Documentação Projeto Final de Programação para Dispositivos móveis

Aluno: Pedro Arthur Cunha Anício

1. Introdução:

O projeto desenvolvido consiste em um aplicativo mobile para rastreamento de hábitos, com foco no gerenciamento de metas diárias de estudo. O aplicativo permite que os usuários criem, editem e excluam hábitos, além de monitorar o tempo dedicado a cada um deles. O objetivo é ajudar os usuários a manterem uma rotina produtiva, fornecendo gráficos e métricas para acompanhar seu progresso.

O aplicativo utiliza um banco de dados SQLite para armazenar informações sobre hábitos, progresso diário e configurações de Pomodoro. A interface é intuitiva, com gráficos interativos e uma lista de hábitos que podem ser gerenciados facilmente.

1.2 Implementação

Estrutura de Dados

O aplicativo foi implementado utilizando a linguagem Kotlin, com foco nos conceitos de Orientação a Objetos (POO). Os principais elementos do programa estão representados pelas classes e objetos a seguir:

Classes Principais:

- Habito: Representa um hábito, com atributos como id, nome, tempoMeta (em minutos) e dataCriacao.
- ProgressoDiario: Armazena o progresso diário de um hábito, com atributos como id, habitoId, data e tempoEstudo (em minutos).
- DatabaseHelper: Classe responsável pela criação e gerenciamento do banco de dados SQLite.
- HabitoRepository: Gerencia as operações de CRUD (Create, Read, Update, Delete) para a tabela de hábitos.
- ProgressoRepository: Gerencia as operações de CRUD para a tabela de progresso diário.

Tabelas do Banco de Dados:

- habitos: Armazena os hábitos criados pelos usuários.
- progresso: Armazena o tempo estudado diariamente para cada hábito.
- pomodoro: Armazena as configurações do Pomodoro.

Funcionalidades Principais

1. Adicionar, Editar e Excluir Hábitos

- Adicionar Hábito: O usuário pode adicionar um novo hábito, definindo um nome e uma meta de tempo diário.
- Editar Hábito: O usuário pode editar o nome ou a meta de tempo de um hábito existente.

- Excluir Hábito: O usuário pode excluir um hábito, removendo-o permanentemente do banco de dados.

2. Rastreamento de Progresso

- Iniciar Temporizador: O usuário pode iniciar um temporizador para registrar o tempo dedicado a um hábito.
- Pausar Temporizador: O temporizador pode ser pausado e retomado posteriormente.
- Atualizar Progresso: O tempo estudado é salvo no banco de dados e exibido em gráficos.

3. Gráficos de Progresso

- Gráfico de Barras: Exibe o tempo total estudado nos últimos 7 dias.
- Gráfico de Pizza: Exibe a distribuição do tempo estudado entre os hábitos no dia atual. Ele pode ser pressionado para expandir os dados e obter uma melhor visualização.

4. Seleção de Data

- O usuário pode selecionar uma data específica para visualizar o progresso registrado naquele dia.

5. Pomodoro

- O usuário pode definir o tempo de estudos e de pausa entre os ciclos de estudo.
- Algum hábito deve ser selecionado no spinner. Todo o tempo de estudo realizado vai ser adicionado à aquele hábito naquele dia em específico.

1.4 Funcionalidades das Classes e Métodos

HabitoRepository

- adicionarHabito: Adiciona um novo hábito ao banco de dados.
- getHabitos: Retorna a lista de hábitos cadastrados.
- updateHabito: Atualiza as informações de um hábito existente.
- deleteHabito: Remove um hábito do banco de dados.
- getIdDoHabito: Retorna o ID de um hábito com base no nome.

ProgressoRepository

- adicionarProgressso: Adiciona um registro de progresso diário.
- getProgressoDiario: Retorna o progresso de um hábito em uma data específica.
- atualizarProgresso: Atualiza o tempo estudado de um registro de progresso.

HabitosAdapter

- updateHabitos: Atualiza a lista de hábitos exibida no RecyclerView.
- bind: Configura a exibição de cada item na lista, incluindo o nome do hábito, a meta de tempo e o tempo estudado.

HabitosActivity

- iniciarTemporizador: Inicia o temporizador para registrar o tempo estudado.

- pausarTemporizador: Pausa o temporizador.
- pararTemporizador: Para o temporizador e salva o progresso no banco de dados.
- atualizarGrafico: Atualiza o gráfico de barras com o progresso dos últimos 7 dias.
- setupPieChartMinimalista: Configura o gráfico de pizza com a distribuição do tempo estudado no dia atual.

PomodoroActivity

- startTimer: Inicia o temporizador para registrar o tempo de foco ou pausa.
- pauseTimer: Pausa o temporizador.
- resetTimer: Reinicia o temporizador.
- carregarHabitos: Carrega os hábitos cadastrados no banco de dados e os exibe no Spinner. O habito escolhido terá o tempo de estudo realizado durante o pomodoro adicionado no progresso daquele hábito em específico.
- mostrarNotifiaçao: Exibe uma notificação para o usuário.
- obterDataAtual: Retorna a data atual no formato yyyy-MM-dd.

1.5 Decisões e Ajustes

- Temporizador: Foi implementado um temporizador para registrar o tempo estudado em tempo real. O tempo é salvo no banco de dados a cada segundo. Para testes, o tempo registrado é em segundos, mas em produção seria em minutos.
- Gráficos Interativos: Os gráficos de barras e pizza foram implementados utilizando a biblioteca MPAndroidChart, proporcionando uma visualização clara do progresso.
- Seleção de Data: A funcionalidade de seleção de data permite ao usuário visualizar o progresso em dias anteriores, mas bloqueia a adição de tempo em datas passadas.

2. Conclusão

O projeto foi desenvolvido com sucesso, implementando um aplicativo funcional para rastreamento de hábitos. Durante o desenvolvimento, foram enfrentados os seguintes desafios:

1. Gerenciamento do Temporizador: Inicialmente, houve dificuldades em garantir que o temporizador funcionasse corretamente ao alternar entre hábitos. A solução foi implementar um mapa (tempoDecorridoMap) para armazenar o tempo decorrido de cada hábito.
2. Atualização dos Gráficos: A atualização em tempo real dos gráficos exigiu ajustes na lógica de renderização e na sincronização com o banco de dados.
3. Integração com o Banco de Dados: Foram necessários ajustes para garantir que os dados fossem persistidos corretamente e recuperados de forma eficiente.

Como melhorias futuras, seria interessante:

- Implementar uma interface gráfica mais elaborada.
- Adicionar funcionalidades adicionais, como notificações e integração com calendários.
- Expandir o suporte a diferentes tipos de hábitos e metas.
- Implementar sistema de conquistas.