funcionários.

As operações de criação, leitura e exclusão foram implementadas por meio do Dapper, simplificando a interação com o banco de dados, e permitindo a execução de consultas e atualizações de maneira eficaz.

Segurança

No contexto da Segurança, este projeto adotou abordagens proativas para garantir a proteção dos dados e a confidencialidade das informações dos usuários. A segurança não foi apenas uma consideração posterior, mas sim um aspecto fundamental desde o início do planejamento.

Desde a fase de ideação e design, o projeto já estava atento às medidas de segurança necessárias. Durante o desenvolvimento, foram tomadas precauções para garantir a integridade dos dados e minimizar riscos potenciais. No entanto, devido ao curto prazo de entrega, algumas funções de segurança planejadas não foram implementadas neste momento.

Entre as medidas de segurança consideradas estão:

Acesso Controlado: A tela de gestão foi protegida por uma combinação de usuário e senha. Embora o usuário "admin" com a senha "admin" seja uma senha fraca, isso foi incluído apenas como prova de conceito no MVP e para facilitar a demonstração. No entanto, vale ressaltar que essa implementação direta no código não está sujeita a ataques de SQL Injection.

Prevenção de SQL Injection: A utilização de um ORM como o Dapper auxiliou na prevenção de ataques de SQL Injection, pois evita a construção manual de consultas SQL.

Futuras Implementações: Embora algumas funcionalidades planejadas para aprimorar a segurança, como a criptografia do banco de dados e o registro de ponto com cálculos de hashes, não tenham sido implementadas no escopo atual do projeto, a equipe tem a intenção de incorporá-las em futuras iterações.

Gerenciamento de Acesso: A segmentação de usuários em Trabalhadores e Gestores contribui para um melhor controle de acesso e atribuição de permissões, aumentando a segurança e evitando acessos não autorizados.

Identificação Positiva: Desde o início do projeto, nos protótipos criados, houve a intenção de implementar uma Identificação Positiva numérica (PID) para a marcação do ponto dos funcionários. Porém, no MVP implementamos poucos dados no cadastro de funcionários.

Política de Senhas Fortes: Embora não tenha sido implementada nessa versão, a criação de uma política de senhas fortes para os usuários é uma consideração importante para garantir a segurança das contas.

Auditoria: Futuramente, a implementação de uma funcionalidade de auditoria permitirá o rastreamento de todas as atividades realizadas no sistema (Logs), o que pode ser valioso para identificar qualquer comportamento suspeito.

Hospedagem

Referente a Hospedagem, este projeto foi planejado com um enfoque voltado para o desenvolvimento local e a entrega do MVP (Produto Mínimo Viável). A hospedagem em nuvem foi considerada, inclusive criado um banco de dados no Azure, mas devido ao curto prazo para entrega e ao escopo do MVP, houve à necessidade de acelerar a entrega, o desenvolvimento inicial foi realizado localmente usando o SQLite como banco de dados.

Durante o desenvolvimento, algumas funcionalidades relacionadas à hospedagem foram planejadas, mas não implementadas neste momento. Entre elas:

Hospedagem Web: A ideia de hospedar uma versão web para a marcação de ponto e gerenciamento remoto foi considerada.

Integração Contínua (CI): A criação de pipelines de integração contínua usando ferramentas como Azure DevOps ou GitHub Actions.

Links Úteis

Link da entrevista:

https://youtu.be/wzmQsVlgbwA

Link do Repositório:

https://github.com/RapGod91/RegistroPonto

Link da Pasta no Google Drive com Executável, Apresentação e Relatório:

https://drive.google.com/drive/folders/1-JWXOzzRX3OnesY8dfk72a-BRyx-u-6I?usp=sharing