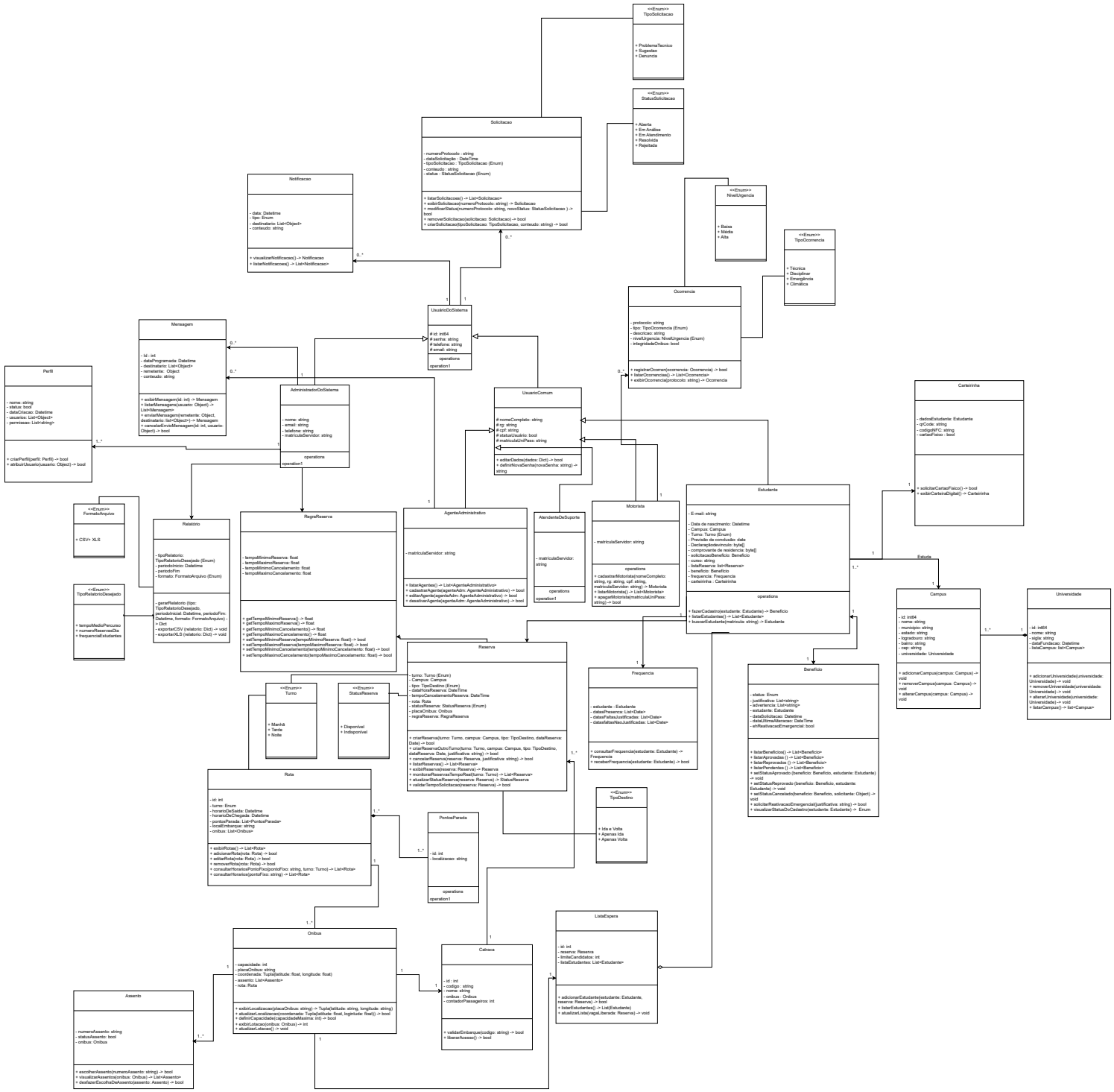
<<Enum>>
TipoSolicitacao
+ ProblemaTecnico
+ Sugestao
+ Denuncia
<<Enum>>
StatusSolicitacao
+ Aberta
+ Em Análise
+ Em Atendimento
+ Resolvida
+ Rejeitada
Solicitacao
- numeroProtocolo : string
- dataSolicitação : DateTime
- tipoSolicitacao : TipoSolicitacao (Enum)
- conteudo : string
- status : StatusSolicitacao (Enum)
+ listarSolicitacoes() -> List<Solicitacao>
+ exibirSolicitacao(numeroProtocolo: string) -> Solicitacao
+ modificarStatus(numeroProtocolo: string, novoStatus: StatusSolicitacao) -> bool
+ removerSolicitacao(solicitacao: Solicitacao) -> bool
+ criarSolicitacao(tipoSolicitacao: TipoSolicitacao, conteudo: string) -> bool
Notificacao
- data: Datetime
- tipo: Enum
- destinatario: List<Object>
- conteudo: string
+ visualizarNotificacoes() -> Notificacao
+ listarNotificacoes() -> List<Notificacao>
<<Enum>>
NivelUrgencia
+ Baixa
+ Média
+ Alta
<<Enum>>
TipoOcorrencia
+ Técnica
+ Disciplinar
+ Emergência
+ Climática
Ocorrencia
- protocolo: string
- tipo: TipoOcorrencia (Enum)
- descricao: string
- nivelUrgencia: NivelUrgencia (Enum)
- integridadeOnibus: bool
+ registrarOcorrencia(ocorrencia: Ocorrencia) -> bool
+ listarOcorrencias() -> List<Ocorrencia>
+ exibirOcorrencia(protocolo: string) -> Ocorrencia
UsuarioDoSistema
id: int64
senha: string
telefone: string
email: string
operations
operation1
Mensagem
- id : int
- dataProgramada: Datetime
- destinatario: List<Object>
- remetente: Object
- conteudo: string
+ exibirMensagem(id: int) -> Mensagem
+ listarMensagens(usuario: Object) -> List<Mensagem>
+ enviarMensagem(remetente: Object, destinatario: list<Object>) -> Mensagem
+ cancelarEnvioMensagem(id: int, usuario: Object) -> bool
Perfil
- nome: string
- status: bool
- dataCriacao: Datetime
- usuarios: List<Object>
- permissao: List<string>
+ criarPerfil(perfil: Perfil) -> bool
+ atribuirUsuario(usuario: Object) -> bool
AdministradorDoSistema
- nome: string
- email: string
- telefone: string
- matriculaServidor: string
operations
operation1
UsuarioComum
nomeCompleto: string
rg: string
cpf: string
statusUsuario: bool
matriculaUniPaas: string
+ editarDados(dados: Dict) -> bool
+ definirNovaSenha(novaSenha: string) -> string
Carteirinha
- dadosEstudante: Estudante
- qrCode: string
- codigoNFC: string
- carteiraFisico : bool
+ solicitarCartaoFisico() -> bool
+ exibirCarteiraDigital() -> Carteirinha
<<Enum>>
FormatoArquivo
+ CSV+ XLS
Relatorio
- tipoRelatorio: TipoRelatorioDesejado (Enum)
- periodoInicio: Datetime
- periodoFim
- formato: FormatoArquivo (Enum)
+ gerarRelatorio (tipo: TipoRelatorioDesejado, periodoInicial: Datetime, periodoFim: Datetime, formato: FormatoArquivo) -> Dict
- exportarCSV (relatorio: Dict) -> void
- exportarXLS (relatorio: Dict) -> void
<<Enum>>
TipoRelatorioDesejado
+ tempoMedioPercurso
+ numeroReservasDia
+ frequenciaEstudantes
RegraReserva
- tempoMinimoReserva: float
- tempoMaximoReserva: float
- tempoMinimoCancelamento: float
- tempoMaximoCancelamento: float
+ getTempoMinimoReserva() -> float
+ getTempoMaximoReserva() -> float
+ getTempoMinimoCancelamento() -> float
+ getTempoMaximoCancelamento() -> float
+ setTempoMinimoReserva(tempoMinimoReserva: float) -> bool
+ setTempoMaximoReserva(tempoMaximoReserva: float) -> bool
+ setTempoMinimoCancelamento(tempoMinimoCancelamento: float) -> bool
+ setTempoMaximoCancelamento(tempoMaximoCancelamento: float) -> bool
AgenteAdministrativo
- matriculaServidor: string
+ listarAgentes() -> List<AgenteAdministrativo>
+ cadastrarAgente(agenteAdm: AgenteAdministrativo) -> bool
+ editarAgente(agenteAdm: AgenteAdministrativo) -> bool
+ desativarAgente(agenteAdm: AgenteAdministrativo) -> bool
AtendenteDeSuporte
- matriculaServidor: string
operations
operation1
Motorista
- matriculaServidor: string
operations
+ cadastrarMotorista(nomeCompleto: string, rg: string, cpf: string, matriculaServidor: string) -> Motorista
+ listarMotoristas() -> List<Motorista>
+ apagarMotorista(matriculaUniPaas: string) -> bool
Estudante
- E-mail: string
- Data de nascimento: Datetime
- Campus: Campus
- Turno: Turno (Enum)
- Previsão de conclusão: date
- Declaração de vínculo: byte[]
- comprovante de residência: byte[]
- solicitacaoBeneficio: Beneficio
- curso: string
- listaReserva: list<Reserva>
- beneficio: Beneficio
- frequencia: Frequencia
- carteirinha : Carteirinha
operations
+ fazerCadastro(estudante: Estudante) -> Beneficio
+ listarEstudantes() -> List<Estudante>
+ buscarEstudante(matricula: string) -> Estudante
Estuda
Campus
- id: int64
- nome: string
- municipio: string
- estado: string
- logradouro: string
- bairro: string
- cep: string
- universidade: Universidade
+ adicionarCampus(campus: Campus) -> void
+ removerCampus(campus: Campus) -> void
+ alterarCampus(campus: Campus) -> void
Universidade
- id: int64
- nome: string
- sigla: string
- dataFundacao: Datetime
- listaCampus: list<Campus>
+ adicionarUniversidade(universidade: Universidade) -> void
+ removerUniversidade(universidade: Universidade) -> void
+ alterarUniversidade(universidade: Universidade) -> void
+ listarCampus() -> list<Campus>
Reserva
- turno: Turno (Enum)
- Campus: Campus
- tipo: TipoDestino (Enum)
- dataHoraReserva: DateTime
- tempoCancelamentoReserva: DateTime
- rota: Rota
- statusReserva: StatusReserva (Enum)
- placaOnibus: Onibus
- regraReserva: RegraReserva
+ criarReserva(turno: Turno, campus: Campus, tipo: TipoDestino, dataReserva: Date) -> bool
+ criarReservaOutroTurno(turno: Turno, campus: Campus, tipo: TipoDestino, dataReserva: Date, justificativa: string) -> bool
+ cancelarReserva(reserva: Reserva, justificativa: string) -> bool
+ listarReservas() -> List<Reserver>
+ exibirReserva(reserva: Reserva) -> Reserva
+ monitorarReservasTempoReal(turno: Turno) -> List<Reserva>
+ atualizarStatusReserva(reserva: Reserva) -> StatusReserva
+ validarTempoSolicitacao(reserva: Reserva) -> bool
<<Enum>>
Turno
+ Manhã
+ Tarde
+ Noite
<<Enum>>
StatusReserva
+ Disponível
+ Indisponível
Frequencia
- estudante : Estudante
- datasPresenca: List<Date>
- datasFaltasJustificadas: List<Date>
- datasFaltasNaoJustificadas: List<Date>
+ consulteFrequencia(estudante: Estudante) -> Frequencia
+ receberFrequencia(estudante: Estudante) -> bool
Beneficio
- status: Enum
- justificativa: List<string>
- advertencia: List<string>
- estudante: Estudante
- dataSolicitacao: Datetime
- dataUltimaAlteracao: DateTime
- emRealocacaoEmergencial: bool
+ listarBeneficios() -> List<Beneficio>
+ listarAprovados () -> List<Beneficio>
+ listarReprovados () -> List<Beneficio>
+ listarPendentes () -> List<Beneficio>
+ setStatusAprovado (beneficio: Beneficio, estudante: Estudante) -> void
+ setStatusReprovado (beneficio: Beneficio, estudante: Estudante) -> void
+ setStatusCancelado(beneficio: Beneficio, solicitante: Object) -> void
+ solicitarRealocacaoEmergencial(justificativa: string) -> bool
+ visualizarStatusDoCadastro(estudante: Estudante) -> Enum
Rota
- id: int
- turno: Enum
- horarioDeSaida: Datetime
- horarioDeChegada: Datetime
- pontosParada: List<PontosParada>
- localEmbarque: string
- onibus: List<Onibus>
+ exibirRotas() -> List<Rota>
+ adicionarRota(rota: Rota) -> bool
+ editarRota(rota: Rota) -> bool
+ removerRota(rota: Rota) -> bool
+ consultarHorariosPontoFixo(pontoFixo: string, turno: Turno) -> List<Rota>
+ consultarHorarios(pontoFixo: string) -> List<Rota>
PontosParada
- id: int
- localizacao: string
operations
operation1
<<Enum>>
TipoDestino
+ Ida e Volta
+ Apenas Ida
+ Apenas Volta
Onibus
- capacidade: int
- placaOnibus: string
- coordenada: Tupla[latitude: float, longitude: float]
- assento: List<Assento>
- rota: Rota
+ exibirLocalizacao(placaOnibus: string) -> Tupla[latitude: string, longitude: string]
+ atualizarLocalizacao(coordenada: Tupla[latitude: float, logitude: float]) -> bool
+ definirCapacidade(capacidadeMaxima: int) -> bool
+ exibirLotacao(onibus: Onibus) -> int
+ atualizarLotacao() -> void
Catraca
- id : int
- codigo : string
- nome: string
- onibus : Onibus
- contadorPassageiros: int
+ validarEmbarque(codigo: string) -> bool
+ liberarAcesso() -> bool
ListaEspera
- id: int
- reserva: Reserva
- limiteCandidatos: int
- listaEstudantes: List<Estudante>
+ adicionarEstudante(estudante: Estudante, reserva: Reserva) -> bool
+ listarEstudantes() -> List[Estudante]
+ atualizarLista(vagaLiberada: Reserva) -> void
Assento
- numeroAssento: string
- statusAssento: bool
- onibus: Onibus
+ escolherAssento(numeroAssento: string) -> bool
+ visualizarAssentos(onibus: Onibus) -> List<Assento>
+ desfazerEscolhaDeAssento(assento: Assento) -> bool