

## Resumo técnico (visão geral)

- **Tipo:** Aplicação desktop Windows Forms em C# (.NET Framework 4.7.2).
- **Projeto principal:** ReservaDeViagens (solução em ReservaDeViagens.sln).
- **Entrada:** Program.cs — instancia e mostra InicioForm e chama Application.Run() (inicia o loop de mensagens sem passar uma Form principal).
- **Propósito:** Sistema de reserva/gestão para agência de viagens com autenticação (inclui login por password e reconhecimento facial), CRUD para clientes/funcionários, gestão de reservas (viagens/hotel/passagem), logs de acesso e geração de relatórios (PDF).

## Stack e dependências principais

- **Framework:** .NET Framework 4.7.2 (TargetFrameworkVersion no ReservaDeViagens.csproj).
- **UI:** Windows Forms (System.Windows.Forms).
- **BD:** MySQL via MySql.Data (v9.5.0). Querys e operações são feitas diretamente nas Forms (ex.: ViagemForm.cs, LoginForm.cs, CadastroForm.cs). Arquivo SQL de esquema: BD Agencia de Viagens.sql.
- **Visão computacional / Face recognition:** Emgu CV (wrapper .NET para OpenCV) — pacotes Emgu.CV, Emgu.CV.Bitmap, Emgu.CV.runtime.windows. Cascade XML incluído: Resources\haarcascade\_frontalface\_default.xml.
- **Reconhecimento:** LBPH Face Recognizer (classe LBPHFaceRecognizer) — treino com imagens carregadas da BD (RostoFuncionario), predição com threshold (ex.: distância < 60).
- **Armazenamento de rosto:** rostos são salvos como blobs JPEG em RostoFuncionario.RostoLbp (inserção via MySqlDbType.LongBlob).
- **PDF/Relatórios:** iTextSharp (geração de PDFs).
- **Outros:** BouncyCastle (crypto lib), várias libs de suporte (System.\* packages, compressão, etc.) configuradas no packages.config e csproj.

## Arquitetura / Fluxos importantes

- **Arranque:** Program.Main() mostra InicioForm e entra no loop da aplicação.
- **Autenticação:** existe LoginForm (username/password) e LerRostoForm (reconhecimento facial). Em LerRostoForm, quando o reconhecedor encontra uma correspondência válida:
  - regista um log em log\_acessos (INSERT com NOW()),
  - obtém LAST\_INSERT\_ID() para idLogAtual,
  - abre MenuForm com funcionarioId e idLogAtual.
- **Cadastro de rosto:** CadastrarRostoForm usa VideoCapture para capturar vídeo, detecta faces com CascadeClassifier e guarda a imagem do rosto (patch) na BD como blob JPEG.

- **Treino do reconhecedor:** na inicialização de LerRostoForm chama CarregarRostosDoBanco() que:
  - lê blobs dos rostos,
  - converte para Bitmap -> Mat -> grayscale,
  - adiciona à lista de treinamento e labels,
  - treina LBPHFaceRecognizer.
- **Operações DB:** múltiplas Forms usam using (SqlConnection conn = new MySqlConnection(data\_source)) com string de conexão embutida em cada class (datasource=localhost;username=root;password=;database=tripagency).

#### Detalhes importantes encontrados no código

- faceDetector = new CascadeClassifier(xmlPath) e DetectMultiScale(...) para detecção.
- Captura de vídeo usando Emgu.CV.VideoCapture e processado em Application.Idle.
- Reconhecimento com LBPHFaceRecognizer e threshold de distância (por ex. 60).
- Rosto salvo como LongBlob no MySQL; treino usa VectorOfMat + Mat labelsMat.
- App.config contém binding redirects; porém a string de conexão está hardcoded nos forms (não no App.config).

#### Build / Run (passos práticos)

- Requisitos:
  - Visual Studio (2017/2019/2022) com suporte .NET Framework 4.7.2.
  - MySQL Server com base de dados tripagency (importar BD Agencia de Viagens.sql).
  - Restaurar packages NuGet (ou garantir pasta packages existente).
  - Emgu CV native runtime (pacote Emgu.CV.runtime.windows) — assegurar que os binários nativos estejam presentes para execução.
- Comandos (PowerShell) — abrir solução e compilar via MSBuild/Visual Studio:

*# abrir solução no Visual Studio (GUI)*

explorer "ReservaDeViagens.sln"

*# ou compilar via MSBuild (exemplo)*

& "C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Current\Bin\MSBuild.exe" .\ReservaDeViagens.sln /p:Configuration=Debug

- Verificações:

- Resources\haarcascade\_frontalface\_default.xml deve estar copiado para o output (é configurado no csproj para PreserveNewest).
- Ajustar a string de conexão (ou criar a BD/usuário root sem password) — alterar as variáveis data\_source nas forms para apontar para o servidor correto.

### Riscos e recomendações rápidas

- **Segurança:** credenciais de BD hardcoded e sem password (muito inseguro). Recomendado mover a connection string para App.config (<connectionStrings>) e usar credenciais seguras / secrets.
- **Manutenção:** lógica de DB espalhada por Forms. Recomendado refatorar para camada de acesso a dados (DAL) para centralizar conexões e queries.
- **Nativo EmguCV:** tenha atenção ao target platform (prefer x64) e garantir que os binários nativos estejam na pasta de saída; csproj já importa runtime targets do pacote.
- **Robustez:** verificar tratamento de exceções em operações de I/O/câmera e proteger conversões de blob; validar entradas do utilizador; evitar usar AddWithValue sem considerar tipos.
- **UX/Loop:** Program usa inicio.Show() + Application.Run() (sem forma “main”) — válido, mas é menos explícito; pode querer usar Application.Run(new InicioForm()) para um ciclo de vida mais claro (fecha app ao fechar a form principal).

Quer que eu:

- gere um ficheiro README técnico organizado com instruções de build/run e dependências?
- refatore rapidamente a gestão da connection string para App.config e faça um patch no repositório?
- ou extraia um diagrama/fluxograma do fluxo de autenticação (login por senha vs. reconhecimento facial)?

jeeloa: Explique me o fluxo do programa, como é que ele funciona

## Visão Geral

- **Tipo:** Aplicação desktop Windows Forms em C# (.NET Framework 4.7.2).
- **Entrada:** Program.Main() (arquivo Program.cs) — cria e mostra InicioForm e chama Application.Run() para iniciar o loop de mensagens.
- **Objetivo principal:** Gestão de uma agência de viagens: autenticação (senha + reconhecimento facial), cadastro de funcionários/clientes, criação/visualização de reservas, logs de acesso e geração de relatórios (PDF).

## Fluxo de arranque e navegação

- **Arranque:** App inicia → InicioForm é exibido (ponto de entrada para o utilizador).
- **A partir de InicioForm:** o utilizador pode abrir:
  - LoginForm — autenticação por username/password,
  - LerRostoForm — autenticação por reconhecimento facial,
  - formulários de cadastro/reserva (CadastroForm, ViagemForm, etc.).

## Autenticação — por senha (LoginForm)

- LoginForm lê os campos de username/password e executa queries MySQL (SqlCommand) para validar credenciais contra a tabela funcionarios (ex.: SELECT ... FROM funcionarios WHERE ...).
- Se credenciais válidas: insere um registo em log\_acessos (registro do login), recupera o ID do log (LAST\_INSERT\_ID()) e abre MenuForm, passando o funcionariold (e possivelmente idLog) para o menu principal.
- As queries e a string de conexão são feitas diretamente na Form (cadeia data\_source hardcoded).

## Autenticação — reconhecimento facial (LerRostoForm)

- Inicialização (LerRostoForm\_Load):
  - Carrega haarcascade\_frontalface\_default.xml (em Resources/output).
  - Chama CarregarRostosDoBanco() para ler blobs de rostos da tabela RostoFuncionario (e nomes em funcionarios).
  - Converte cada blob JPEG para Bitmap → Mat → grayscale e popula rostosTreinados + labels.
  - Cria e treina um LBPHFaceRecognizer com VectorOfMat e um labelsMat.
- Captura:
  - Ao clicar btnIniciar cria VideoCapture(0) e regista o handler Application.Idle += ProcessarFrame.

- Em ProcessarFrame: captura frame, converte para grayscale, detecta faces com CascadeClassifier.DetectMultiScale.
- Para cada face detectada: extrai patch (gray.Copy(face)), chama reconhecedor.Predict(faceMat).
- Se result.Label != -1 e result.Distance < 60 (threshold definido no código):
  - considera como utilizador reconhecido → obtém funcionariold do label.
  - chama RegistrarLogin(funcionariold) que insere um registo em log\_acessos e guarda idLogAtual com LAST\_INSERT\_ID().
  - pára a captura (Application.Idle -= ProcessarFrame, capture.Stop()), abre MenuForm(funcionariold, idLogAtual) e fecha LerRostoForm.
- Se não reconhece, continua mostrando "Desconhecido" no labelReconhecido.

#### **Cadastro de rosto (CadastrarRostoForm)**

- Carrega haarcascade\_frontalface\_default.xml no Load.
- Ao iniciar captura (btnIniciar) usa VideoCapture e Application.Idle += ProcessarFrame para mostrar feed.
- Ao clicar btnCapturar: detecta faces no currentFrame, pega o primeiro rosto detectado, converte para JPEG em MemoryStream e guarda na tabela RostoFuncionario (Funcionariold, RostoLbp) como LongBlob.
- Depois de inserir, mostra confirmação.

#### **Fluxo de reservas e CRUD**

- ViagemForm, verReservaForm, ResumoReservaForm e outros formulários executam queries MySQL para:
  - listar hotéis, companhias, passagens, etc.;
  - inserir reservas (INSERT), obter IDs relacionados e mostrar sumários;
  - gerar PDFs (usa iTextSharp) — presumivelmente em ResumoReservaForm/relatórios.
- AdminClientForm, AdminFuncionarioForm e CadastroForm fazem a gestão (CRUD) de clientes/funcionários com queries diretas.
- Muitas Forms usam a mesma string de ligação data\_source embutida: datasource=localhost;username=root;password=;database=tripagency.

#### **Interação com a base de dados (pontos chave)**

- Library: MySql.Data (MySQL Connector/NET).
- Tabelas observadas no código:
  - funcionarios (id, Nome, Username, ...),

- RostoFuncionario (FuncionarioId, RostoLbp blob),
- log\_acessos (funcionario\_id, username, data\_login, e possivelmente data\_logout),
- tabelas para reservas/hotel/companhia (consultar BD Agencia de Viagens.sql para esquema completo).
- Padrão: cada Form abre a sua MySqlConnection com using, executa MySqlCommand e fecha; nem sempre há camada de acesso a dados centralizada.

### Gestão de recursos e robustez

- Câmera: VideoCapture é iniciada/parada, e Application.Idle é usado para processar frames (padrão simples mas pode ter implicações de performance).
- Arquivos nativos EmguCV: o csproj importa runtime targets do pacote Emgu.CV.runtime.windows; é preciso garantir os binários nativos no output para execução.
- App.config contém redirects de assembly; porém a connection string está hardcoded nas classes em vez de App.config.

### Exemplo sequencial (passo a passo)

- Login via rosto:
  1. Abrir app → InicioForm → escolher autenticação facial.
  2. LerRostoForm carrega cascade e rostos do BD; treina LBPHFaceRecognizer.
  3. Iniciar câmera → frames processados por ProcessarFrame.
  4. Detecção → predição → se distância < 60 e label != -1  
→ RegistrarLogin insere log\_acessos.
  5. Ler LAST\_INSERT\_ID() → abrir MenuForm com funcionarioId e idLogAtual.
- Cadastro de rosto:
  1. CadastrarRostoForm → iniciar câmera → detectar rosto.
  2. Capturar → extrair patch → converter para JPEG → inserir na tabela RostoFuncionario como blob.
  3. O rosto ficará disponível para treino em LerRostoForm no próximo arranque.

### Principais pontos de atenção / recomendações rápidas

- **Segurança:** connection string e credenciais hardcoded (e password vazio) — mover para App.config e usar credenciais seguras.
- **Arquitetura:** separar lógica de acesso a dados numa camada (DAL/Repository) em vez de queries espalhadas nas Forms.
- **EmguCV / runtimes nativos:** garantir pacotes Emgu.CV.runtime.windows e escolher target platform (x64 vs x86) consistente com os bins nativos.

- **Threshold do LBPH:** o valor 60 pode necessitar de afinamento para reduzir falsos positivos/negativos.
- **Robustez:** melhorar tratamento de exceções, timeouts na BD, e liberação de recursos da câmera (especialmente em erros).
- **UX:** considerar feedback ao utilizador (ex.: contadores/tempo de tentativa) e fluxo claro ao falhar reconhecimento.

Quer que eu faça uma dessas ações agora:

- gerar um README\_tech.md com este fluxo e instruções de build/run;
- mover a connection string para App.config e aplicar um patch simples para usar ConfigurationManager;
- ou desenhar um diagrama sequencial mais detalhado (texto) do login por rosto?

Certo! Vou explicar de forma simples para um iniciante entender. Imagine que o programa é como um jogo:

### 1. O ARRANQUE (Beginning)

App inicia → Mostra a "tela inicial" (InicioForm)

O utilizador vê um ecrã com opções: "Login com senha", "Login com rosto", "Cadastro", etc.

---

### 2. AUTENTICAÇÃO POR SENHA (Password Login)

Utilizador clica → "Login com senha"



Abre formulário com 2 campos: username e password



Utilizador escreve e clica "Entrar"



O programa pergunta ao banco de dados:

"Existe alguém com este username e password?"



Se SIM → "Bem-vindo! Entrando no menu..."

Regista a hora de entrada (log) no banco

Abre o Menu Principal

Se NÃO → "Dados errados, tenta novamente"

---

### 3. AUTENTICAÇÃO POR ROSTO (Face Recognition)

É mais interessante! Aqui é como um "sistema de segurança":

Utilizador clica → "Login com rosto"



O programa carrega TODOS os rostos dos funcionários

que já foram guardados no banco de dados



Ligação da câmara (webcam)





O programa vê a tua cara e diz:

"Espera... estou a comparar a tua cara com  
as caras que conheço..."



Se encontra correspondência (e é suficientemente parecida):

"Ah! Tens uma cara conhecida! Bem-vindo!"

Regista a hora de entrada

Abre o Menu Principal

Se não encontra:

"Não reconheço esta cara..."

Continua a tentar

---

#### **4. CADASTRO DE ROSTO (Face Registration)**

Antes de poder fazer login com rosto, o funcionário precisa guardar a sua cara:

Utilizador abre → "Cadastrar rosto"



Câmera ligada



O programa deteta caras no vídeo

(desenha um quadrado verde à volta da cara)



Utilizador clica → "Capturar"



O programa guarda uma "foto da cara"

no banco de dados (como se fosse uma impressão digital)



"Rosto guardado! Agora podes fazer login com ele"

---

#### **5. FAZER UMA RESERVA (Booking)**

Depois de fazer login, o utilizador entra no Menu:

Menu Principal



Utilizador clica → "Fazer Reserva"



Abre formulário:

- Escolher Destino (viagem)
- Escolher Hotel
- Escolher Companhia Aérea (passagem)
- Preencher datas, número de pessoas, etc.



Utilizador clica → "Confirmar Reserva"



O programa guarda tudo no banco de dados



Mostra um resumo:

"Reserva confirmada!"

Hotel XYZ, Passagem ABC, de 10 a 15 de Dezembro"

---

## 6. VER HISTÓRICO (Check Bookings & Access Logs)

Menu Principal



Utilizador clica → "Ver minhas reservas"

ou "Ver histórico de acessos"



O programa pergunta ao banco:

"Quais são as reservas/acessos deste utilizador?"



Mostra uma lista:

- Data 24/Nov, Hotel Porto
- Data 23/Nov, Hotel Lisboa
- etc.

---

## 7. FUNÇÕES DE ADMINISTRADOR

Se o utilizador é admin (tem permissão especial):

Menu Admin



Pode:

- Ver todos os clientes e funcionários
- Editar/apagar dados de pessoas
- Ver todos os logs de acesso (quem entrou quando)
- Gerar relatórios/PDFs

---

## RESUMO VISUAL DO FLUXO GERAL

APP INICIA



TELA INICIAL

- └--> Login por Senha → Menu (se credenciais corretas)
- └--> Login por Rosto → Menu (se rosto reconhecido)
- └--> Cadastro Rosto → Guarda cara no banco
- └--> Cadastro Cliente → Adiciona nova pessoa

MENU PRINCIPAL (depois de login)

- └--> Fazer Reserva → Escolher viagem/hotel/passagem
- └--> Ver Reservas → Lista as minhas reservas
- └--> Ver Histórico → Ver quando entrei
- └--> Sair → Desconectar

---

## ANALOGIA SIMPLES

Pensa no programa como um **cinema com biometria**:

1. **Entrada:** Tu chegas ao cinema. Podes entrar mostrando o teu bilhete (senha) ou mostrando a tua cara registada (face recognition).
2. **Primeira vez:** Se é a primeira vez com rosto, vais à "receção" e registam a tua cara (cadastro de rosto).

3. **Dentro:** Já dentro do cinema, podes comprar bilhetes para filmes (fazer reservas).
4. **Segurança:** O cinema regista quando entraste e saíste (logs de acesso).
5. **Gerente:** O gerente pode ver todos os clientes, as reservas, etc. (admin).

Faz sentido agora? 😊



Ótimo! Vou explicar o reconhecimento facial em linguagem simples:

### **1. PREPARAÇÃO (Setup)**

Antes de o programa conseguir reconhecer rostos, ele precisa de:

a) Um arquivo XML especial (haarcascade\_frontalface\_default.xml)

→ É como um "livro de instruções" que ensina o programa  
a reconhecer o que é uma cara

b) Uma lista de rostos conhecidos guardados no banco de dados

→ Todas as caras dos funcionários que já se registaram

---

### **2. TREINAMENTO (Learning Phase)**

Quando o programa arranca, ele faz isto:

PASSO 1: Ir ao banco de dados

"Dá-me todos os rostos guardados"

PASSO 2: Ler cada rosto guardado

(estão guardados como "fotos" no banco)

PASSO 3: Converter para preto e branco (grayscale)

(mais fácil de processar)

PASSO 4: Ensinar o programa a reconhecer

"Esta é a cara do João"

"Esta é a cara da Maria"

"Esta é a cara do Pedro"

(Isto é feito por um algoritmo chamado LBPH -

Local Binary Patterns Histograms -

mas é só uma "máquina de aprender" padrão)

---

### **3. RECONHECIMENTO EM TEMPO REAL (Live Recognition)**

Agora, quando o utilizador faz login com rosto:

PASSO 1: Câmera ligada

O programa vê o vídeo em direto

PASSO 2: Para cada frame (fotograma) do vídeo:

a) Detecta caras

(usa aquele arquivo XML para encontrar  
retângulos onde há caras)

b) Para cada cara detectada:

- Extrai apenas o rosto (corta a imagem)
- Converte para preto e branco

c) Compara com os rostos conhecidos

"Isto é parecido com a cara do João?"

"Isto é parecido com a cara da Maria?"

d) Calcula uma "pontuação de semelhança"

- Se for MUITO parecido = baixa pontuação (ex: 30)
- Se for POUCO parecido = alta pontuação (ex: 100)

e) Verifica se a pontuação é boa o suficiente

"Se pontuação < 60, é alguém que conhecemos"

PASSO 3: Se encontra correspondência:

"Encontrei! Esta é a cara do João!"

Login automático

PASSO 4: Se não encontra:

"Não reconheço... continua a tentar"

(mostra "Desconhecido" no ecrã)

---

#### 4. ANALOGIA - O GUARDA DE SEGURANÇA

Imagina um guarda de segurança numa discoteca:

TREINAMENTO:

- O gerente dá uma foto de cada VIP ao guarda
- "Este é o João", "Esta é a Maria", etc.
- O guarda memoriza as caras

NOITE DO EVENTO:

- Pessoas chegam à porta
- O guarda olha para cada pessoa
- "Hmm... esta cara é parecida com a do João! 70% de certeza"

Se a semelhança é MUITO alta (tipo 95% de certeza):

"Bem-vindo João! Entra!"



Se a semelhança é baixa (tipo 20% de certeza):

"Não te reconheço, desculpa"

Se é meio-termo (tipo 60%):

"Não sei, precisas de mostrar o bilhete"

No programa, aquele número **60** é o "limite de aceitação":

- Semelhança **menor que 60** = É alguém conhecido 
- Semelhança **maior que 60** = Pode ser impostor 

---

#### 5. POR QUE FUNCIONA ASSIM?

Problema: Como reconhecer uma cara?

- As caras mudam (expressões, ângulos, iluminação)
- Não dá para comparar pixel por pixel




Solução: Usar padrões

- O LBPH analisa "padrões" na cara  
(como as sombras, as linhas, as texturas)
- Guarda esses padrões como um "resumo"  
(tipo uma impressão digital, mas da cara)
- Depois compara resumos, não imagens inteiras


---

## 6. EXEMPLOS PRÁCTICOS

SUCESSO:

- João registou a sua cara
- Próximo dia, João faz login com rosto
- Câmera vê a cara dele
- Programa: "Isto é muito parecido com a cara  
do João (semelhança = 45)"
- $45 < 60$ ? SIM → Login bem-sucedido! 

FALHA - Cara desconhecida:

- Uma pessoa que não está registada tenta entrar
- Programa: "Isto não é parecido com ninguém  
(semelhança = 95)"
- $95 < 60$ ? NÃO → Acesso negado 

FALHA - Luz fraca:

- João tenta, mas está muito escuro
- Programa não consegue ver bem
- Não encontra padrões = não reconhece
- Mostra: "Desconhecido" e continua a tentar

---

## 7. O FLUXO COMPLETO VISUAL

PRIMEIRA VEZ (Cadastro de Rosto):

\_\_\_\_\_

```
| Funcionário abre câmera      |  
| Sistema detecta face        |  
| Guarda a cara no banco de dados |  
| "Rosto registado!"          |  
|_____|
```

↓

PRÓXIMAS VEZES (Login com Rosto):

```
|_____|  
| Sistema treina: carrega todos os |  
| rostos do banco e aprende as padrões |  
|_____|
```

↓

```
|_____|  
| Câmera ligada                |  
| Detecta: "Vejo uma cara aqui!" |  
| Compara: "Isto é parecido com..." |  
| Calcula: "Semelhança = 45"      |  
| Verifica: "45 < 60? SIM!"       |  
| Resultado: "Bem-vindo João!"   |  
|_____|
```

---

### RESUMO EM 3 LINHAS

1. **Treino:** Guarda as caras dos funcionários com padrões especiais
2. **Captura:** Câmera vê uma cara nova
3. **Comparação:** Programa diz "isto é parecido com qual cara conhecida?" e se a semelhança for boa o suficiente (< 60), faz login automático

Ficou claro? 😊 Quer que explique como é que o programa **guarda** as caras no banco (a parte de "capturar e salvar")?