**SAF**

**Programa Gestão**

**P R O J E T O J A V A** ( 50 HORAS )

**Roberto Carvalho** Nº 14

**Serviços Fúnebres**

Serviço Assistente Funerária



Í N D I C E

[P r o j e t o 3](#_Toc165288881)

[Título e Objetivo 3](#_Toc165288882)

[Autor, data, contexto do trabalho 3](#_Toc165288883)

A black and grey acorn with a flower

Description automatically generated[Descrição do problema 3](#_Toc165288884)

[Ferramentas e versões 3](#_Toc165288885)

[Análise de dados 3](#_Toc165288886)

[Descrição de processos 4](#_Toc165288887)

[“String connection”, instâncias, bases de dados 6](#_Toc165288888)

[Lista de tabelas, campos e tipos 7](#_Toc165288889)

[Namespaces 7](#_Toc165288890)

[Classes e métodos 8](#_Toc165288891)

[Outras informações 8](#_Toc165288892)

# P r o j e t o

## Título e Objetivo

O nome do software será: **SAF** ( Serviço Assistente Funerária ).

O objetivo deste software será agilizar o ato de registar e procurar informações relacionadas com serviços aplicados a defuntos.

## Autor, data, contexto do trabalho

Este programa será criado por Roberto Filipe Rocha Carvalho.

A partir do dia 29 de abril 2024.

Com o intuito de servir para pequenas ou médias empresas dentro da área fúnebre.

## Descrição do problema

Registar em formato papel não é prático e em certos casos, é demoroso.  
Este programa visa agilizar o processo e permitir adicionalmente retificar erros.

## Ferramentas e versões

* Apache NetBeans IDE 15
* MySql (v. 8.0)
* Microsoft SQL Server Management Studio (v. 19.1)

## Análise de dados

A análise de dados envolve a identificação das principais informações necessárias para realizar o registo de defuntos, familiares e serviços aplicados. Além das funcionalidades extras que seriam pertinentes o programa fornecer.

## Descrição de processos

**(opcional) Registo de Utilizadores**

Registo de dados sobre os utilizadores.   
Numa fase inicial presumo algo como:

* Nome
* Senha
* Preferência de idioma
* Entre outros

Adicionalmente, se possível, implementar uma opção de ser possível associar uma imagem ao falecido, que ficará alojada dentro de uma pasta interna com o nome “falecido”.

**Registo de Defuntos**

Registo de dados sobre o falecido.   
Numa fase inicial presumo algo como:

* Nome
* Data de falecimento
* Causa da morte
* Entre outros

Adicionalmente, se possível, implementar uma opção de ser possível associar uma imagem ao falecido, que ficará alojada dentro de uma pasta interna com o nome “falecido”.

**Registo de Familiares**

Registo dos dados do relacionado com o falecido.

Numa fase inicial presumo algo como:

* Nome
* Relação
* Endereço;
* Contacto
* Entre outros

Adicionalmente, se possível, implementar uma opção de ser possível associar uma imagem ao familiar do falecido, que ficará alojada dentro de uma pasta interna com o nome “Familiar”.

**Serviços Fúnebres**

Registo dos dados relacionados com o serviço contratado pelo familiar do falecido.

Numa fase inicial presumo algo como:

* Preparação do corpo
* Velório
* Transporte
* Entre outros

**Filtragem e Procura**

Permitir filtrar dados de múltiplas formas de forma a encontrar o que é pretendido de forma rápida.

Estes processos foram desenvolvidos com o objetivo de facilitar e agilizar o registo e a gestão de informações relacionadas, garantindo um atendimento eficiente e respeitoso durante este momento delicado.

## “String Connection”, instâncias, bases de dados

**Servidor**  
Lagostim (MySQL)

**URL**   
62.28.39.1353306

**Username**  
Raquel

**Password**  
Silva1234

**Base de dados**

safJava

## Lista de tabelas, campos e tipos

|  |  |  |
| --- | --- | --- |
| **Family** Id : int Name : string Contact : string  Address : string Relation : string  (opcional) Photo : string |  | **Deceased** Id : int Name : string Age : int Sex : char  Comment : string  (opcional) Photo : string |
|  |  |  |
| **Service** Id : int FamilyID : int DeceasedID : int  UserID : int  commnent : string  Date : date  Price : decimal CheckPay : tinyInt (bool) |  | **(opcional) User**  Id : int Name : string Password : string Admin : tinyInt (bool)  Theme: int  Language : string  Photo : string |

## Namespaces

safJava  
safJava.Business  
safJava.Persistence  
safJava.Views

## Classes e métodos

**Classes**

* (opcional) User
* (opcional) PhotoManager
* Family (modelo)
* Deceased (modelo)
* Service (modelo)
* ProgramController
* DbWorker
* DbAdapter
* ConnectionManager

**Métodos**

**ProgramController (faz “forwarding” de métodos para managers)**

**Family**

* family\_SelectALL
* family\_Add
* family\_Update
* family\_Remove

**Deceased**

* deceased\_SelectALL
* deceased \_Add
* deceased \_Update
* deceased \_Remove

**Service**

* service\_SelectALL
* service \_Add
* service \_Update
* service \_Remove

**Search**

**(for family)**

* search\_SelectAllFamilyNames
* search\_ById
* search\_ByName
* search\_ByTableList

**(for deceased)**

* search\_SelectAllDeceasedNames
* search\_ById
* search\_ByName
* search\_ByTableList

**(for service)**

* search\_filter
* search\_SelectAll

**DbWorker (funciona como: data base manager)**

**Family**

* family\_SelectALL
* family\_Add
* family\_Update
* family\_Remove

**Deceased**

* deceased\_SelectALL
* deceased \_Add
* deceased \_Update
* deceased \_Remove

**Service**

* service\_SelectALL
* service \_Add
* service \_Update
* service \_Remove

**Search**

**(for family)**

* search\_SelectAllFamilyNames
* search\_ById
* search\_ByName
* search\_ByTableList

**(for deceased)**

* search\_SelectAllDeceasedNames
* search\_ById
* search\_ByName
* search\_ByTableList

**(for service)**

* search\_filter
* search\_SelectAll

**DbAdapter**

**Family**

* family\_SelectALL
* family\_Add
* family\_Update
* family\_Remove

**Deceased**

* deceased\_SelectALL
* deceased \_Add
* deceased \_Update
* deceased \_Remove

**Service**

* service\_SelectALL
* service \_Add
* service \_Update
* service \_Remove

**Search**

**(for family)**

* search\_SelectAllFamilyNames
* search\_ById
* search\_ByName
* search\_ByTableList

**(for deceased)**

* search\_SelectAllDeceasedNames
* search\_ById
* search\_ByName
* search\_ByTableList

**(for service)**

* search\_filter
* search\_SelectAll

## Outras informações

Uma imagem com texto, captura de ecrã, ecrã, número

Descrição gerada automaticamente**Diagrama**

**Workbench Script**

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema safJava

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema safJava

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `safJava` DEFAULT CHARACTER SET utf8 ;

USE `safJava` ;

-- -----------------------------------------------------

-- Table `safJava`.`Family`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `safJava`.`Family` (

`Id` INT NOT NULL AUTO\_INCREMENT,

`Name` VARCHAR(50) NOT NULL,

`Contact` VARCHAR(50) NOT NULL,

`Address` VARCHAR(50) NULL,

`Relation` VARCHAR(50) NULL,

`Photo` VARCHAR(50) NULL,

PRIMARY KEY (`Id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `safJava`.`Deceased`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `safJava`.`Deceased` (

`Id` INT NOT NULL AUTO\_INCREMENT,

`Name` VARCHAR(50) NOT NULL,

`Age` INT NOT NULL,

`Sex` CHAR(1) NOT NULL,

`Comment` VARCHAR(50) NULL,

`Photo` VARCHAR(50) NULL,

PRIMARY KEY (`Id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `safJava`.`Service`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `safJava`.`Service` (

`Id` INT NOT NULL AUTO\_INCREMENT,

`FamilyId` INT NOT NULL,

`DeceasedId` INT NOT NULL,

`Comment` VARCHAR(50) NOT NULL,

`Date` DATE NULL,

`Price` DECIMAL(18,0) NOT NULL,

`CheckPay` TINYINT NOT NULL,

PRIMARY KEY (`Id`),

INDEX `fk\_Service\_Family\_idx` (`FamilyId` ASC) VISIBLE,

INDEX `fk\_Service\_Deceased1\_idx` (`DeceasedId` ASC) VISIBLE,

CONSTRAINT `fk\_Service\_Family`

FOREIGN KEY (`FamilyId`)

REFERENCES `safJava`.`Family` (`Id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Service\_Deceased1`

FOREIGN KEY (`DeceasedId`)

REFERENCES `safJava`.`Deceased` (`Id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

**Lógica do programa**

Este projeto será elaborado com arquitetura ***Three-Tier***  
(três camadas).

É um modelo de *design* de software em que o código é dividido em três partes distintas:

* Camada de apresentação;
* Camada de lógica do negócio;
* Camada de dados;

Cada camada tem uma responsabilidade específica dentro do programa.

A camada de apresentação lida com a interface com o utilizador, a camada de lógica do negócio contém a lógica da aplicação e a camada de dados é responsável pela interação com a bse de dados.

Este modelo é comum na construção de aplicações web e facilita a manutenção e escalabilidade do software.

De forma que o software deverá funcionar da seguinte forma:

Uma imagem com texto, círculo, captura de ecrã, design

Descrição gerada automaticamente