PROJETO INTEGRADOR 2025-1	
SENAI Serviço Nacional de Aprendizagem Industrial	**FACULDADE SENAI FATESG**
Coordenação Pedagógica: Fernanda Garcia **Coordenação Técnica: Fabricia Neres Borge** **Professor Líder: Luiz Mário Lustosa Pascoal**	**Ano-Semestre** **2025-2**
Curso: Superior de Bacharelado em Engenharia de Software	**Período: 4º**

Competências associadas:

- UC1 - Fundamentar, sistematizar, medir, disciplinar, qualificar e quantificar estruturas e processos da engenharia software;
- UC2 - Elicitar, analisar, especificar e validar requisitos de software, bem como gerenciar requisitos durante todo o ciclo de vida do produto de software; planejar, criar e manter o design e a arquitetura do software
- UC3 - Compreender e aplicar processos, técnicas e procedimentos de construção, evolução e avaliação de software; analisar e selecionar tecnologias adequadas para a construção de software.
- UC4 - Gerenciar a configuração de software; testar software; conhecer os direitos e propriedades intelectuais inerentes à produção e utilização de software; avaliar a qualidade de sistemas de software.
- UC5 - Gerenciar processos de engenharia de software; Gerenciar Engenharia de Software; Gerenciar Qualidade de Software.

1 – Informações do Projeto Integrador

Tema: Detalhamento de Rotas para Pontos de Coleta	**Data de Início: 17/11/2025** **Data de Conclusão: 08/12/2025**

RESUMO

Desenvolvimento de um sistema computacional que permite a partir da leitura de uma base de dados de entrada em arquivo no formato CSV, implementar o algoritmo de roteamento baseado nos dados fornecidos e retornar a melhor rota encontrada.

Palavra-chave:	Métodos ágeis, Processamento de Arquivos, Desenvolvimento de Sistemas.

2 – Contexto

A empresa fictícia Vitalis Tech, especializada em soluções tecnológicas para saúde e gestão de emergências, foi contratada pela Secretaria Municipal de Saúde da cidade de Cidália para desenvolver um Sistema Inteligente de Gestão e Despacho de Atendimentos de Emergência, denominado SOS-Rota.

Com o crescimento da cidade, aumento do fluxo de veículos e da quantidade de acidentes e ocorrências clínicas, o serviço de atendimento pré-hospitalar enfrenta dificuldades em decidir rapidamente qual ambulância enviar,

de onde ela deve sair e se o tempo de resposta cumpre o SLA estabelecido por gravidade do caso. Diante desse cenário, a Secretaria de Saúde deseja um sistema que:

- Permita o cadastro e gerenciamento de ocorrências (acidentes e demais emergências), incluindo localização, gravidade e status;
- Permita o cadastro e gerenciamento de ambulâncias (placa, tipo, status) e equipes de atendimento (médico, enfermeiro, condutor);
- Modele a cidade como um grafo, com bairros/bases como vértices e ruas como arestas ponderadas em km, permitindo o cálculo de caminho mínimo (Dijkstra);
- Auxilie o regulador a selecionar automaticamente as ambulâncias aptas para cada ocorrência, respeitando regras de SLA por gravidade e compatibilidade de tipo de ambulância;
- Registre o histórico de despachos e atendimentos, permitindo consultas gerenciais posteriores.

A Vitalis Tech decidiu atribuir a missão de desenvolver o SOS-Rota à equipe de desenvolvimento júnior — isto é, vocês, alunos do curso de Engenharia de Software.

A empresa **Vitalis Tech** deseja automatizar o processo de cadastro das **ocorrências de emergência**, das **bases e vias** que se conectam a esses pontos, das **ambulâncias** que realizam os atendimentos e de suas **equipes e escalas**, por meio da manipulação estruturada dos dados. O objetivo é garantir uma **visualização organizada** das informações e **facilitar a gestão operacional**, por meio de operações como: criação, leitura, atualização e deleção.

Essas quatro operações fundamentais em sistemas de gerenciamento de dados são representadas pelo acrônimo **CRUD** (Create, Read, Update, Delete), sendo utilizadas em bases de dados relacionais e disponibilizadas aos utilizadores do sistema. Em resumo, elas traduzem os seguintes conceitos:

- **Create:** A operação de criação refere-se ao processo de adicionar novos registros (ou dados) ao banco de dados. É o primeiro passo no ciclo de vida dos dados — etapa em que o sistema acumula informações essenciais para suas funções. No contexto do SOS-Rota, por exemplo, ao cadastrar uma nova ocorrência de emergência ou registrar uma nova ambulância na frota, a operação Create é utilizada para inserir esses dados no banco.
- **Read:** A operação de leitura é utilizada para consultar e recuperar dados do banco de dados. Com essa função, é possível visualizar as informações armazenadas sem modificar o conteúdo já existente. No sistema, isso inclui listar todas as ocorrências em aberto, consultar ambulâncias disponíveis por tipo e base, ou ainda buscar uma ocorrência específica por filtros como gravidade, bairro ou status.
- **Update:** Esta etapa refere-se ao processo de modificar dados já existentes no banco de dados. Essa operação é essencial para manter as informações sempre coerentes com a realidade operacional. Exemplos no SOS-Rota incluem alterar o status de uma ocorrência (de “Aberta” para “Despachada” ou “Concluída”), atualizar o status de uma ambulância (de “Disponível” para “Em atendimento”), ajustar

a associação de uma equipe a determinada ambulância ou corrigir dados de distância/custo em uma ligação viária do grafo.

- **Delete:** O objetivo da operação de exclusão é um: remover dados do banco de dados. Essa função é importantíssima na hora de gerenciar o ciclo de vida dos dados — já que, com ela, os sistemas podem eliminar informações desnecessárias ou supérfluas. Contudo, fica o adendo: a função Delete (Excluir) deve ser usada sempre com muita cautela — já que a exclusão de dados é, na maioria das vezes, irreversível.

Fonte e maiores detalhes sobre Kanban para compreensão sobre a metodologia.
https://mercadoonlinedigital.com/blog/crud/
https://www.brasilcode.com.br/o-que-e-crud-por-que-voce-precisa-criar-um-crud/
https://pt.stackoverflow.com/questions/359961/o-que-caracteriza-um-projeto-crud-b%C3%A1sico

3 – Problema

Para que o sistema atenda de forma eficaz à realidade do atendimento de emergências em uma cidade de médio porte como Cidália, é necessário ir além de um cadastro simples de ocorrências. A complexidade envolve **múltiplas ambulâncias**, **várias bases de atendimento**, **restrições de gravidade**, **tipos de ambulância** (Básica, UTI), **equipes completas** e, principalmente, o **tempo de resposta (SLA)** impactando diretamente a vida dos pacientes. Assim, o sistema a ser desenvolvido deverá contemplar as seguintes necessidades fundamentais:

1. Cadastro e Gerenciamento de Ambulâncias

Cada ambulância precisa estar devidamente registrada no sistema com os seguintes atributos:

- Identificação (placa);
- Tipo (ex.: Básica, UTI);
- Status operacional (Disponível, Em atendimento, Em manutenção);
- Base à qual está vinculada (bairro - ponto físico de partida);

Essas informações são essenciais para restringir quais ambulâncias podem ser usadas em cada ocorrência, de acordo com tipo requerido e disponibilidade.

2. Cadastro de Profissionais e Montagem de Equipes

O atendimento é realizado por equipes de saúde, alocadas às ambulâncias por turno. O sistema deve permitir:

- Cadastro de profissionais (nome, função: Médico, Enfermeiro, Condutor, contato);
- Criação de equipes de atendimento associadas a uma ambulância;
- Definição de equipes mínimas, por exemplo, para uma UTI: Condutor + Enfermeiro + Médico;

- Garantia de que um profissional não esteja alocado a duas equipes ativas simultaneamente.

3. Cadastro de Mapa (Grafo) e Cálculo de Caminho Mínimo

A cidade será modelada como um grafo, onde:

- Os **bairros/bases são os vértices**;
- As **ligações viárias (ruas) são as arestas**, cada uma com um peso (distância em km).
- Algumas bases são pontos de partida das ambulâncias;
- As ocorrências são associadas a um vértice (bairro/local de atendimento).

O sistema deve ser capaz de calcular o **menor caminho entre a base da ambulância e o local da ocorrência**, utilizando o **algoritmo de Dijkstra**, considerando apenas ligações válidas e conectadas.

4. Registro e Triagem de Ocorrências (Acidentes e Emergências)

O sistema deve permitir o **cadastro de ocorrências**, contemplando:

- Local (bairro/ponto, associado ao grafo);
- Tipo de ocorrência (acidente de trânsito, mal súbito, trauma, etc.);
- Gravidade (**Alta**, **Média**, **Baixa**);
- Data/hora de abertura;
- Status (Aberta, Despachada, Em atendimento, Concluída, Cancelada);
- Observações e dados complementares.

A **gravidade da ocorrência** define o **SLA de atendimento** e o **tipo de ambulância requerido**. Por exemplo:

- Gravidade **Alta** → SLA de **8 minutos**, exigindo **ambulância UTI**;
- Gravidade **Média** → SLA de **15 minutos**, exigindo ao menos **ambulância Básica**.
- Gravidade **Baixa** → SLA de **30 minutos**, exigindo ao menos **ambulância Básica**.

Para fins didáticos, consideraremos a regra:

- **1 km de distância = 1 minuto de tempo**, assumindo velocidade média de 60 km/h.

5. Despacho Otimizado e Registro de Atendimentos

O coração do sistema é a lógica de despacho. A partir de uma ocorrência recém-cadastrada, o sistema deve:

- Identificar as **ambulâncias disponíveis** e com **equipe mínima ativa**;
- Verificar **compatibilidade de tipo** (UTI ou Básica) com o requerido pela ocorrência;
- Utilizar o **algoritmo de Dijkstra** para calcular a **distância (em km)** entre a base da ambulância e o local da ocorrência;
- Validar se a ambulância candidata **cumpre o SLA** (distância em km ≤ SLA em minutos);
- Registrar o **despacho**, atualizando status da ambulância e da ocorrência;
- Manter histórico de **atendimentos realizados** para consultas posteriores.

Para garantir coerência, integridade e segurança operacional, o sistema SOS-Rota deverá seguir as seguintes regras de negócio:

1. Acesso e Login

- O sistema deverá possuir uma funcionalidade de **autenticação de usuários por login e senha**;
- Senhas devem ser armazenadas de forma segura (hash, nunca em texto plano);
- Usuários não autenticados não poderão acessar nenhuma funcionalidade do sistema.

2. Cadastro de Ocorrências

- Toda ocorrência deve ter: identificação, local, tipo, gravidade, data/hora de abertura e status;
- **Gravidade Alta**:
  - Define SLA de **8 minutos**;
  - Requer ambulância do tipo **UTI**;
  - A ambulância selecionada deve estar a uma distância **≤ 8 km** do local.
- **Gravidade Média**:
  - Define SLA de **15 minutos**;
  - Requer ao menos ambulância do tipo **Básica**;
  - A ambulância selecionada deve estar a uma distância **≤ 15 km**.
- Uma ocorrência não pode ser marcada como **Concluída** sem ter passado por status **Despachada/Em atendimento**.

3. Cadastro de Ambulâncias e Equipes

- Cada ambulância deve possuir **placa única**, tipo, status e base associada;
- Uma ambulância só pode estar com status **Disponível** se possuir **equipe completa** alocada;
- Para uma ambulância do tipo **UTI**, a equipe mínima deve conter: **Condutor + Enfermeiro + Médico**;
- Um profissional não pode estar alocado em **mais de uma equipe ativa** ao mesmo tempo.

4. Despacho e Atendimento (Lógica Central)

- O sistema **só pode despachar** uma ambulância se:
  - A ambulância estiver **Disponível**;
  - O tipo da ambulância for **compatível** com o tipo requerido pela ocorrência;
  - A **distância calculada por Dijkstra** entre a base da ambulância e o local da ocorrência for **menor ou igual ao SLA** (em minutos) definido pela gravidade;
- Ao despachar uma ambulância, o sistema deve:
  - Alterar o Status da ambulância para **Em atendimento**;
  - Alterar o Status da ocorrência para **Despachada/Em atendimento**;
- Uma ambulância não pode ser despachada para **duas ocorrências simultâneas**.

5. Funcionalidades de Consulta

O sistema deverá permitir diversas consultas, incluindo:

- Listar **ambulâncias disponíveis**, por tipo, base e turno;
- Listar **ocorrências em aberto**, filtrando por gravidade, bairro e status;
- Consultar **histórico de atendimentos** por ambulância, por período, por gravidade;
- Consultar **tempo médio de resposta** por tipo de gravidade;
- Consultar **mapa de ocorrências** (quantidade de chamadas) por bairro.

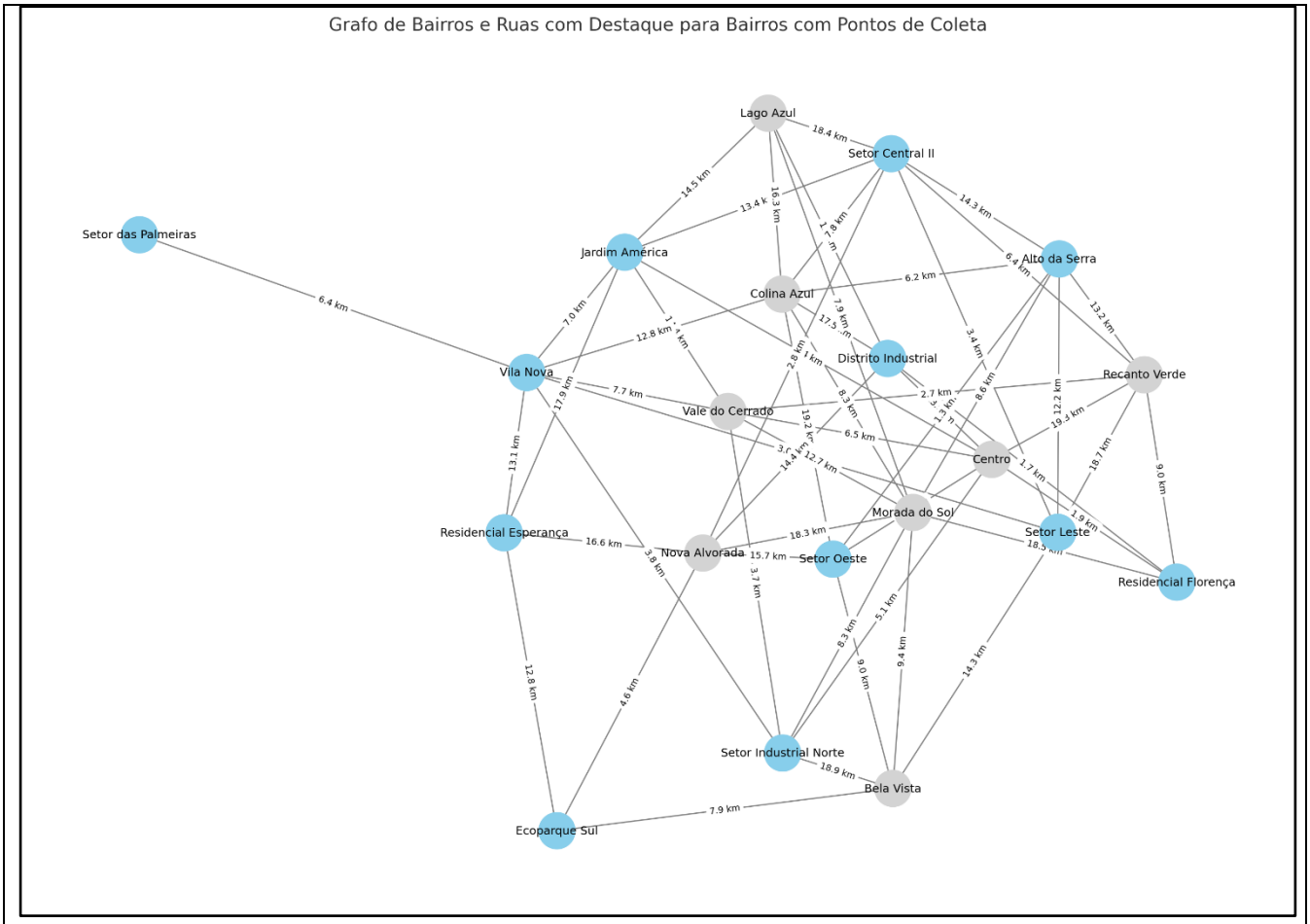
6. Restrições e Validações Globais

- Não é permitido **excluir** uma ambulância que esteja vinculada a **atendimentos históricos** (somente inativar);
- Não é permitido **excluir** profissional que esteja em **equipe ativa**;
- Não é permitido **excluir** uma ocorrência se ela já possui registro de despacho/atendimento (apenas cancelamento com justificativa);
- Qualquer tentativa de violar regras de negócio deve ser bloqueada com **mensagem clara ao usuário**.

Para iniciar o desenvolvimento, os alunos deverão utilizar dois arquivos CSV fornecidos, que representam o cenário urbano. Esses dados devem ser carregados em um banco de dados relacional (PostgreSQL ou MySQL) e servirão como base para as consultas e funcionalidades do sistema.

- **Bairros.csv:** contém os bairros do município. Link para download: https://raw.githubusercontent.com/luizmlpascoal/datasets-to-share/refs/heads/main/bairros.csv
- **arestas_conexoes.csv**: contém as ligações (ruas) entre os bairros com as respectivas distâncias (em km). Link para download: https://github.com/luizmlpascoal/datasets-to-share/blob/main/ruas_conexoes.csv

A seguir, apresenta-se uma representação visual dos bairros (vértices) e das ruas que os conectam (arestas). Os bairros destacados em azul indicam os locais que podem ser sinalizados pelo usuário como ponto de ocorrência de um acidente. Para cada novo acidente, o usuário deverá informar o bairro da ocorrência e o sistema, automaticamente, selecionará o ponto de partida (base) com ambulância disponível que possua o menor tempo estimado de deslocamento até o bairro informado.



Cada grupo deverá criar as tabelas correspondentes no banco de dados, importar os dados dos arquivos CSV e garantir que estejam prontos para uso no sistema, especialmente para o cálculo de rotas e execução de consultas. O banco de dados deve ser relacional (preferencialmente PostgreSQL ou MySQL). Essa etapa é obrigatória e será avaliada.

Em seguida, este sistema deverá ser obrigatoriamente na **linguagem JAVA**, podendo ser utilizada o Java Nativo com bibliotecas de apoio a interface gráfica (Swing ou JavaFX) ou com Java Web (JSF com Primefaces para componentes) ou desenvolvimento com microserviços (SpringBoot) e *frontend* em Javascript (Framework Livre). A arquitetura de software adotada deverá ser de escolha livre do grupo, porém obrigatoriamente as regras de negócio e conexão com o banco de dados deverão ser feitas **em JAVA.**

Material de Apoio:

- DEITEL, Harvey M. Java: como programar. 8a. Ed. São Paulo: Pearson, 2010.
- RANGEL, Pablo; DE CARVALHO JR, José Gomes. Sistemas Orientados a Objetos: teoria e prática com UML e Java. Brasport.
- BARNES, David John; KÖLLING, Michael; GOSLING, James. Objects First with Java: A practical introduction using BlueJ. Pearson/Prentice Hall, 2016.

4 – Produtos de entrega

Componente de Documentação de Requisitos do Sistema (ERS):

Cada grupo deverá elaborar um **Documento de Requisitos de Software (ERS)** descrevendo de forma clara e objetiva:

- O que o sistema SOS-Rota deve realizar;
- Como os usuários irão interagir com ele;
- Quais são as regras e fluxos fundamentais.

A proposta é manter um nível de documentação **útil e prático**, sem excesso de formalismo, mas suficientemente organizado para dar suporte ao desenvolvimento, testes e apresentação final.

1. Capa

- Nome do projeto: **SOS-Rota: Gestão e Despacho de Atendimentos de Emergência**;
- Nome dos integrantes do grupo;

2. Objetivo Específico (Escopo)

- Apoiar a gestão de atendimentos de emergência na cidade de Cidália, permitindo o cadastro estruturado de ocorrências, a gestão de ambulâncias e equipes, o cálculo de rotas mínimas com Dijkstra e o despacho otimizado de viaturas, respeitando SLAs de atendimento definidos por gravidade.

3. Interfaces do Software

Descrição textual (ou protótipos) das principais telas, por exemplo:

- Tela de Login;
- Tela de Dashboard/Monitoramento (lista de ocorrências abertas, mapa/lista de bases e ambulâncias);
- Tela de Cadastro de Ocorrências;
- Tela de Cadastro de Ambulâncias;
- Tela de Cadastro de Profissionais e Equipes;
- Tela/funcionalidade de Despacho de Ambulância;
- Tela de Consultas e Relatórios (histórico de atendimentos, tempo de resposta, etc.).
- Pode ser feito com ferramentas como Figma, papel digitalizado, ou captura da interface real.

4. Interfaces de Comunicação

- Como o sistema se comunica com o banco de dados (PostgreSQL ou MySQL);
- Se for web: como os dados trafegam (ex: Java + JDBC + HTML);
- Se for desktop: como ocorre a persistência dos dados (JDBC, ORM, etc.).

5. Requisitos Específicos

Histórias de Usuário (HU):

- Ex.: “Como Regulador, quero cadastrar uma nova ocorrência de Gravidade Alta, para que o sistema busque automaticamente as ambulâncias do tipo UTI aptas a atendê-la dentro do SLA de 8 minutos.”
- Ex.: “Como Gestor de Frota, quero montar a equipe do dia e associá-la às ambulâncias, para que apenas viaturas com equipe completa sejam consideradas disponíveis para despacho.”

Requisitos Funcionais (RF):

- RF01 – O sistema deve permitir o cadastro, consulta, atualização e cancelamento de ocorrências.
- RF02 – O sistema deve permitir o cadastro de ambulâncias com placa, tipo, base e status.
- RF03 – O sistema deve permitir o cadastro de profissionais e montagem de equipes.
- RF04 – O sistema deve calcular o caminho mínimo (Dijkstra) entre uma base e o local da ocorrência.
- RF05 – O sistema deve sugerir ambulâncias aptas para uma ocorrência, respeitando tipo e SLA.
- RF06 – O sistema deve registrar o despacho de ambulância e atualizar os status correspondentes.
- RF07 – O sistema deve permitir consultas de histórico de atendimentos por ambulância, período e gravidade.
- (… completar com outros RFs relevantes).

Requisitos Não Funcionais (RQ):

- RNF01 – O sistema deve armazenar senhas com uso de hash criptográfico.
- RNF02 – O sistema deve estar disponível em ambiente local sem necessidade de internet.

Regras de Domínio:

Citar explicitamente as regras mais importantes, como:

- Uma ambulância só pode ser despachada se estiver Disponível e com equipe completa;
- Gravidade Alta exige ambulância UTI e SLA de 8 minutos;
- Não é possível atribuir a mesma ambulância a duas ocorrências simultâneas.

Restrições Técnicas ou de Projeto:

- O banco de dados deve ser relacional (PostgreSQL ou MySQL);
- A lógica de negócio, incluindo Dijkstra, deve ser implementada em Java;

- Utilizar estruturas de dados da biblioteca java.util (lista, etc.).

6. Descrição dos Dados do Sistema

Breve explicação das principais entidades do banco, por exemplo:

- **Bases** (id, nome, bairro, endereço);
- **Ambulancias** (id, placa, tipo, status, base_id);
- **Profissionais** (id, nome, função, contato, ativo);
- **Equipes** (id, descrição, ambulancia_id);
- **Equipe_Profissional** (id, equipe_id, profissional_id);
- **Ocorrencias** (id, tipo, gravidade, local, data_hora_abertura, status, observacao);
- **Atendimentos** (id, ocorrencia_id, ambulancia_id, data_hora_despacho, data_hora_chegada, distancia_km);
- **Usuarios** (id, login, senha_hash, perfil).
- (opcional) Mini-diagrama relacional ou dicionário de dados.

7. Validação e Verificação

- Cada grupo deverá demonstrar que o sistema implementado **reflete corretamente os requisitos documentados**;
- O professor poderá cobrar **evidência (prints, execução, testes)** de pelo menos uma HU, um RF e uma regra de domínio aplicada no sistema.

Componente: Estrutura de dados II

- Utilizar POO e as estrutura de dados da biblioteca Util do JAVA.
- Usar a Interface LIST da Java util;
- A estrutura de dados básica pode ser: lista, pilha ou fila;
- Implementação de Algoritmo de Caminho único (Dijsktra) para cálculo de rota no sistema.
- Projeto contendo o código-fonte do sistema.

Componente: Banco de Dados II

- Realizar a criação das tabelas do Banco de Dados, juntamente com as suas chaves primárias e estrangeiras.
- Criar as tabelas do banco de dados, com suas **chaves primárias** e **estrangeiras**;
- Realizar a criação de duas consultas SQL que retornem algum valor de negócio relevante para a proposta do software proposto.

- Os resultados dessas consultas devem ser exibidos pelas telas do sistema desenvolvido.

Componente: Teoria da Computação

Cada grupo deverá aplicar e documentar **ao menos dois** dos tópicos abaixo, com: teoria aplicada, contextualização prática no projeto, demonstração numérica ou implementação em código. Exemplos:

Lógica Proposicional / Predicados:

- Modelar a regra de despacho:
    - P(a): "Ambulância a está Disponível";
    - Q(a, o): "Ambulância a possui o tipo requerido pela ocorrência o";
    - R(a, o): "Distância entre base(a) e local(o), calculada por Dijkstra, é ≤ SLA(o)";
    - Despachar(a, o) ⇔ P(a) ∧ Q(a, o) ∧ R(a, o).
- Utilizar **tabelas verdade** para validar combinações de condições críticas.

5 – Critérios de avaliação

Apresentar o documento de requisitos (ERS) do sistema desenvolvido, contemplando as regras de negócio implementadas, histórias de usuário e testes funcionais realizados.

Apresentar o software funcionando, com suas devidas validações de dados.

Apresentar o código fonte do sistema, com suas devidas validações de dados, bem como com as lógicas desenvolvidas e comentários gerais.

Deverá ser feito na linguagem Java, preferencialmente fazer uso de interface gráfica, ou Java Web (JSF) ou Java com Microserviços (SpringBoot) e frontend Javascript (Framework livre).

Uso de banco de dados relacional PostgreSQL ou MySQL.

Observações:

A nota deste projeto será composta de 50% referente às atividades em grupo e 50% ao desempenho individual, cujos critérios serão estabelecidos por cada professor;

Para cada ausência do aluno, será descontada 10% da nota individual;

O conteúdo de cada unidade curricular será avaliado individualmente por cada professor;

6 – Cronograma

AÇÃO	DATA

Início do Projeto	17/11/2025
Ponto de Controle com professor líder *(status report)*	01/12/2025
Apresentação final do Projeto Integrador	08/12/2025
Entrega de todos os artefatos do Projeto Integrador	08/12/2025

7 – Instruções

1. Os alunos deverão procurar os professores de cada disciplina para receber os detalhes do projeto referente a sua entrega.
2. A nota desse projeto será composta em 50% para as atividades em grupo e 50% ao desempenho individual, em que os critérios serão estabelecidos por cada professor;
3. O desempenho será avaliado na equipe como um todo, então o fracasso da sua equipe implica no seu fracasso. Por isso, escolha bem os seus pares;
4. Para cada ausência do aluno, será descontada 10% da nota individual;
5. Cada grupo será composto por: no mínimo 3 alunos e no máximo 4 alunos – nos casos em que não for possível o cumprimento desta instrução, competirá ao professor encarregado a resolução do conflito;
6. Cada grupo entregará apenas um corpo de documentos;
7. Os alunos terão o período de 17/11/2025 até 08/12/2025 para projetar, construir, implementar/configurar suas soluções aplicadas;
8. Na semana do dia 08/12/2025 ocorrerão as apresentações dos trabalhos de todos os grupos e períodos.
    - A entrega dos trabalhos acontecerá dia 08/12/2025, até às 23h59min, com o envio do trabalho na pasta compartilhada pelo professor líder;
9. O dia da apresentação conta como presença. Caso o aluno falte no dia da apresentação, ele terá sua nota individual descontada em 10%;

8 – Grupos

9 – Contatos

1. [LÍDER] Luiz Mário Lustosa Pascoal - luizpascoal.senai@fieg.com.br
2. [COLABORADOR] Reinaldo de Souza Júnior - reinaldosouza.senai@fieg.com.br
3. [COLABORADOR] Gustavo Siqueira Vinhal - gustavovinhal.senai@fieg.com.br