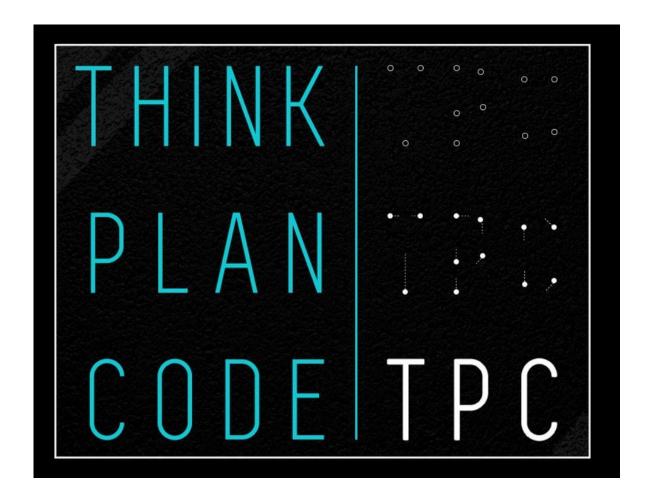


24 DE NOVEMBRO DE 2023



DOMAIN DRIVEN DESIGN GLOBAL SOLUTION

THINK, PLAN & CODE

BEATRIZ LUCAS - RM99104 | ENZO FARIAS - RM98792 | EWERTON GONÇALVES - RM98571 | GUILHERME TANTULLI - RM97890 | THIAGO ZUPELLI - RM99085



SUMÁRIO

Domain Driven Design	2
Objetivo e Escopo do projeto	2
Breve descrição das principais funcionalidades	3
Protótipo do Projeto	3
Diagrama Relational (ou modelo Relacional – Banco de dados)	6
Tabela dos Endpoints	7
Diagrama de classes	8
Procedimentos para rodar a aplicação	c



Domain Driven Design

Objetivo e Escopo do projeto

A saúde é a garantia de vida para as pessoas. Ter a possibilidade de cuidar de uma doença, antes mesmo que ela se espalhe dentro do organismo ou torne-se extremamente prejudicial faz a diferença em nossas vidas. Visando diminuir esses casos e com foco nas metas da ODS 3, o projeto Safe Vision foi concebido. Ele é uma plataforma criada para diagnósticos antecipados referentes ao Retinoblastoma, um tipo de câncer que afeta pessoas de todas as idades, mas principalmente crianças.

Dessa maneira, o Safe Vision emerge como uma solução pioneira, empregando inteligência artificial para identificar possíveis neoplasias oculares em crianças de até 5 anos. Oferecemos um diagnóstico prévio de fácil acesso à população, orientando o usuário, com base em seu convênio (caso o tenha), médicos e hospitais que podem auxiliar no tratamento da doença, de modo a direcionar ao cuidado da mesma de maneira direta. Com isso, a rede Hapvida terá pleno espaço e disponibilidade para implementar essas tecnologias nos programas de triagem, permitindo a detecção precoce e tratamento mais eficaz, reduzindo os altos custos quando a doença é descoberta de maneira tardia.



Breve descrição das principais funcionalidades

O objetivo principal é fornecer uma ferramenta que ajude na identificação preventiva com a utilização de inteligência artificial, onde a identificação é feita com acurácia a partir de um dataset disponibilizado na Kaggle com possíveis tumores oculares para ajudar as crianças de até 5 anos serem identificadas com algum tipo de neoplasias. Com isso, nossa solução tem como foco analisar e melhorar a identificação preventiva para os casos de neoplasias oculares, afim de ajudar na prevenção que não descarta o acompanhamento médico. Além disso, a plataforma será passível de ser utilizada por usuários de outras idades, sejam adolescentes, adultos ou idosos, de modo a disponibilizar a possível prevenção para a população como um todo.

Protótipo do Projeto









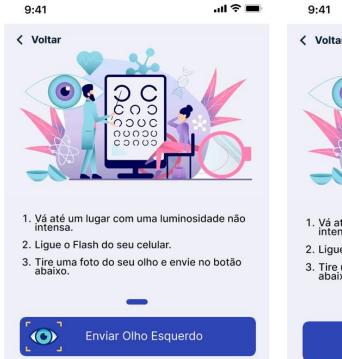








.ıl 🗢 🔳





Precisão: 94% Grau: 0.92 Indicação: Procure um médico o mais breve possivel.



Diagrama Relational (ou Modelo Relacional)

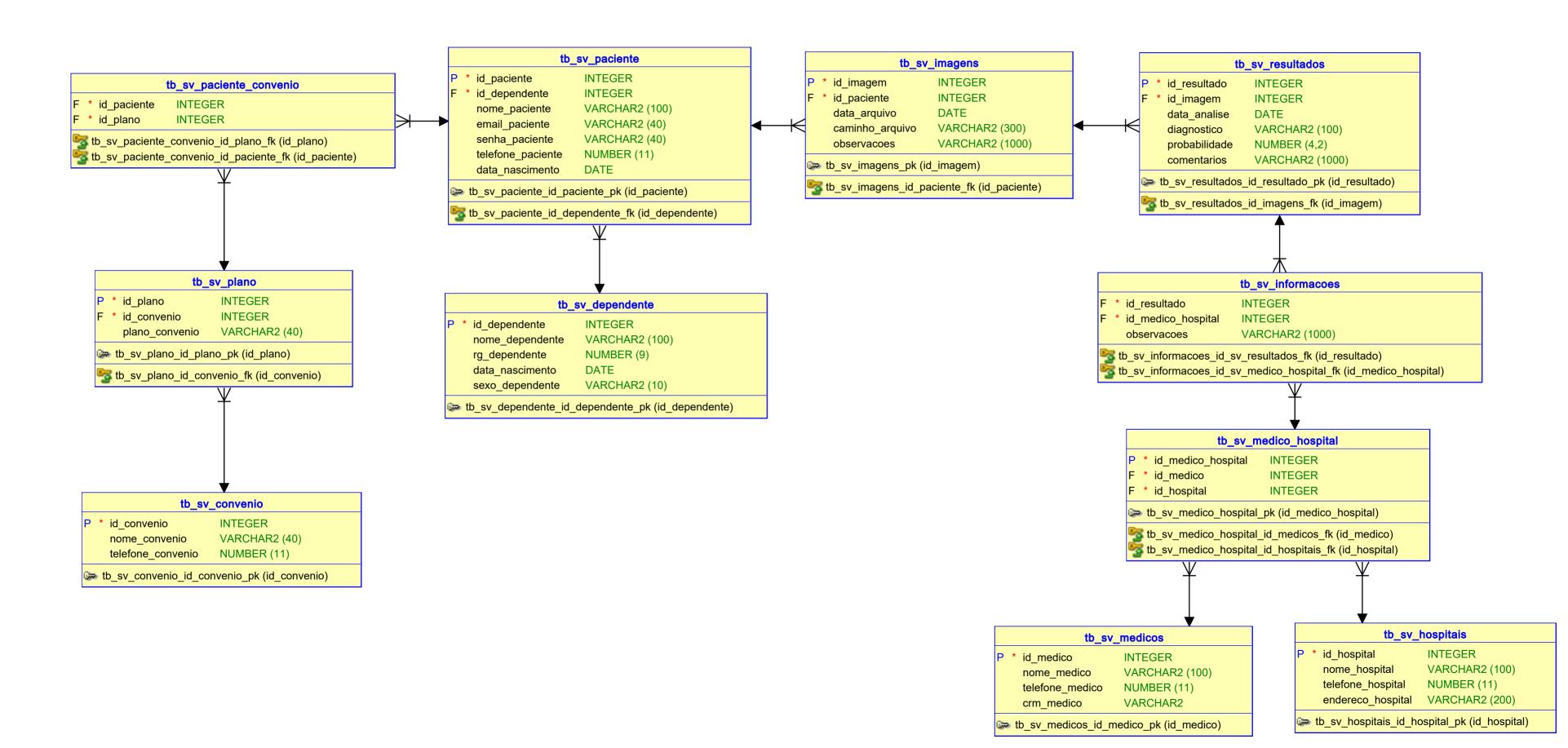




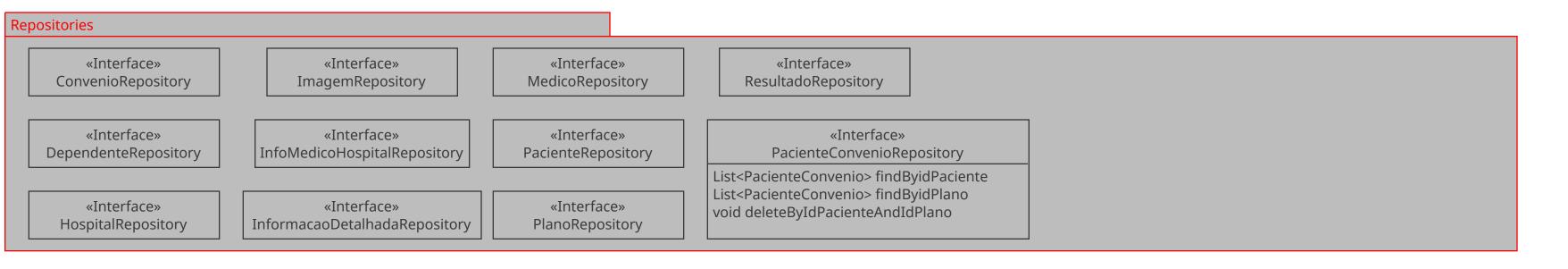
Tabela dos Endpoints

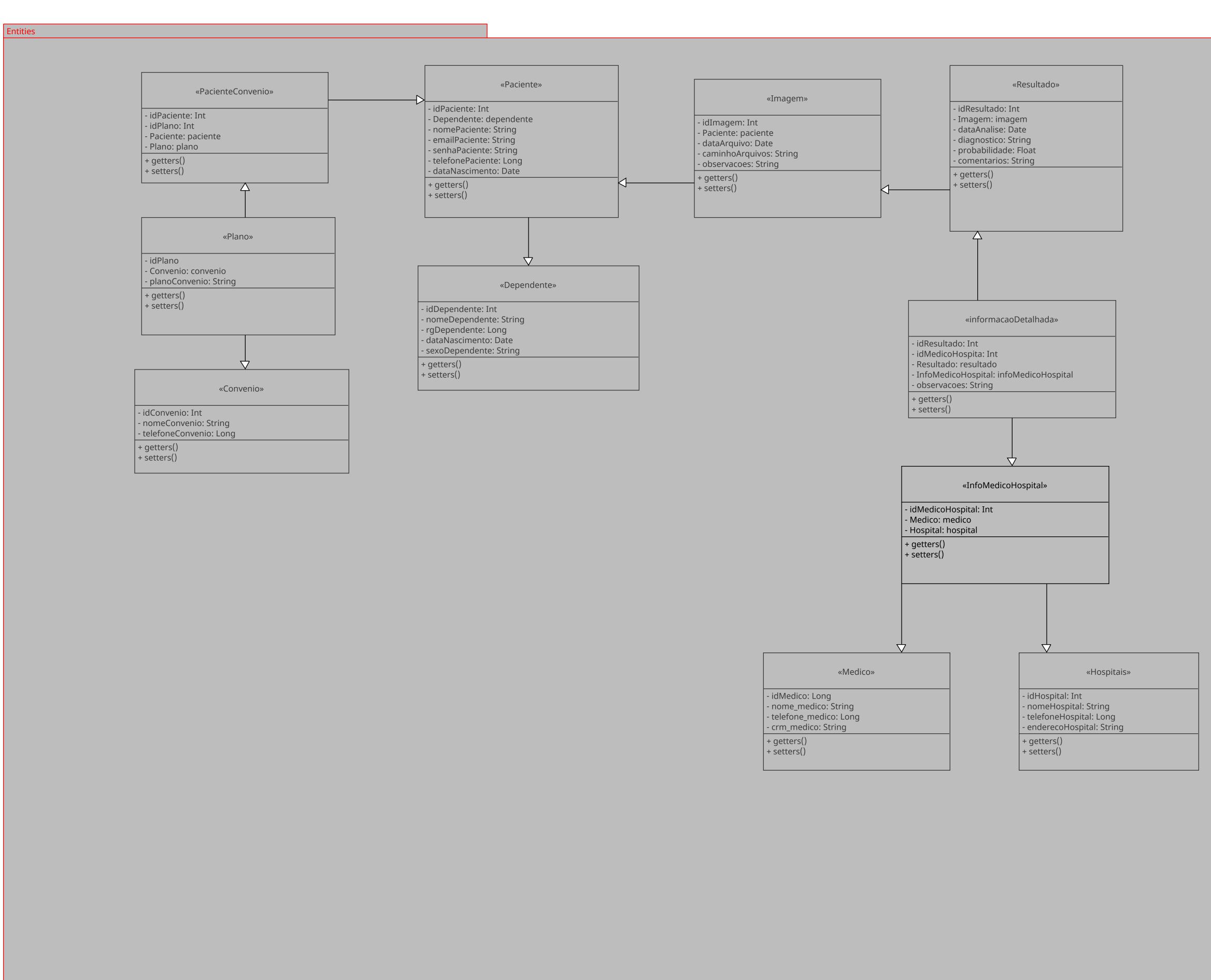
Tabela acessada	Endpoint
Convenio	/convenios
Dependente	/dependentes
Hospital	/hospitais
Imagem	/imagens
MedicoHospital	/info-medico-hospital
informacoes	/informacoes
medicos	/medicos
pacientes	/pacientes
pacientes-convenios	/pacientes-convenios
planos	/planos
resultados	/resultados

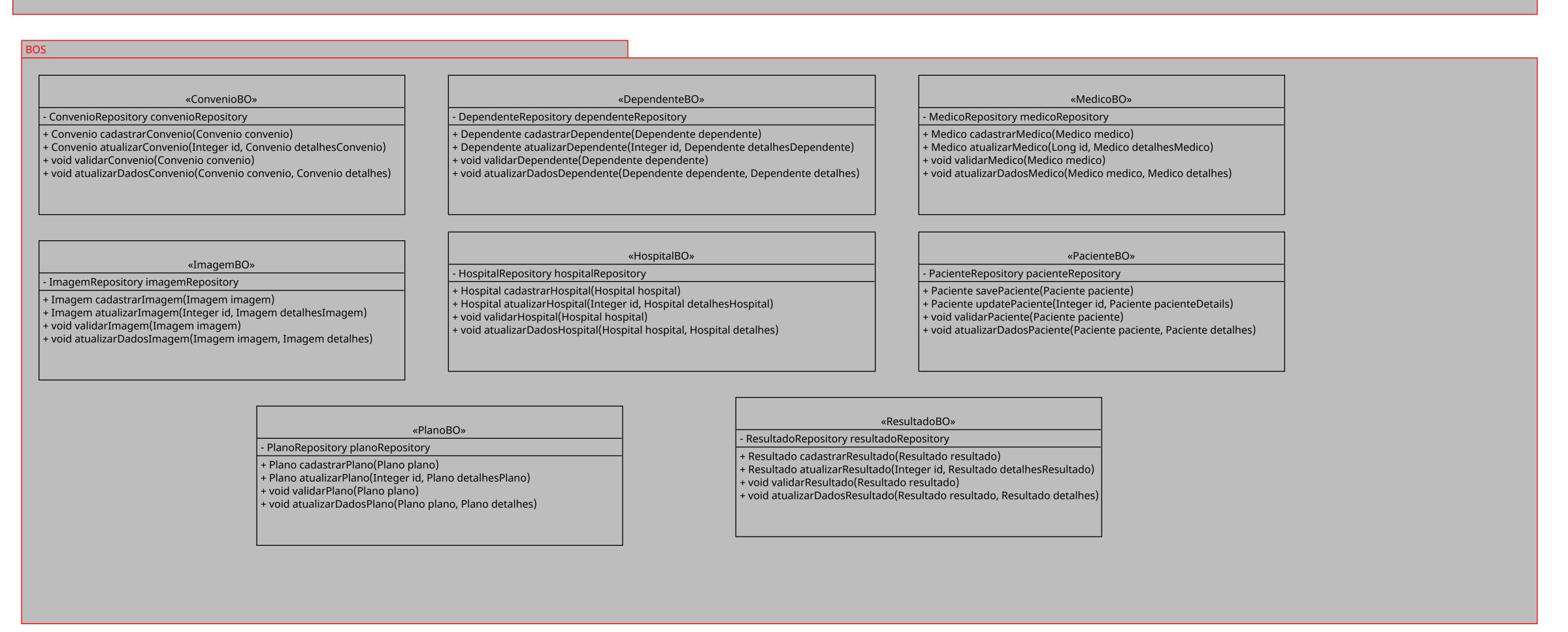
Em todos os endpoints, é possível acessar id's específicos para acessar apenas um dado em especial, como um veículo ou um cliente determinado. Para isso, basta adicionar "/id" após os endpoints acima, onde o id pode variar dentro dos dados armazenados em cada uma das tabelas.

Diagrama de Classes

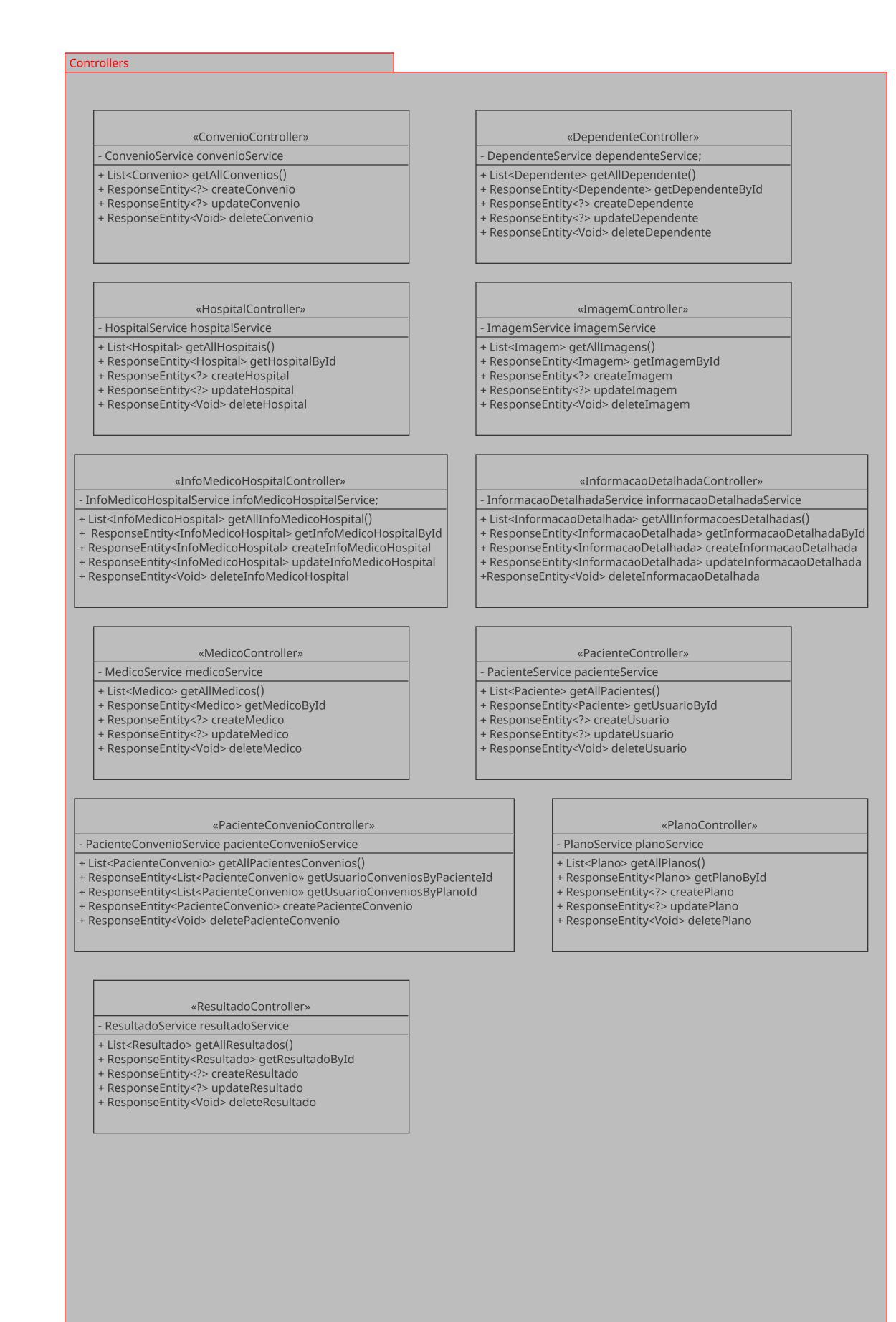
«ConvenioService» - ConvenioRepository convenioRepository; «DependenteService» ConvenioBO convenioBO; - DependenteRepository dependenteRepository + List<Convenio> getAllConvenios() - DependenteBO dependenteBO + Optional<Convenio> findConvenioById + Convenio saveConvenio (Convenio convenio) + List<Dependente> getAllDependentes() + Optional<Dependente> findDependenteById + Convenio updateConvenio + void deleteConvenio + Dependente saveDependente + Dependente updateDependentes + void deleteDependente «HospitalService» - HospitalRepository hospitalRepository «ImagemService» - HospitalBO hospitalBO - ImagemRepository imagemRepository + List<Hospital> getAllHospitais() - ImagemBO imagemBO + Optional<Hospital> findHospitalById + List<Imagem> getAllImagens() + Hospital saveHospital + Optional<Imagem> findImagemById + Hospital updateHospital + Imagem saveImagem + void deleteHospital + Imagem updateImagem + void deleteImagem «InfoMedicoHospitalService» «InformacaoDetalhadaService» InfoMedicoHospitalRepository infoMedicoHospitalRepository · InformacaoDetalhadaRepository informacaoDetalhadaRepository List<InfoMedicoHospital> getAllInfoMedicoHospital() + List<InformacaoDetalhada> getAllInformacoesDetalhadas() + Optional<InfoMedicoHospital> findInfoMedicoHospitalById + Optional<InformacaoDetalhada> findInformacaoDetalhadaById - InfoMedicoHospital saveInfoMedicoHospital + InformacaoDetalhada saveInformacaoDetalhada + InfoMedicoHospital updateInfoMedicoHospital + InformacaoDetalhada updateInformacaoDetalhada + void deleteInfoMedicoHospital + deleteInformacaoDetalhada «MedicoService» - MedicoRepository medicoRepository - MedicoBO medicoBO «PacienteConvenioService» + List<Medico> getAllMedicos() - PacienteConvenioRepository pacienteConvenioRepositor + Optional<Medico> findMedicoById + List<PacienteConvenio> getAllPacientesConvenios() + Medico saveMedico + List<PacienteConvenio> findByPacienteId + Medico updateMedico + List<PacienteConvenio> findByPlanoId + void deleteMedico + PacienteConvenio savePacienteConvenio + void deletePacienteConvenio «PacienteService» - PacienteRepository pacienteRepository - PacienteBO pacienteBO + List<Paciente> getAllPacientes() «PlanoService» + Optional<Paciente> findPacienteById - PlanoRepository planoRepository + Paciente savePacienteOptional - PlanoBO planoBO + Paciente updatePaciente + List<Plano> getAllPlanos() + void deletePaciente + Optional<Plano> findPlanoById + Plano savePlano + Plano updatePlano + void deletePlano «ResultadoService» ResultadoRepository resultadoRepository - ResultadoBO resultadoBO + List<Resultado> getAllResultados() + Optional<Resultado> findResultadoById + Resultado saveResultado + Resultado updateResultado + void deleteResultado













Procedimentos para rodar a aplicação

Para rodar a aplicação e colocar em atividade a API, basta dar play no arquivo GlobalsolutionApplication.java. Ele irá rodar o programa e iniciar o funcionamento da API, trazendo os dados já armazenados no banco de dados. É importante ressaltar que a API está configurada para rodar na porta 8090, logo, ela será aberta pelo caminho localhost:8090/. É importante ressaltar, também, que no arquivo application.properties, localizado em globalsolution/src/main/resources, foi configurado com o comando "spring.jpa.hibernate.ddl-auto=update", que faz com que os inserts, localizados em import.sql (na mesma pasta) não sejam inseridos automaticamente, de modo que apenas há a criação do banco de dados, sem a população do mesmo. Para que esse comportamento seja alterado e funcione de forma a inserir os dados no banco de dados, basta adicionar "#" no início da linha, de modo a comentar este comando e executar a criação do banco de dados, junto de sua população.