GIOVANNI ALEIXO DE SOUZA
RONNI APARECIDO FERNANDES DE OLIVEIRA

ATIVIDADE INTEGRADORA [AVA3]:

BANCO DE DADOS AVANÇADOS

Londrina
2023

GIOVANNI ALEIXO DE SOUZA
RONNI APARECIDO FERNANDES DE OLIVEIRA

ATIVIDADE INTEGRADORA [AVA3]:

BANCO DE DADOS AVANÇADOS

Atividade Integradora da materia de Banco de Dados Avançados apresentado ao Curso de Tecnólogo Análise e Desenvolvimento de Sistemas do Centro Universitário Filadélfia – UniFil

Orientador: Prof. Marcus Vinícius Santana Maziero

Londrina
2023

O banco de dados escolhido para será aplicado no proieto do aplicativo ‘Viajantes‘, projeto em que se baseia em desenvolver um aplicativo com o objetivo de facilitar o planejamento de viagens de carro, será o bando de dados Firebase Realtime Database, que armazena e sincroniza todos os dados em tempo real entre clientes.

O Firebase Realtime Database usa transações para atualizar várias entradas de dados ao mesmo tempo, o que garante a integridade dos dados.
Durante uma transação, o Firebase Realtime Database acaba bloqueando a leitura e gravação de dados em todas as referências envolvidas na transação para garantir que apenas uma transação atue em um conjunto de dados em determinado momento.
Para se iniciar uma transação é usado o método runTransaction do Firebase Realtime Database SDK. O método aceita um objeto Transaction.Handler que implementa a lógica de tratamento da transação.Com isso, você pode ler e escrever dados em várias referências do Firebase Realtime Database. Se várias transações estiverem tentando atualizar as mesmas entradas de dados simultaneamente, o Firebase Realtime Database tentará executá-las sequencialmente, garantindo que apenas uma transação atualize cada entrada de dados.
Se uma transação for bem-sucedida, o Firebase Realtime Database vai iniciar as atualizações na ordem em que foram feitas.
Com isso, o Firebase Realtime Database usa transações para garantir a integridade dos dados e evitar conflitos de atualização simultânea.

O Firebase Realtime Database trabalha em um ambiente distribuído e em tempo real, onde várias pessoas podem acessar e atualizar os dados simultaneamente. Devido a essa situação o banco de dados precisa lidar com a concorrência de informações para garantir a consistência dos dados.
O Firebase Realtime Database trata a concorrência de informações usando um modelo de dados inspirado em árvore, cada nó da árvore representa um caminho para um dado específico. Quando dois usuários tentam atualizar o mesmo nó simultaneamente, o Firebase Realtime Database usa as seguintes estratégias para lidar com a concorrência de informações.
Se por exemplo, dois usuários tentam atualizar o mesmo nó simultaneamente, a última atualização a ser gravada no banco de dados vai substituir todas as atualizações anteriores.
O Firebase Realtime Database também pode combinar as alterações de vários usuários em uma única atualização que preserva todas as alterações. Isso significa que as alterações são mantidas, em vez de apenas a última.
O Firebase Realtime Database também permite que os desenvolvedores implementem suas próprias estratégias de tratamento de concorrência de informações. Isso pode ser feito usando as transações, que garantem que uma série de atualizações sejam aplicadas como uma única operação atômica. As transações ajudam a evitar conflitos e garantem que as atualizações sejam aplicadas em uma ordem específica, independentemente da ordem em que foram recebidas pelos usuários.

Conforme já informado anteriormente, o Firebase Realtime Database é um banco de dados em tempo real baseado em nuvem que armazena e sincroniza dados em tempo real entre clientes. Para recuperar dados, os clientes podem usar várias opções de

consulta, como a leitura de uma única vez (once), a escuta de eventos (on) ou consultas complexas que permitem filtrar e ordenar dados.

- Once: A leitura de uma única vez permite que um cliente recupere dados do banco de dados em tempo real apenas uma vez, essa opção é útil para recuperar dados estáticos do banco de dados.
- On: A escuta de eventos permite que um cliente ouça as atualizações de dados em tempo real e responda a elas de acordo. Quando os dados são alterados no banco de dados, o Firebase Realtime Database vai notificar o cliente imediatamente, o que permite que os clientes atualizem a interface do usuário em tempo real e criem aplicativos colaborativos em tempo real.

O Firebase Realtime Database permite que os clientes realizem consultas complexas que permitem filtrar e ordenar dados com base em vários critérios. As consultas complexas são especialmente úteis para aplicativos que exibem grandes quantidades de dados e precisam de recursos de filtragem e ordenação.

O Firebase Realtime Database armazena os dados em um formato JSON, o que acaba sendo acessível por aplicativos de várias plataformas.

Aqui está como o Firebase Realtime Database trata cada um desses requisitos:

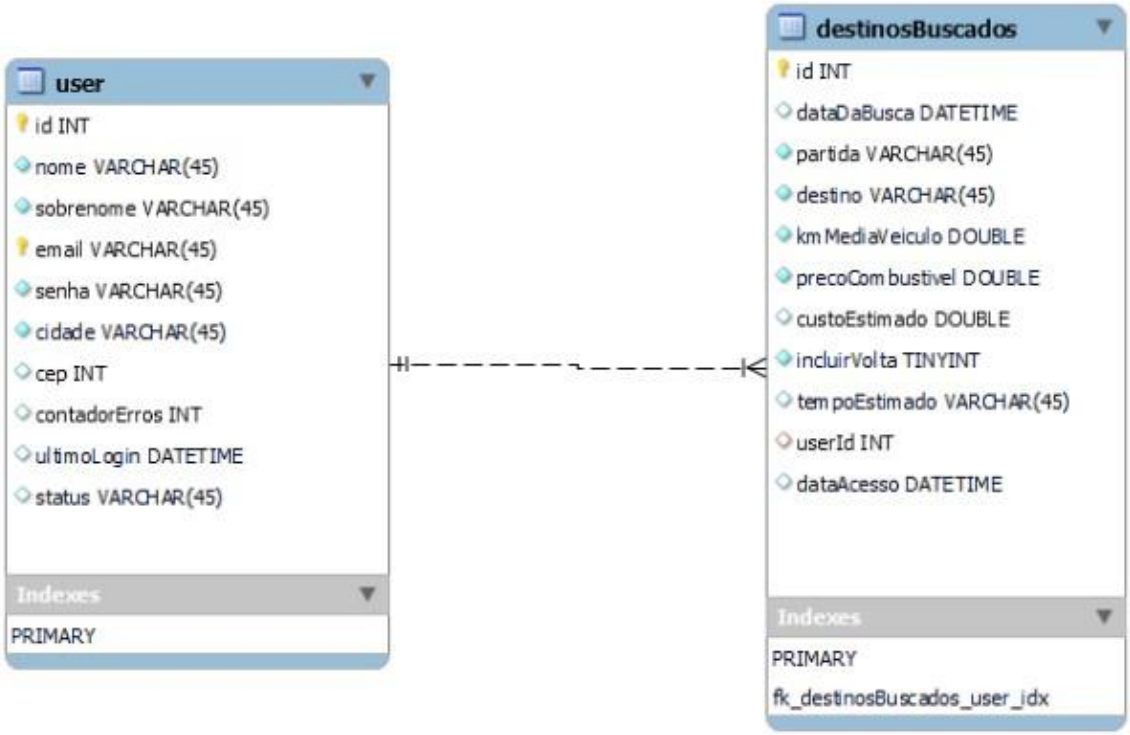
- Integridade: Usa transações para garantir a integridade dos dados. As transações garantem que uma série de atualizações seja aplicada como uma única operação atômica, garantindo que as atualizações sejam aplicadas em uma ordem específica e que não haja conflitos de atualização. O Firebase Realtime Database também mantém uma cópia atualizada dos dados em tempo real em todos os clientes conectados, garantindo que todos os usuários tenham acesso aos dados mais recentes.

- Segurança: O Firebase Realtime Database oferece várias opções de segurança para proteger os dados do banco de dados, Usando regras de segurança para definir quem pode ler e gravar nos dados do banco de dados, garantindo que apenas usuários autorizados possam acessar os dados. Também utiliza a autenticação do usuário para garantir que apenas usuários autenticados possam acessar o banco de dados. O Firebase Realtime Database fornece criptografia em repouso e em trânsito para garantir que os dados sejam protegidos durante o armazenamento e a transmissão.

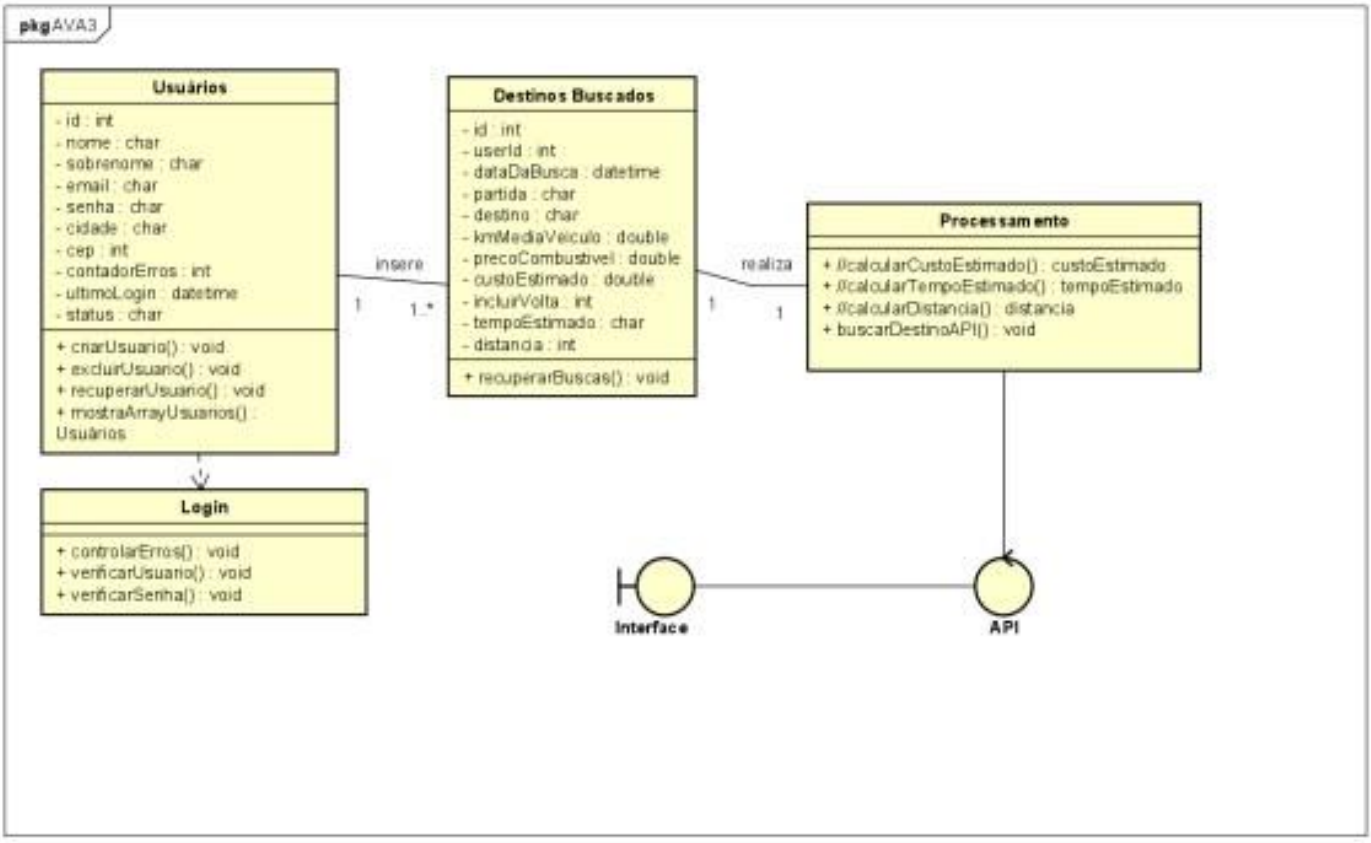
- Velocidade: O Firebase Realtime Database é otimizado para fornecer alta velocidade de acesso aos dados em tempo real. Ele usa uma arquitetura distribuída para garantir que os dados sejam acessíveis rapidamente de qualquer lugar do mundo. Também utiliza caching inteligente para reduzir o tempo de resposta das solicitações de dados e oferece suporte para consultas e indexação para permitir que os dados sejam acessados e filtrados rapidamente.

Resumidamente o Firebase Realtime Database tem um grande destaque pela sua facilidade de uso, escalabilidade, eficiência e confiabilidade. Ele oferece uma solução simples para aplicativos que precisam de dados em tempo real e é uma escolha popular para desenvolvedores devido à sua integração com outras ferramentas do Firebase, como autenticação, hospedagem, funções em nuvem e muito mais.

A figura a seguir mostra o diagrama de entidade e relacionamento do projeto:



A figura a seguir mostra o diagrama de classes (inicial) do projeto:



Classe Java (1°versão)

```
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.util.Scanner;

public class Login {

    public static void main(String[] args) {

        String[][] Usuarios = { //ID, NOME , SOBRENOME, EMAIL, SENHA, CIDADE, CEP, CONTADOR ERROS, ULTIMO LOGIN, STATUS
                ("221454113", "Douglas", "Sanches", "douglsanches@gmail.com", "456", "Londrina", "80010010", "0", "10/09/2022", "Desbloqueado"),
                ("221454060", "Felipe", "Kunioka", "felipematsuda17@gmail.com", "789", "Londrina", "80010010", "0", "10/09/2022", "Desbloqueado"),
                ("222454038", "Giovanni", "Souza", "giovanni_aleixo@edu.unifil.br", "abc", "Londrina", "80010010", "0", "10/09/2022", "Desbloqueado"),
                ("222454140", "Ronni", "Oliveira", "ronni.fernandes@uol.com.br", "123", "Apucarana", "86810390", "0", "10/09/2022", "Desbloqueado"));

        Scanner entrada = new Scanner(System.in);
        boolean usuarioEncontrado = false;
        int erro = 0;

        String dataHoraAtual = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(Calendar.getInstance().getTime());

        System.out.print("\nDigite o usuario (MATRICULA UNIFIL): ");
        String usuarioDigitado = entrada.next();
        verificarUsuario(Usuarios, entrada, usuarioEncontrado, usuarioDigitado, erro, dataHoraAtual);
    }

    private static void verificarUsuario(String[][] Usuarios, Scanner entrada, boolean usuarioEncontrado,
                                         String usuarioDigitado, int erro, String dataHoraAtual) {
        for (int i = 0; i < Usuarios.length; i++) {
            String usuario = Usuarios[i][0];
            String senha = Usuarios[i][4];

            //SE O USUARIO FOR ENCONTRADO
            if (usuario.equals(usuarioDigitado)) {
                usuarioEncontrado = true;
                Usuarios[i][8] = dataHoraAtual;
                System.out.println("\nBem vindo " + Usuarios[i][1]);
                System.out.println(Usuarios[i][8]);
                System.out.print("\nDigite a senha: ");
                String senhaDigitada = entrada.next();
                verificarSenha(senha, senhaDigitada, erro, Usuarios, i);

            }
        }
        // APÓS PERCORRER O ARRAY, SE O USUARIO NÃO FOR ENCONTRADO
        if (usuarioEncontrado == false)
            System.out.println("Usuario " + usuarioDigitado + " não encontrado");
    }

    private static void verificarSenha(String Senha, String senhaDigitada, int erro, String[][] Usuarios, int i) {
        boolean senhaValida = false;
        String senha = Senha;

        //VERIFICA SE A SENHA É VALIDA OU INVALIDA
        senhaValida = isSenhaValida(senhaDigitada, senhaValida, senha, Usuarios, i);
        isSenhaInvalida(senha, erro, senhaValida, Usuarios, i);
    }

    private static boolean isSenhaValida(String senhaDigitada, boolean senhaValida, String senha, String[][] Usuarios, int i) {
        // SE A SENHA FOR VÁLIDA
        if (senha.equals(senhaDigitada)) {
            senhaValida = true;
            Usuarios[i][9] = "Ativo";
            System.out.println("Acesso concedido.\n\n");
            mostrarArrayUsuarios(Usuarios);
        }
        return senhaValida;
    }
```

```
    private static void isSenhaInvalida(String senha, int erro, boolean senhaValida, String[][] Usuarios, int i) {
        //SE A SENHA FOR INVALIDA
        if (senhaValida == false) {
            System.out.println("Senha incorreta");
            erro++;
            Usuarios[i][7] = Integer.toString(erro); // CONTAGEM DE ERROS ATRIBUIDA AO VETOR

            //VERIFICA AS TENTATIVAS ERRADAS
            if (erro < 3) {
                System.out.println("\n" + (erro + 1) + "ª tentativa");
                System.out.print("Digite a senha novamente: ");
                Scanner entrada = new Scanner(System.in);
                String senhaDigitada = entrada.next();
                verificarSenha(senha, senhaDigitada, erro, Usuarios, i);
            } else {
                System.out.println("Número de tentativas excedidas");
                System.out.println("Usuário bloqueado");
                Usuarios[i][9] = "Bloqueado"; // ALTERA O STATUS DO USUARIO PARA "Bloqueado"
                mostrarArrayUsuarios(Usuarios); // MOSTRA O ARRAY ALTERADO
            }
        }
    }


    private static void mostrarArrayUsuarios(String[][] Usuarios) {
        System.out.println("\nMostrando o Array USUARIOS. " +
                "\nVerifique as alterações nos dados do ultimo login, status do usuário, bem como a contagem de erros, se for o caso:");
        System.out.println("\nID          | NOME | SOBRENOME | EMAIL | SENHA | CIDADE | CEP | CONTADOR ERROS | ULTIMO LOGIN | STATUS
USUARIO");
        for (int i = 0; i < Usuarios.length; i++) {
            for (int j = 0; j < Usuarios[0].length; j++) {
                System.out.print(Usuarios[i][j] + " | ");
            }
            System.out.println("");
        }
    }
}
```

Os diagramas de entidade e relacionamento, diagrama de classes e Classe Java (1°versão) foram desenvolvimentos no início do projeto, porém com o decorrer do desenvolvimento foi realizado alterações de ideias e por causa disso foi migrado a utilização de banco de dados da aplicação de SQL para o Firebase Realtime Database, que é um banco NOSQL.

Referências

Material do curso EAD – Banco de Dados Avançados UNIFIL EAD. Acesso em 20 fev. 2023

Material do curso EAD – Linguagem de Programação Orientada a Objetos UNIFIL EAD. Acesso em 22 fev. 2023

Material do curso EAD – Introdução à Banco de Dados UNIFIL EAD. Acesso em 15 março. 2023

Material do curso EAD – Algoritmos e Estrutura de Dados UNIFIL EAD. Acesso em 15 março. 2023

Material do curso EAD – Análise e Projeto Orientado a Objetos UNIFIL EAD. Acesso em 15 março. 2023

Material do curso EAD – Desenvolvimento Web UNIFIL EAD. Acesso em 15 março. 2023

Material do curso EAD – Processo de Desenvolvimento de Software UNIFIL EAD. Acesso em 15 março. 2023

Material do curso EAD – Lógica de Programação Orientada a Objetos UNIFIL EAD. Acesso em 15 março. 2023

Firebase Realtime Database, Disponível em: https://firebase.google.com/docs/database?hl=pt-br

Realtime Database, Disponível em: https://rnfirebase.io/database/usage

Entendendo o Firebase e suas principais funcionalidades, Disponível em: https://www.alura.com.br/artigos/entendendo-firebase-principaisfuncionalidades?gclid=Cj0KCQjw2v-gBhC1ARIsAOQdKY3yCLWsZ4oqyklIKtJwZl4DYHiKT7bKwTlW81zDXnS45rvmx_s_H78aAo_HEALw_wcB