

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
PUC Minas Virtual

Pós-graduação *Lato Sensu* em Arquitetura de *Software Distribuído*

Projeto Integrado

Relatório Técnico

Inside Nutri

Michael Tadeu Alves de Oliveira

Belo Horizonte
Dezembro de 2022.

Projeto Integrado – Arquitetura de Software Distribuído

Sumário

Projeto Integrado – Arquitetura de Software Distribuído	2
1. Introdução	3
2. Cronograma do Trabalho	5
3. Especificação Arquitetural da solução	7
3.1 Restrições Arquiteturais	7
3.2 Requisitos Funcionais	7
3.3 Requisitos Não-funcionais	9
3.4 Mecanismos Arquiteturais	10
4. Modelagem Arquitetural	12
4.1 Diagrama de Contexto	12
4.2 Diagrama de Container	13
4.3 Diagrama de Componentes	15
5. Prova de Conceito (PoC)	19
5.1 Integrações entre Componentes	20
5.2 Código da Aplicação	24
6. Avaliação da Arquitetura (ATAM)	28
6.1 Análise das abordagens arquiteturais	28
6.2 Cenários	29
6.3 Evidências da Avaliação	31
6.4 Resultados Obtidos	46
7. Avaliação Crítica dos Resultados	48
8. Conclusão	51
Referências	52

1. Introdução

Uma das maiores influências negativas sobre as taxas de morbidade e mortalidade por diferentes causas em pacientes internados é o comprometimento do estado nutricional (LOPES et al.; 2009). A desnutrição eleva a chance de o paciente apresentar complicações durante o período de internação hospitalar e ainda pode determinar um maior tempo de recuperação e reabilitação e, consequentemente, a elevação do custo do sistema de saúde, além da piora da qualidade de vida do paciente (OLIVEIRA et al.; 2010).

A desnutrição pode acometer rapidamente o doente hospitalizado principalmente devido ao estado hipercatabólico que acompanha as patologias, a presença de traumatismos e infecções em resposta ao estresse metabólico que ocorre nestas condições, em especial quando a ingestão nutricional é insuficiente (MINISTÉRIO DA SAÚDE, 2016).

O Inquérito Brasileiro de Avaliação Nutricional - Ibranutri, realizado em pacientes internados em hospitais da rede pública, observou uma prevalência de desnutrição em 48,1% dos 4 mil pacientes internados e submetidos à avaliação nutricional. E em uma amostra de 709 pacientes, a incidência de complicações nos desnutridos foi de 27,0%, ocasionando um aumento nos custos hospitalares de 60,5% para desnutridos e 3 vezes mais mortalidade nesse grupo quando comparados aos pacientes bem nutridos (WAITZBERG; CAIAFFA; CORREIA, 1999).

Mesmo com os avanços em terapia nutricional nas últimas décadas, a desnutrição continua sendo frequente em pacientes hospitalizados, com prevalência variando entre 30 e 65% (LEITE; CARVALHO; MENESES, 2005). Desta forma, o cuidado nutricional adequado, possui efeitos benéficos na recuperação dos pacientes e melhora da sua qualidade de vida (GARCIA; PADILHA; SANCHES, 2012).

Para que quadros de desnutrição hospitalar sejam raros, os pacientes hospitalizados necessitam de uma ingestão nutricional adequada que forneça as necessidades básicas de manutenção do organismo e de recuperação da saúde. Assim, o papel do nutricionista clínico inclui realizar o atendimento dietoterápico com base no quadro clínico do paciente, avaliar fatores de risco relacionados, analisar o estado nutricional através de medidas antropométricas e exames laboratoriais para elaborar objetivos e metas para o cuidado nutricional, utilizando fórmulas matemáticas para o cálculo adequado de calorias necessárias, de acordo com as diretrizes mais atuais para cada patologia (CFN, 2018).

Logo, para que o trabalho do nutricionista clínico hospitalar seja otimizado, reduzindo a probabilidade de erros nos cálculos, o objetivo deste trabalho é apresentar a descrição do projeto arquitetural da aplicação Inside Nutri para realizar cálculos precisos e relevantes para os profissionais, sendo a partir da entrada de dados específicos, o valor calórico total diário para o paciente de acordo com a patologia, a quantidade de proteína, o volume total de dieta enteral a ser administrada e velocidade de infusão, fazendo com que a Terapia Nutricional seja efetiva, sendo uma ferramenta de auxílio para a redução da desnutrição hospitalar. Para alcançar o objetivo proposto, os seguintes objetivos específicos são necessários:

- Realizar levantamento na literatura das fórmulas matemáticas para os cálculos necessários;
- Realizar levantamento das recomendações nutricionais para cada patologia;
- Mapeamento das dietas enterais para cada patologia.

2. Cronograma do Trabalho

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

Datas		Atividade / Tarefa	Produto / Resultado
De	Até		
19 / 06 / 2022	19 / 06 / 2022	1. Cronograma de Trabalho	Tabela com o cronograma de trabalho
20 / 06 / 2022	22 / 06 / 2022	2. Brainstorming	Proposta do projeto a ser desenvolvido
23 / 06 / 2022	25 / 06 / 2022	3. Contextualização do Trabalho	Construção da Introdução deste trabalho, contendo o problema, motivação e objetivos
26 / 06 / 2022	30 / 06 / 2022	4. Elicitação das restrições Arquiteturais	Lista com as restrições arquiteturais identificadas
01 / 07 / 2022	05 / 07 / 2022	5. Elicitação dos Requisitos Funcionais	Lista com os requisitos funcionais identificados
06 / 07 / 2022	10 / 07 / 2022	6. Elicitação dos Requisitos Não-Funcionais	Lista com os requisitos não-funcionais identificados
11 / 07 / 2022	15 / 07 / 2022	7. Elicitação dos Mecanismos Arquiteturais	Lista com os mecanismos arquiteturais identificados
16 / 07 / 2022	20 / 07 / 2022	8. Elaboração do Diagrama de Contexto – Modelo C4	Diagrama de contexto do projeto proposto
21 / 07 / 2022	28 / 07 / 2022	9. Revisão do documento da Etapa 1	Documento revisado da Etapa 1 do projeto proposto
29 / 07 / 2022	05 / 08 / 2022	10. Elaborar apresentação da Etapa 1	Apresentação do projeto proposto
06 / 08 / 2022	10 / 08 / 2022	11. Gravar vídeo da Etapa 1	Vídeo gravado da Etapa 1 do projeto proposto
11 / 08 / 2022	14 / 08 / 2022	12. Publicar Etapa 1 no repositório do GitHub	Arquivos da etapa 1 disponibilizados no repositório do GitHub
15 / 08 / 2022	15 / 08 / 2022	13. Entrega da Etapa 1 no canvas da disciplina Projeto Aplicado	Etapa 1 concluída e enviada para avaliação
16 / 08 / 2022	29 / 08 / 2022	14. Construção do Diagrama de Container	Diagrama de container do projeto proposto
30 / 08 / 2022	10 / 09 / 2022	15. Construção do Diagrama de Componentes	Diagrama de componentes do projeto proposto
11 / 09 / 2022	23 / 09 / 2022	16. Desenho do Wireframe da POC	Protótipo de telas do projeto proposto
24 / 09 / 2022	04 / 10 / 2022	17. Código da aplicação	Aplicação com três requisitos implementados

Datas		Atividade / Tarefa	Produto / Resultado
De	Até		
05 / 10 / 2022	12 / 10 / 2022	18. Revisão do documento da Etapa 2	Documento revisado da Etapa 2 do projeto proposto
13 / 10 / 2022	14 / 10 / 2022	19. Publicar Etapa 2 no repositório do GitHub	Arquivos da etapa 2 disponibilizados no repositório do GitHub
15 / 10 / 2022	15 / 10 / 2022	20. Entrega da Etapa 2 no canvas da disciplina Projeto Aplicado	Etapa 2 concluída e enviada para avaliação
16 / 10 / 2022	29 / 10 / 2022	21. Análise das abordagens arquiteturais	Seção do documento produzido
30 / 10 / 2022	10 / 11 / 2022	22. Construção dos Cenários	Seção do documento produzido
11 / 11 / 2022	23 / 11 / 2022	23. Evidências da avaliação	Seção do documento produzido
24 / 11 / 2022	04 / 12 / 2022	24. Resultados obtidos	Seção do documento produzido
05 / 12 / 2022	10 / 12 / 2022	25. Avaliação crítica dos resultados	Seção do documento produzido
11 / 12 / 2022	13 / 12 / 2022	26. Conclusão	
14 / 12 / 2022	14 / 12 / 2022	27. Revisão do documento da Etapa 3	Documento revisado da Etapa 3 do projeto proposto
15 / 12 / 2022	15 / 12 / 2022	28. Produção do vídeo 2 na Etapa 3, apresentando o trabalho de forma completa	Vídeo gravado da Etapa 3 do projeto proposto
15 / 12 / 2022	15 / 12 / 2022	29. Publicar Etapa 2 no repositório do GitHub	Arquivos da etapa 3 disponibilizados no repositório do GitHub
15 / 12 / 2022	15 / 12 / 2022	30. Entrega da Etapa 2 no canvas da disciplina Projeto Aplicado	Etapa 3 concluída e enviada para avaliação

3. Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permitem visualizar a macroarquitetura da solução.

3.1 Restrições Arquiteturais

A qualidade do projeto arquitetural e das descrições arquiteturais são fortemente influenciadas por restrições arquiteturais. Seu impacto na qualidade da arquitetura engloba propriedades relativas à qualidade do sistema de software em construção, bem como o processo de design, incluindo sua eficiência e suas perspectivas. Assim, para esse projeto foi proposto restrições arquiteturais que devem ser satisfeitas, sendo:

ID	Descrição
RA01	Deve ser usado tecnologias <i>Open source</i> para o desenvolvimento de toda a aplicação Inside Nutri.
RA02	Deve ser usado o serviço de nuvem da <i>Amazon Web Services</i> (AWS) ou <i>Microsoft Azure</i> como provedora da infraestrutura necessária para a aplicação Inside Nutri.
RA03	Deve ser usado o serviço <i>OAuth 2.0</i> do Google bem como a possibilidade de criação de conta diretamente na aplicação Inside Nutri, para o gerenciamento de autenticação dos usuários.
RA04	Deve ser implementado a aplicação Inside Nutri utilizando uma solução móvel (<i>Apps Mobile</i>) que deverá suportar os sistemas operacionais móveis mais populares, a saber Android e IOS.
RA05	Deve ser implementado uma <i>API RESTful</i> para prover todos os dados e comunicação com os clientes de forma agnóstica e desacoplada com o backend, facilitando uma possível mudança tecnológica no frontend e a utilização de microserviços na aplicação Inside Nutri.
RA06	Implementar <i>RabbitMQ</i> como <i>event broker</i> , sendo necessário a comunicação orientada a eventos na aplicação Inside Nutri.
RA07	A comunicação na aplicação Inside Nutri de origem externa ao ambiente deve passar por um <i>API Gateway</i> .
RA08	O registro de log's deve ser feito utilizando a ferramenta slack ELK para CI/ CD da aplicação Inside Nutri.

3.2 Requisitos Funcionais

Os requisitos de um sistema de software correspondem às descrições do que esse sistema deve fazer, aos serviços que ele oferece e às restrições em sua funcionalidade. Tais requisitos representam a demanda dos clientes desse sistema (SOMMERVILLE, 2019; CHITCHYAN et al., 2005). Em geral, os requisitos são classificados como (SOMMERVILLE, 2019): i) Requisitos Funcionais (RF) e (ii) Requisitos Não-Funcionais.

Os Requisitos Funcionais descrevem funções que o sistema deve fornecer, como ele deve reagir às entradas específicas e como ele deve se comportar em determinadas situações. A seguir, são apresentados os Requisitos Funcionais da aplicação Inside Nutri.

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	A aplicação Inside Nutri deve permitir o auto cadastramento do usuário nutricionista.	B	A
RF02	A aplicação Inside Nutri deve permitir o auto cadastramento do usuário nutricionista através de uma conta do Google.	B	A
RF03	A aplicação Inside Nutri deve atender as orientações estabelecidas pela LGPD.	A	A
RF04	A aplicação Inside Nutri deve listar as funcionalidades na tela principal.	B	A
RF05	A aplicação Inside Nutri deve permitir que o nutricionista possa realizar uma avaliação nutricional.	M	A
RF06	A aplicação Inside Nutri deve permitir que o nutricionista calcule as seguintes métricas nutricionais, sendo Índice de Massa Corporal, Peso Ideal, Peso Ajustado, Peso Corrigido, Peso Estimado, Adequação de Peso, Altura Estimada e Gasto Energético Total	M	A
RF07	A aplicação Inside Nutri deve permitir que a partir dos dados obtidos no RF06 o nutricionista possa calcular o volume total de dieta enteral, hidratação e velocidade de infusão (Bomba de Infusão Contínua ou Infusão Intermitente) para cada patologia.	M	A
RF08	A aplicação Inside Nutri deve permitir que o nutricionista gere um relatório com todas as informações obtidas através dos cálculos.	M	M
RF09	A aplicação Inside Nutri deve permitir o nutricionista cadastrar a dieta enteral utilizada pela unidade de hospitalar, caso não encontre cadastrada na aplicação.	A	A
RF10	A aplicação Inside Nutri deve permitir a importação de dados de dietas enterais de fornecedores regionais.	M	M
RF11	A aplicação Inside Nutri deve permitir que o nutricionista cadastre o paciente, com os dados de nome, data de nascimento, sexo e raça/ cor.	B	A
RF12	A aplicação Inside Nutri deve permitir que o nutricionista possa trocar de senha.	B	B

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF13	A aplicação Inside Nutri deve permitir que o nutricionista visualize uma listagem dos pacientes cadastrados.	B	M
RF14	A aplicação Inside Nutri deve permitir que o nutricionista altere alguma informação dos pacientes cadastrados.	B	B
RF15	A aplicação Inside Nutri deve calcular a idade dos pacientes a partir da data de nascimento.	M	A
RF16	A aplicação Inside Nutri deve realizar os cálculos com base nas informações fornecidas de cada paciente, por exemplo, caso seja de uma certa idade e raça/ cor irá calcular o Peso Estimado de acordo com uma formula específica.	M	A
RF17	A aplicação Inside Nutri deve permitir que o nutricionista visualize a porcentagem de energia fornecida através da dieta enteral em comparação ao gasto energético total.	M	A
RF18	A aplicação Inside Nutri deve permitir que o nutricionista visualize a porcentagem de proteína fornecida através da dieta enteral em comparação às proteínas totais diárias.	M	A
RF19	A aplicação Inside Nutri deve permitir que o nutricionista visualize a porcentagem de água fornecida em comparação à hidratação diária necessária.	M	A
RF20	A aplicação Inside Nutri deve permitir que o nutricionista envie os cálculos por e-mail.	B	B
RF21	A aplicação Inside Nutri deve permitir que o administrador possa realizar todas as funcionalidades e ainda descredenciar nutricionistas, caso esteja divulgando informações dos pacientes.	B	M

*B=Baixa, M=Média, A=Alta.

3.3 Requisitos Não-funcionais

Os Requisitos Não-Funcionais (RNF), representam restrições impostas às funções oferecidas pelo sistema ou ao processo de desenvolvimento. A seguir, são apresentados os Requisitos Não-Funcionais da aplicação Inside Nutri.

ID	Descrição	Prioridade B/M/A
RNF01	A aplicação Inside Nutri deve habilitar a autenticação baseado no modelo <i>OAuth2</i> do <i>Google</i> e diretamente no sistema.	A
RNF02	O sistema deve operar em tempo integral em 24h x 7d x 365, tendo disponibilidade mínima de 90%, e para atender essa necessidade deverá usar a hospedagem AWS ou <i>Microsoft Azure</i> com mecanismo de escala automática para responder ao aumento de demanda.	A
RNF03	A comunicação entre o sistema Back-End, Front-End e Mobile da aplicação Inside Nutri deve ser implementado através do padrão de serviços REST.	A
RNF04	O sistema deverá utilizar para persistência em um Banco de Dados NoSQL para atender a necessidade de alta performance na resposta em alta demanda, aceitando em contrapartida uma integridade eventual. Porém, pode ser necessário a utilização de um Banco de Dados Relacional em determinados Microsserviços.	A
RNF05	As notificações enviadas da aplicação Inside Nutri por e-mail ou <i>push</i> devem operar por meio de filas de mensagens por não haver a necessidade de serem em tempo real e atenderem a grande demanda.	A
RNF06	A versão Web da aplicação Inside Nutri deve suportar os navegadores modernos, tais como Google Chrome, Microsoft Edge e Mozilla Firefox.	M
RNF07	Deve ser registrado uma trilha de auditoria para todas as alterações que ocorrerem no cadastro de clientes, cadastro de refeições e cadastro de dietas.	M
RNF08	O deploy em produção da aplicação Inside Nutri e a infraestrutura devem ser automatizadas usando pipelines de CI/CD.	M
RNF09	As API's devem possuir um <i>throughput</i> de no mínimo 300 tps.	A
RNF10	O processamento de grandes quantidades de dados deve ser feito preferencialmente entre 0hs e 4hs.	M

3.4 Mecanismos Arquiteturais

Esta seção deve apresentar uma visão geral dos mecanismos que compõem a arquitetura do software, baseando-se em três estados: (1) análise, (2) design e (3) implementação. Em termos de Análise devem ser listados os aspectos gerais que compõem a arquitetura do software, como: persistência, integração com sistemas legados, geração de logs do sistema, ambiente de front end, tratamento de exceções, formato dos testes, formato de distribuição/implantação (deploy), dentre outros. Em Design deve-se identificar o padrão tecnológico a seguir para cada mecanismo identificado na análise. Em Implementação deve-se identificar o produto a ser utilizado na solução.

Os Mecanismos arquiteturais são soluções comuns para problemas comuns que podem ser usados durante o desenvolvimento para minimizar a complexidade. Eles representam os principais conceitos técnicos que serão padronizados em toda a solução. Facilitam a evolução de aspectos arquiteturais significativos do sistema. Eles permitem que a equipe mantenha uma arquitetura coesa enquanto permitem que os detalhes da implementação sejam adiados até que eles realmente precisem ser feitos.

Assim, os Mecanismos Arquiteturais são usados para satisfazer requisitos arquitetonicamente significativos. Normalmente, esses são requisitos não funcionais, como problemas de desempenho e segurança. Quando totalmente descritos, os Mecanismos Arquiteturais mostram padrões de estrutura e comportamento no software. Eles formam a base do software comum que será aplicado de forma consistente em todo o produto que está sendo desenvolvido. Eles também formam a base para padronizar a maneira como o software funciona, portanto, eles são um elemento importante da arquitetura geral do software. A definição de mecanismos de arquitetura também permite decisões sobre se os componentes de software existentes podem ser aproveitados para fornecer o comportamento necessário ou se um novo software deve ser comprado ou construído. Com isso, segue os mecanismos arquiteturais da aplicação Inside Nutri.

Análise	Design	Implementação
Persistência	ORM	NHibernate
Persistência	Banco de Dados NoSql	MongoDB ou Amazon Dynamo DB
Front-End	Single Page Application	Angular
Front-End	Mobile	Kotlin
Back-End	Arquitetura em Camadas	.Net Core
Gateway	API Gatway	AWS API Gateway
Integração	API RestFull	APIs externas a aplicação Inside Nutri
Log do sistema	Gestão de Logs da aplicação	Slack ELK
Teste de Software	Testes de Unidade	JUnit ou XUnit
Versionamento	Repositório contendo o código fonte	GitHub
Deploy	Integração e Entrega continua (CI/CD)	AWS Code Pipelines
Mensagens	Mensageria e/ou notificação de push	Amazon Pinpoint e/ou RabbitMQ
API	Documentação das APIs da aplicação Inside Nutri	Swagger
Autenticação	OAuth 2.0	Open Authorization 2.0

4. Modelagem Arquitetural

Esta seção apresenta a modelagem arquitetural da solução proposta, de forma a permitir seu completo entendimento visando à implementação da prova de conceito (PoC) da aplicação Inside Nutri, descrito na seção 5 deste trabalho.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Mais informações a respeito podem ser encontradas aqui: <https://c4model.com/> e aqui: <https://www.infoq.com/br/articles/C4-architecture-model/>. Dos quatro nível que compõem o modelo C4 três serão apresentados aqui e somente o Código será apresentado na próxima seção (5).

4.1 Diagrama de Contexto

O diagrama de contexto modelo C4 é usada para descrever e definir arquiteturas de forma abstrata e simples. O modelo C4 é uma maneira de apresentar a macroarquitetura da solução que aborda o desenvolvimento de software, apresentando em sua modelagem a concentração em quatro c's: (i) contexto (pessoas); (ii) contêineres; (iii) componentes; e (iv) código. (BROWN, 2020). Para a aplicação proposta temos o seguinte diagrama de contexto:

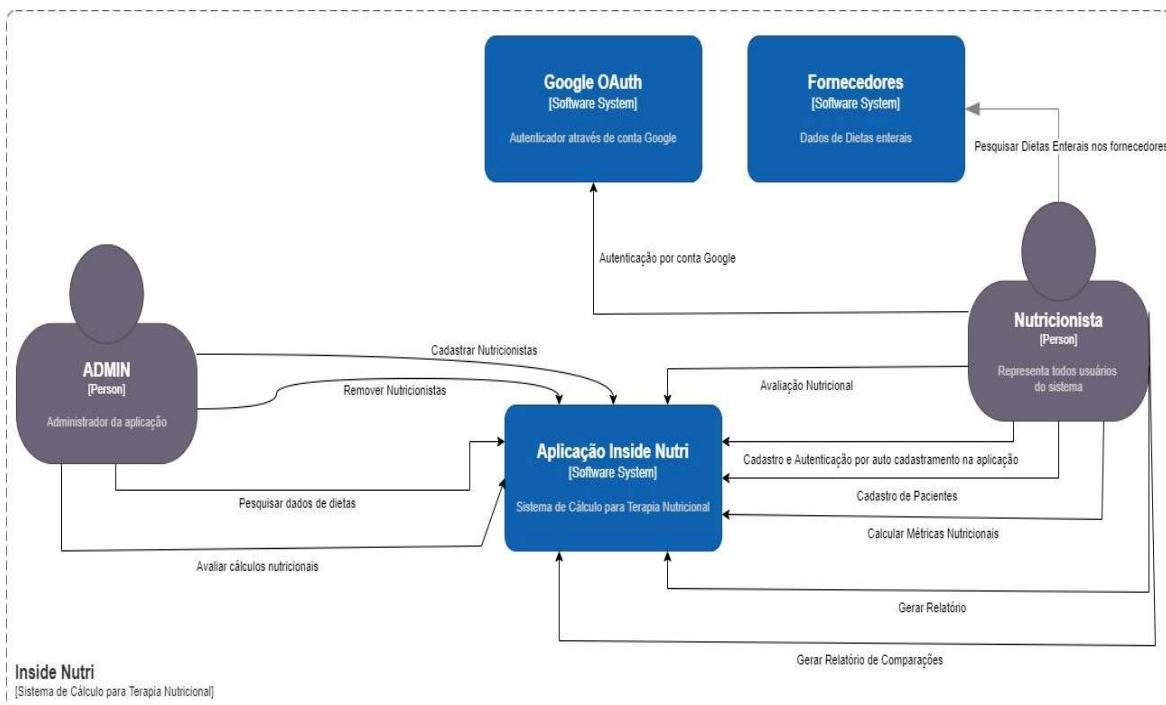


Figura 1 - Visão Geral da Solução Inside Nutri. Fonte: Do Autor.

A Figura 1 - **Visão Geral da Solução Inside Nutri.** **Fonte: Do Autor** mostra a especificação o diagrama geral da solução proposta, com todos seus principais módulos e suas interfaces, tais como a integração da aplicação Inside Nutri com a autenticação do OAuth2.0 e com sistemas de fornecedores via API. Podemos verificar os dois atores da aplicação, sendo o administrador e o nutricionista. É possível verificar as funcionalidades que cada ator poderá realizar na aplicação.

4.2 **Diagrama de Container**

O diagrama *Container* mostra a forma de alto nível da arquitetura de software e como as responsabilidades são distribuídas por ela. Ele também mostra as principais opções de tecnologia e como os *containers* se comunicam entre si. É um diagrama simples e focado em tecnologia de alto nível que é útil tanto para desenvolvedores de software quanto para equipes de suporte/operações. Para a aplicação proposta temos o seguinte Diagrama de *Container*:

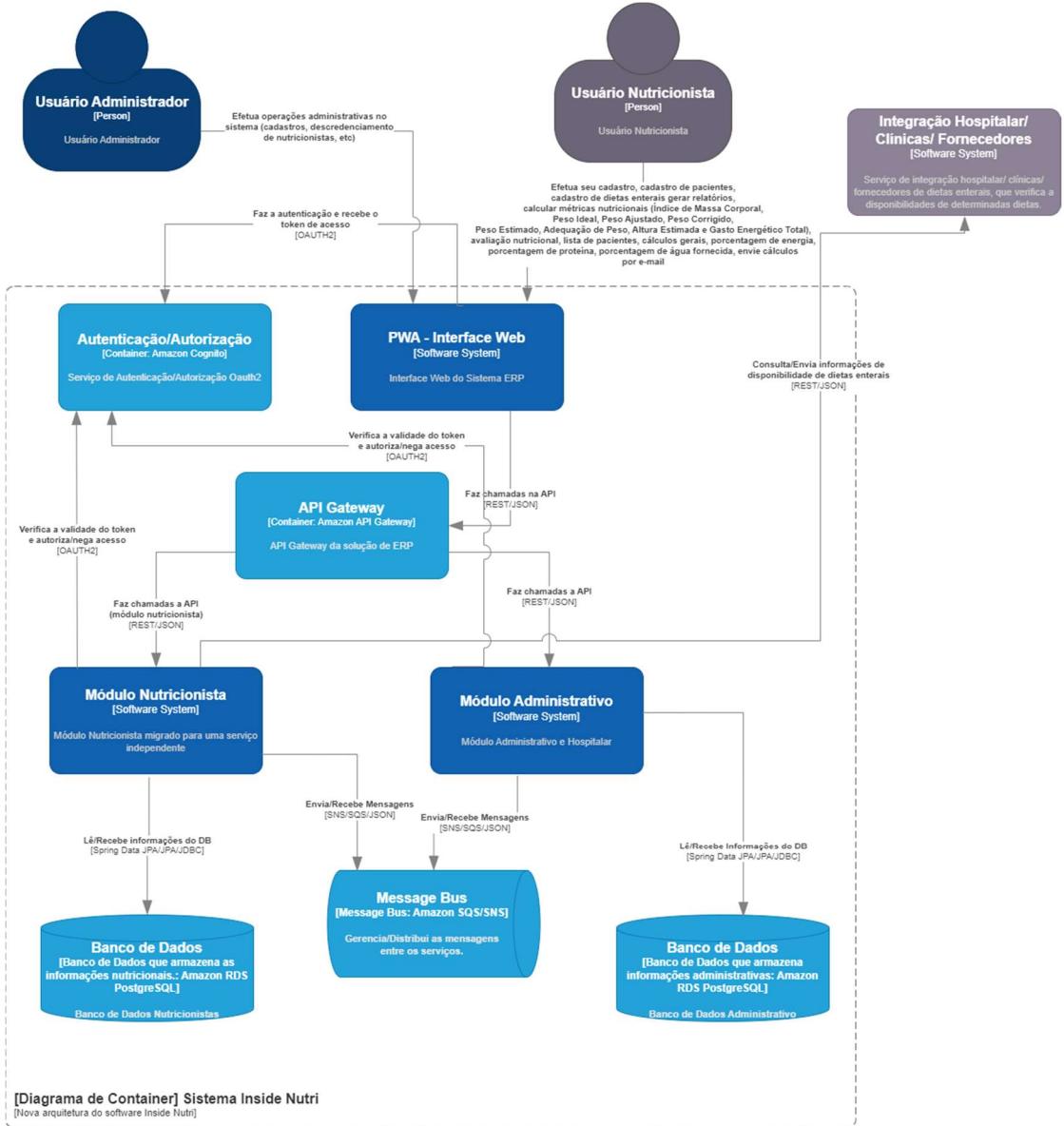


Figura 2 – Diagrama de Container. Fonte: Do Autor.

A Figura 2 – **Diagrama de Container. Fonte: Do Autor** apresenta os containers da aplicação Inside Nutri de forma amplificada os containers que compõem os módulos Nutricionista e Administrativo, onde em cada *container* é evidenciado sua natureza, bem como a tecnologia que é usada e também os protocolos de integração com os quais eles interagem entre si. A página web é onde são servidos os conteúdos estáticos (HTML, CSS e JavaScript) fornecido pelo NGINX que compõem a aplicação web feita em Angular que por sua vez será renderizada aos usuários via navegador web.

Existem 2 papéis, para utilização do sistema, o usuário do Administrador e o usuário do módulo Nutricionista. No dia a dia pode acontecer do mesmo usuário assumir ambos os papéis, mas para fins de explicação, foi escolhido detalhar como figuras distintas.

O usuário faz o login na aplicação frontend em Angular. A aplicação frontend, em sua tela de login, requisita um *token* de acesso para o serviço o OAUTH2, que é fornecido pelo Amazon Cognito. Com o *token* de acesso disponível, a aplicação web é autorizada a fazer requisições.

As requisições são feitas através do API Gateway, utilizando a tecnologia Amazon API Gateway. Dependendo da requisição, o API Gateway redireciona as requisições para o módulo nutricionista ou módulo administrativo.

Tanto o módulo nutricionista quanto o módulo administrativo possuem bancos de dados independentes. A comunicação entre os módulos será feita via mensageria, funcionando assim de forma desacoplada e utilizando as tecnologias Amazon SQS (Fila) e Amazon SNS (tópicos). E para requisições web dos usuários, as aplicações, de forma independente, fazem a checagem do *token* de acesso com o servidor OAUTH2 (Amazon Cognito).

Toda e qualquer autenticação e autorização será feita diretamente no API Gateway, devido características internas e tecnológica. O módulo nutricionista comunica-se via interface REST com o Integração Hospitalar/ Clínicas/ Fornecedores, que é um serviço externo de uma empresa parceira. O Integração Hospitalar/ Clínicas/ Fornecedores, possui uma interface hospitalar, clínicas ou fornecedores de dietas enterais, que verifica a disponibilidades de determinadas dietas na região.

4.3 Diagrama de Componentes

Um diagrama de componentes, também conhecido como diagrama de componentes UML, descreve a organização e a fixação dos componentes físicos em um sistema. Os diagramas de componentes geralmente são desenhados para ajudar a modelar os detalhes da implementação e verificar novamente se todos os aspectos das funções exigidas do sistema são cobertos pelo desenvolvimento planejado. Para a aplicação proposta temos o seguinte Diagrama de Componentes:

Inside Nutri

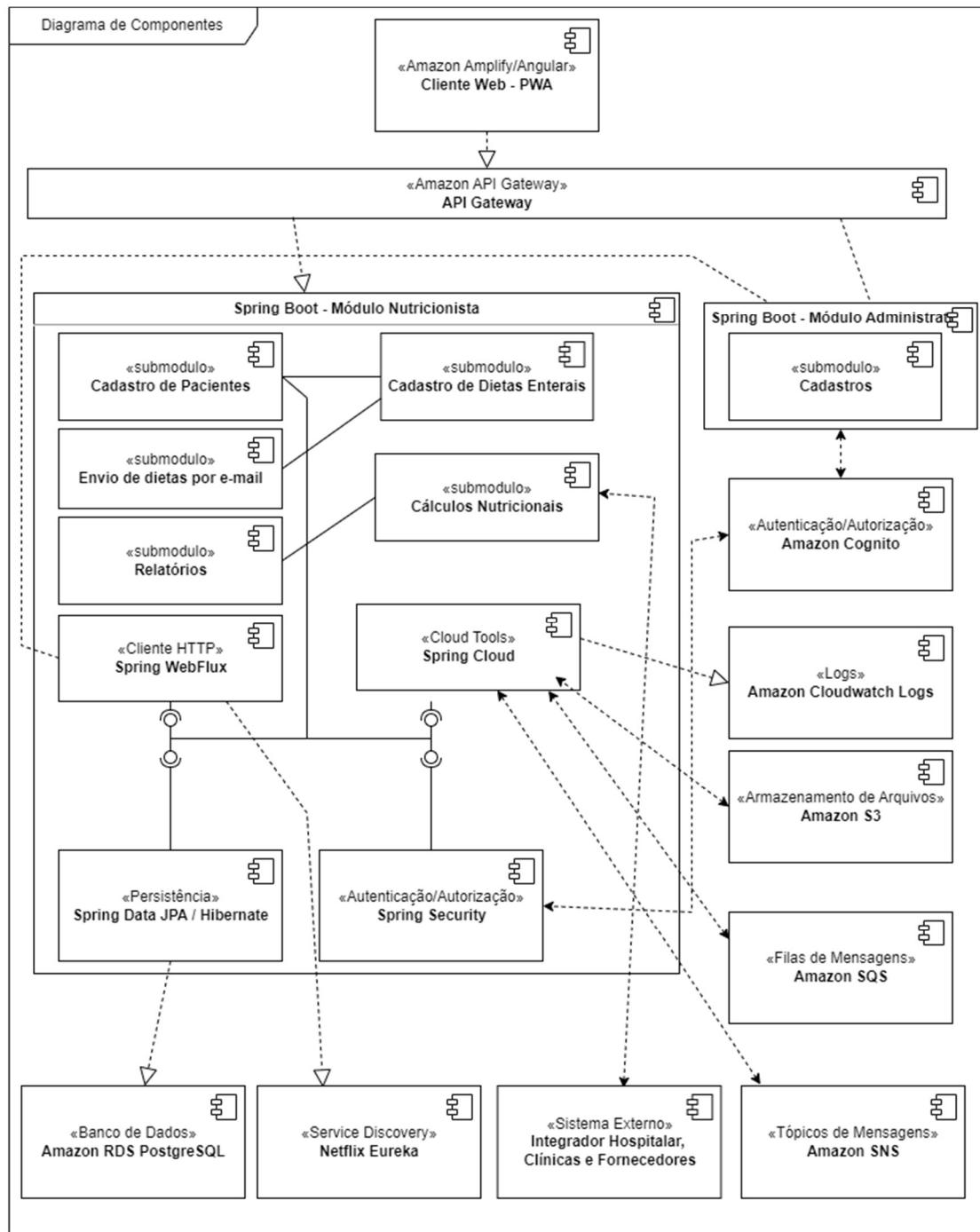


Figura 3 – Diagrama de Componentes. Fonte: Do Autor.

A **Figura 3 – Diagrama de Componentes. Fonte: Do Autor.** mostra o Diagrama de Componentes do sistema de software Inside Nutri, seguindo o padrão UML. Descrevendo suas dependências com mais detalhes da base tecnológica da aplicação. Os componentes da solução:

- **Cliente Web:** É uma aplicação desenvolvida em Angular, utilizando o pacote de componentes angular e também a biblioteca AWS Amplify para a integração com o serviço Amazon Cognito (servidor OAuth2).

- **API Gateway:** É um serviço Amazon API Gateway como solução de API Gateway, que é utilizado para unificar a API dos serviços do Módulo Nutricionista e Módulo Administrativo em um único *endpoint*;
- **Integração Hospitalar/ Clínicas/ Fornecedores:** serviço externo de uma empresa parceira responsável pela integração de dietas enterais. A comunicação é realizada via disponibilização de uma API REST para fazer essa integração;
- **Amazon RDS PostgreSQL:** É o servidor de banco de dados do módulo nutricionista. Assim, a utilização do serviço de bancos de dados é gerenciada pela Amazon, o Amazon RDS. É utilizado a engine de banco de dados PostgreSQL;
- **Netflix Eureka:** Serviço de Service Discovery é um dos principais princípios da arquitetura baseada em microservices. Tentando configurar a mão para cada cliente ou alguma forma de convenção. O Eureka é o Service Discovery do Netflix que é usado no Server e no Cliente;
- **Amazon Cognito:** É um servidor OAUTH2, responsável por realizar as autenticações e autorizações dos usuários do sistema de software Inside Nutri;
- **Amazon S3:** É um serviço de armazenamento de objetos (arquivos) da Amazon;
- **Amazon SQS:** É um *Amazon Simple Queue Service* (SQS) é um serviço de filas de mensagens gerenciado;
- **Amazon SNS:** É um serviço de notificação da Amazon. Com sua utilização é possível criar tópicos de mensagens. É uma solução para padronizar as mensagens em pub/sub de mensagens;
- **Amazon Cloudwatch Logs:** É utilizado para o armazenamento de logs dos serviços;
- **Módulo Administrativo:** É um módulo que tem funcionalidades específicas para manter o sistema de software Inside Nutri executando com alta disponibilidade e o mínimo de manutenção;

- **Módulo Nutricionista:** É um módulo principal do sistema de software Inside Nutri que é a solução proposta nesse trabalho. Utiliza como base o *framework* ABP Framework. Tendo como principais funcionalidades:
 - **Cadastro de Pacientes:** É um submodulo dentro do Módulo Nutricionista, responsável pelo cadastro dos pacientes que cada nutricionista está atendendo e acompanhando;
 - **Cadastro de Dietas Enterais:** É um submódulo dentro do Módulo Nutricionista, responsável pelo cadastro das dietas enterais para cada paciente atendido e acompanhado pelos nutricionistas;
 - **Envio de Dietas por E-mail:** É um submódulo dentro do Módulo Nutricionista, responsável pelo envio das dietas enterais por e-mail para cada paciente cadastrado;
 - **Relatórios:** É um submódulo dentro do Módulo Nutricionista, responsável por gerar relatórios qualitativos e quantitativos dos atendimentos e/ ou pacientes, bem como das dietas enterais;
 - **Cálculos Nutricionais:** É um submódulo dentro do Módulo Nutricionista, responsável pela parte mais importante do sistema de software Inside Nutri. Este realiza todos os cálculos necessários para os nutricionistas definir quais e quanto de dieta enteral cada paciente necessita e por quanto tempo;
 - **API Controllers:** é uma biblioteca do responsável por consultas HTTP REST API endpoint;
 - **Cloud:** É uma biblioteca utilizada para auxiliar no gerenciamento de serviços cloud, tais como, Amazon Web Services. Nessa solução, servirá para realizar a comunicação com o Amazon SQS e SNS. E também será possível comunicar com o serviço Amazon S3. Além disso, será utilizado para disponibilizar o serviço de Service Discovery utilizando o Netflix Eureka para o API;
 - **Entity Framework Core:** O EF (Entity Framework) Core é uma versão leve, extensível, de software livre e multiplataforma da popular tecnologia de acesso a dados do Entity Framework. O EF

Core pode servir como um O/RM (mapeador relacional de objeto), que: (i) Permite que os desenvolvedores do .NET trabalhem com um banco de dados usando objetos .NET; e (ii) Elimina a necessidade da maior parte do código de acesso a dados que normalmente precisa ser gravado. O EF Core é compatível com vários mecanismos de banco de dados.

5. *Prova de Conceito (PoC)*

Uma prova de conceito (PoC) é o primeiro passo no processo de desenvolvimento de software após o desenvolvimento da ideia geral do produto. Uma PoC visa validar a viabilidade do projeto e verificar a viabilidade geral da ideia. O objetivo do PoC, do ponto de vista do desenvolvimento de software, é mostrar a viabilidade tanto da ideia do produto quanto do plano de negócios. Assim, uma prova de conceito devidamente apresentada ajuda a evitar erros dispendiosos e facilita a atração de investidores.

Para a aplicação proposta temos aspectos importantes para a arquitetura e que foram contemplados, sendo:

- **Persistência de dados:** O objetivo dessa PoC foi avaliar o Entity Framework Core a fim de entender sua implementação e funcionamento, buscando comprovar sua performance que é considerada uma vantagem em relação a outras bibliotecas semelhantes. Espera-se que com a utilização da biblioteca seja possível obter maior controle na performance de querys;
- **Design pattern para API:** O objetivo dessa PoC foi comprovar que a arquitetura em camadas alinhada a conceitos de DDD (Domain-Driven Design) pode organizar bem o código diante da grande quantidade de regras de negócio que compõem a aplicação. Espera-se que seja possível incluir, retirar ou modificar regras de negócio de forma simples e rápida;

- **Angular:** O objetivo dessa POC foi estudar a utilização do framework Angular para desenvolvimento do front-end. Espera-se que com a utilização desse framework seja possível desenvolver um sistema responsivo, performático e que não dependa de muitos componentes de terceiros para seu funcionamento.

5.1 Interações entre Componentes

A PoC foi desenvolvida a partir de requisitos prioritários coerentes a especificação proposta e implementada de maneira eficaz afim de permitir a validação das integrações entre os componentes.

Cadastro de Usuários:

- **Componentes e Middlewares Envolvidos:** Portal Web (Angular) e API de autenticação;
- **Interações e Protocolos de Comunicação:** Comunicação com a API RESTful de usuários (JSON);
- **Requisitos Não-Funcionais:** Segurança e Usabilidade.

The screenshot shows the 'Cadastro de Usuário' (User Registration) page of the Inside Nutri application. The page has a teal header with the 'Inside Nutri' logo. Below the header, there's a teal decorative graphic with wavy patterns. The main form area has a white background. It contains five input fields: 'Nome' (Name), 'E-mail' (Email), 'CRN' (CRN), 'Senha' (Password), and 'Confirmar Senha' (Confirm Password). Each field is preceded by a small circular icon with a symbol: a person for Name, an envelope for Email, a ruler for CRN, a lock for Password, and a lock for Confirm Password. Below the inputs is a large teal button labeled 'Cadastrar' (Register). A horizontal line with the text 'OR' in the center separates this from a smaller link 'Realizar Login' (Log In) with a login icon.

Figura 4 - Tela de Cadastro de Usuários. Fonte: Do Autor.

Acesso (Login) de Usuários:

- **Componentes e Middlewares Envolvidos:** Portal Web (Angular);
- **Interações e Protocolos de Comunicação:** Comunicação via OAuth2 e Comunicação com Banco de Dados via TCP/JDBC;
- **Requisitos Não-Funcionais:** Segurança, Usabilidade, Compatibilidade.

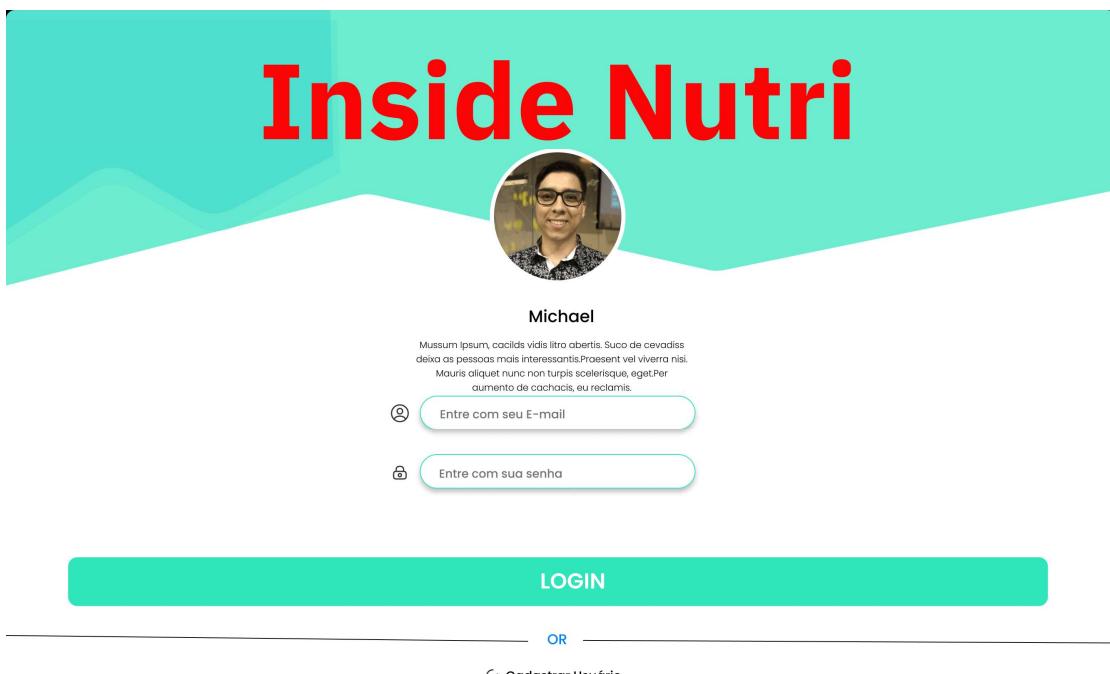


Figura 5 - Tela de Login. Fonte: Do Autor.

Tela Inicial (Dashboard):

- **Componentes e Middlewares Envolvidos:** Portal Web (Angular);
- **Interações e Protocolos de Comunicação:** Comunicação com a API de paciente, atendimentos e dietas via RESTful (JSON);
- **Requisitos Não-Funcionais:** Segurança, Usabilidade, Compatibilidade.

The screenshot shows the Inside Nutri dashboard. On the left, there's a vertical sidebar with a profile picture of Michael and a teal header bar. The main area has a white background with various sections:

- Bem Vindo Michael!** (Welcome Michael!)
- Out 04, 2022**
- Estatísticas do aplicativo** (App Statistics):
 - 12 Dietas Prescritas**
 - 12 Pacientes Atendidos**
 - 8 Pacientes Aguardando**
- Consultas** (Consultations):
 - João** Paciente (Patient) Out 4, 2022 @12:30 PM - 1:30 PM
- Dietas Disponíveis** (Available Diets):
 - 6 Dietas com Proteínas**
 - 4 Dieta com Glicose**
 - 6 Dietas com Eletrólitos**
 - 2 Dietas com Vitaminas**
- Quadro de Status** (Status Table):

Paciente	Dieta	Duração	Quantidade	Status
Maria	Vitaminas	6 meses	250 ml	Aguardando Retorno
José	Sais Minerais	3 meses	50 ml	Aguardando Retorno
Pedro	Proteínas	1 mês	150 ml	Aguardando Retorno
Joaquim	Eletrólitos	2 meses	100 ml	Aguardando Retorno
Orlando	Vitaminas	4 meses	50 ml	Aguardando Retorno
Cristina	Vitaminas	7 dias	250 ml	Agendando Retorno

Figura 6 - Tela Inicial (Dashboard). Fonte: Do Autor.

Tela de Cadastro de Nova Dieta do Usuário:

- Componentes e Middlewares Envolvidos:** Portal Web (Angular);
- Interações e Protocolos de Comunicação:** Comunicação com a API de paciente via RESTful (JSON), Comunicação com a API de dieta via RESTful (JSON) e Comunicação com Banco de Dados via TCP/JDBC;
- Requisitos Não-Funcionais:** Segurança, Usabilidade, Compatibilidade.

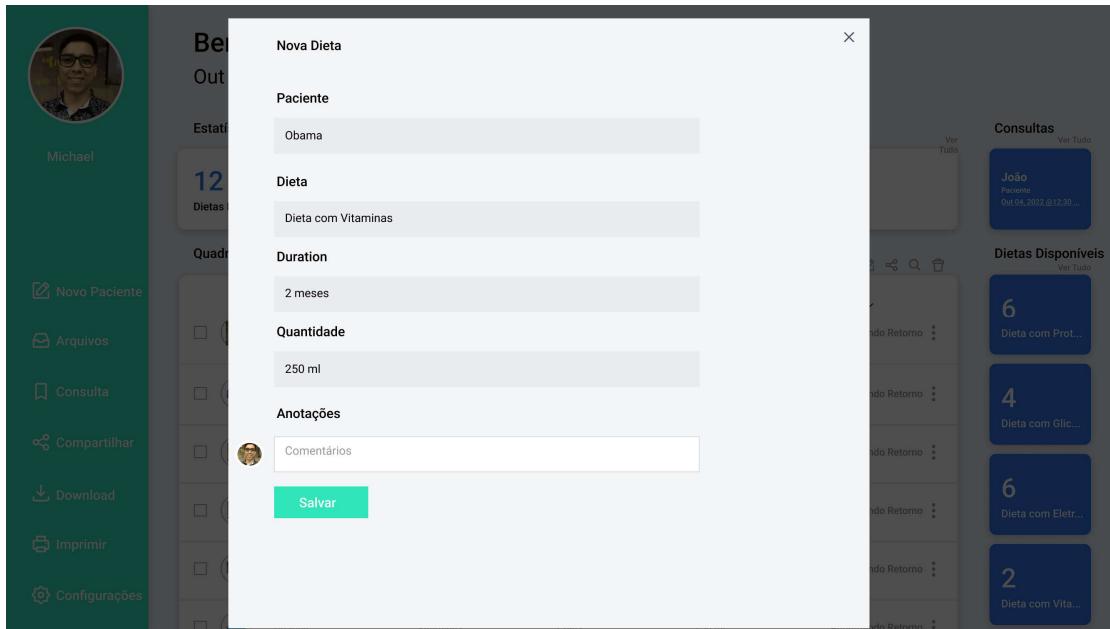


Figura 7 - Tela de Cadastro de Nova Dieta do Usuário. Fonte: Do Autor.

Tela de Realizar Cálculos de Dieta:

- **Componentes e Middlewares Envoltos:** Portal Web (Angular);
- **Interações e Protocolos de Comunicação:** Comunicação com a API de paciente via RESTful (JSON), Comunicação com a API de dieta via RESTful (JSON), Comunicação com a API de cálculos via RESTful (JSON) e Comunicação com Banco de Dados via TCP/JDBC;
- **Requisitos Não-Funcionais:** Segurança, Usabilidade, Compatibilidade.

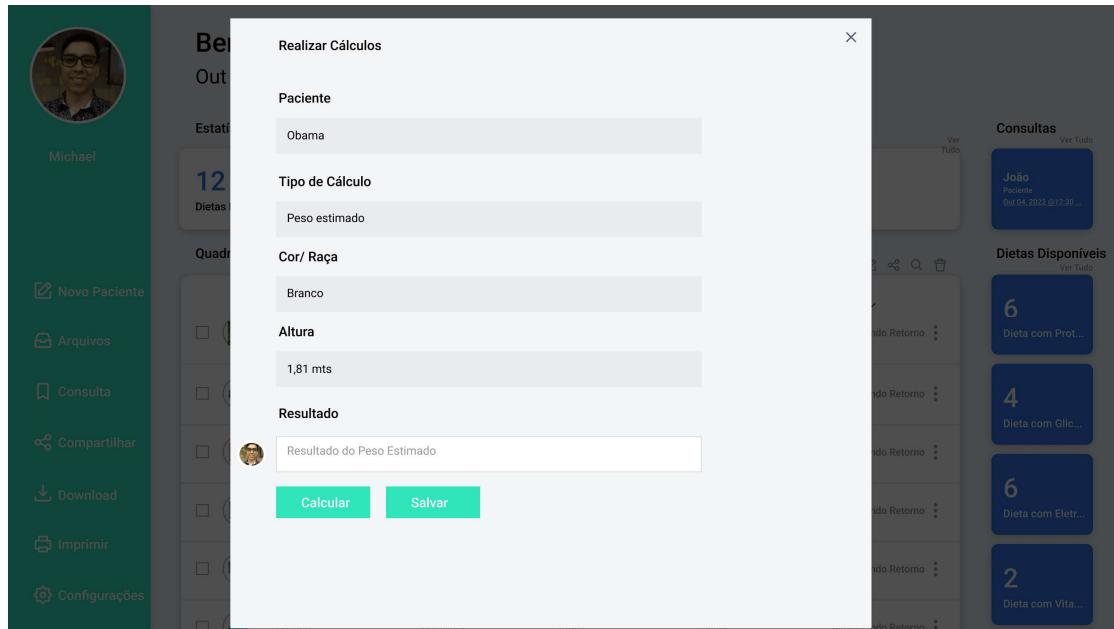


Figura 8 - Tela de Realizar Cálculos de Dieta. Fonte: Do Autor.

5.2 Código da Aplicação

Nessa sessão será explicado a nível de código o funcionamento dos requisitos escolhidos. O código fonte completo da aplicação pode ser acessado no endereço: <https://github.com/michaelTadeu/projeto-aplicado>:

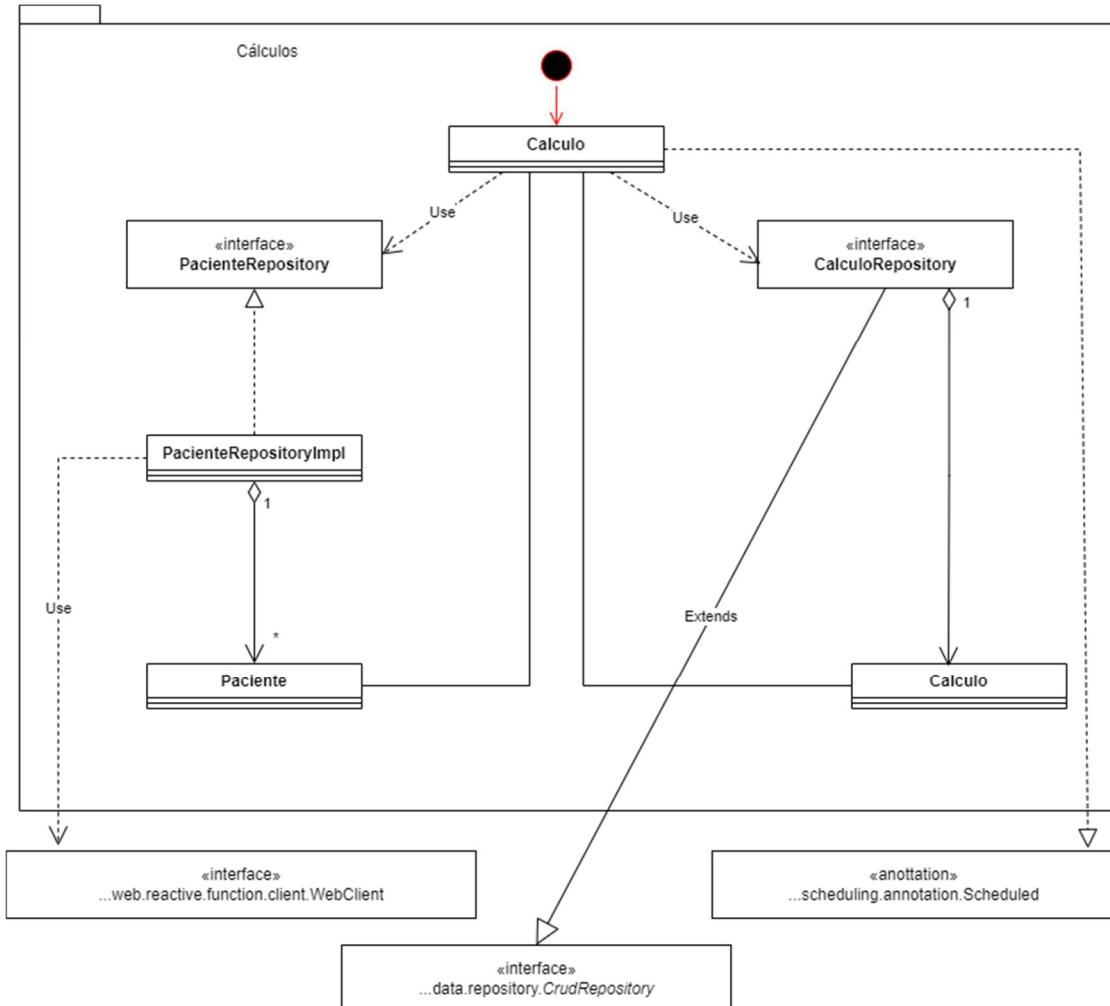


Figura 9 - Código do fluxo para realizar Cálculos. Fonte: Do Autor.

Na Figura 9 - **Código do fluxo para realizar Cálculos.** Fonte: Do Autor. é detalhado o funcionamento dos Cálculos. A classe Cálculo é uma *task* agendada, configurada com a anotação *Scheduled*. São consultadas no banco de dados todas os cálculos possíveis de serem feitos através da interface *CalculoRepository*, que utiliza o *CrudRepository* do Entity Framework Core para gerar uma implementação de um *Repository* em tempo de execução. Após a consulta dos dados, são enviados os cálculos para o respectivo paciente via classe *PacienteRepository*. Assim, os cálculos realizados para cada paciente são armazenados e ficam disponíveis para o nutricionista consultar posteriormente.

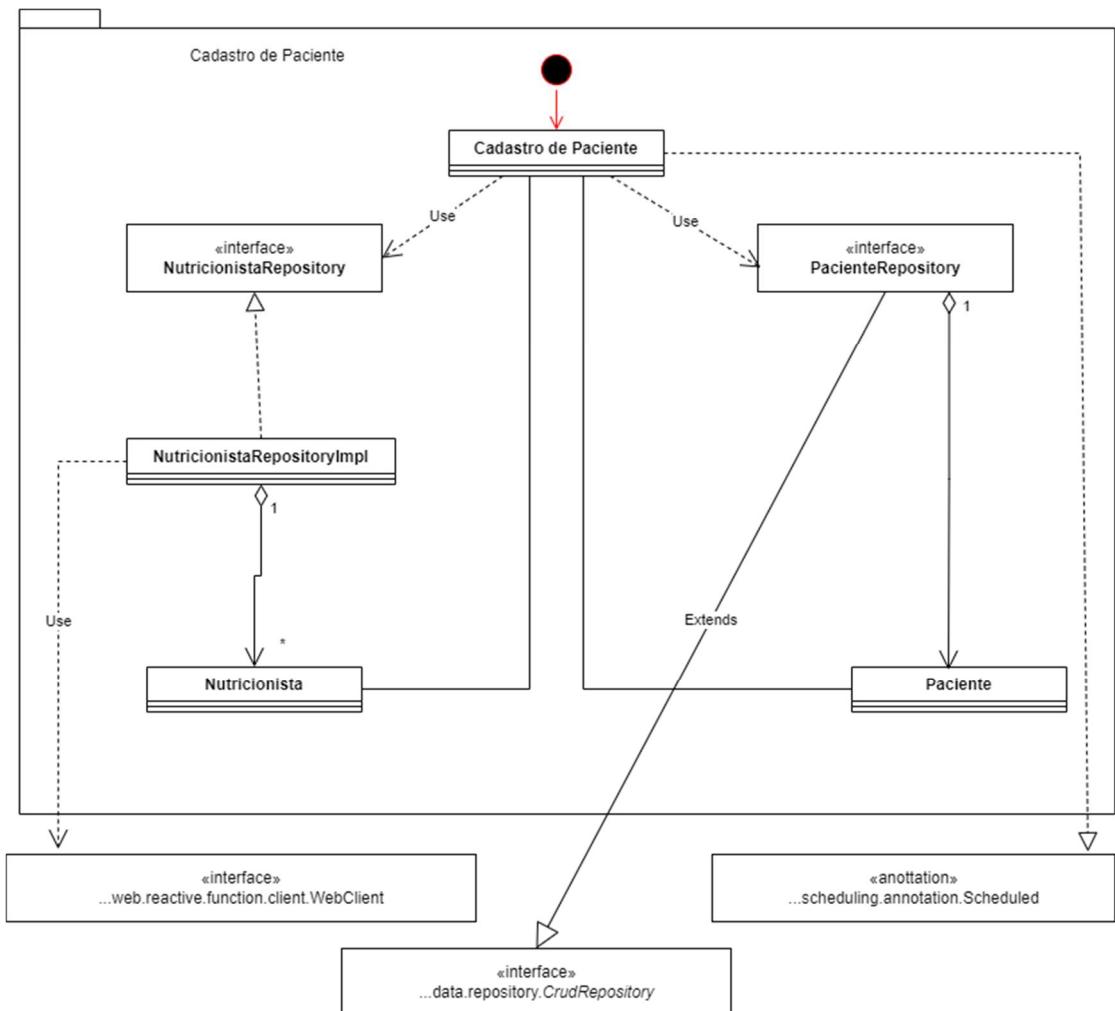


Figura 10 - Código do fluxo para realizar Cadastro de Pacientes. Fonte: Do Autor.

Na Figura 10 - **Código do fluxo para realizar Cadastro de Pacientes. Fonte: Do Autor.** é detalhado o funcionamento do Cadastro de Pacientes. A classe Cadastro de Pacientes consulta os dados do nutricionista no banco de dados através da interface NutricionistaRepository, que utiliza o CrudRepository do Entity Framework Core para gerar uma implementação de um Repository em tempo de execução. Após a consulta dos dados, são enviados os dados dos pacientes para a via classe PacienteRepository. Assim, os pacientes cadastrados são armazenados e ficam atrelados aos nutricionistas que efetuou o cadastro.

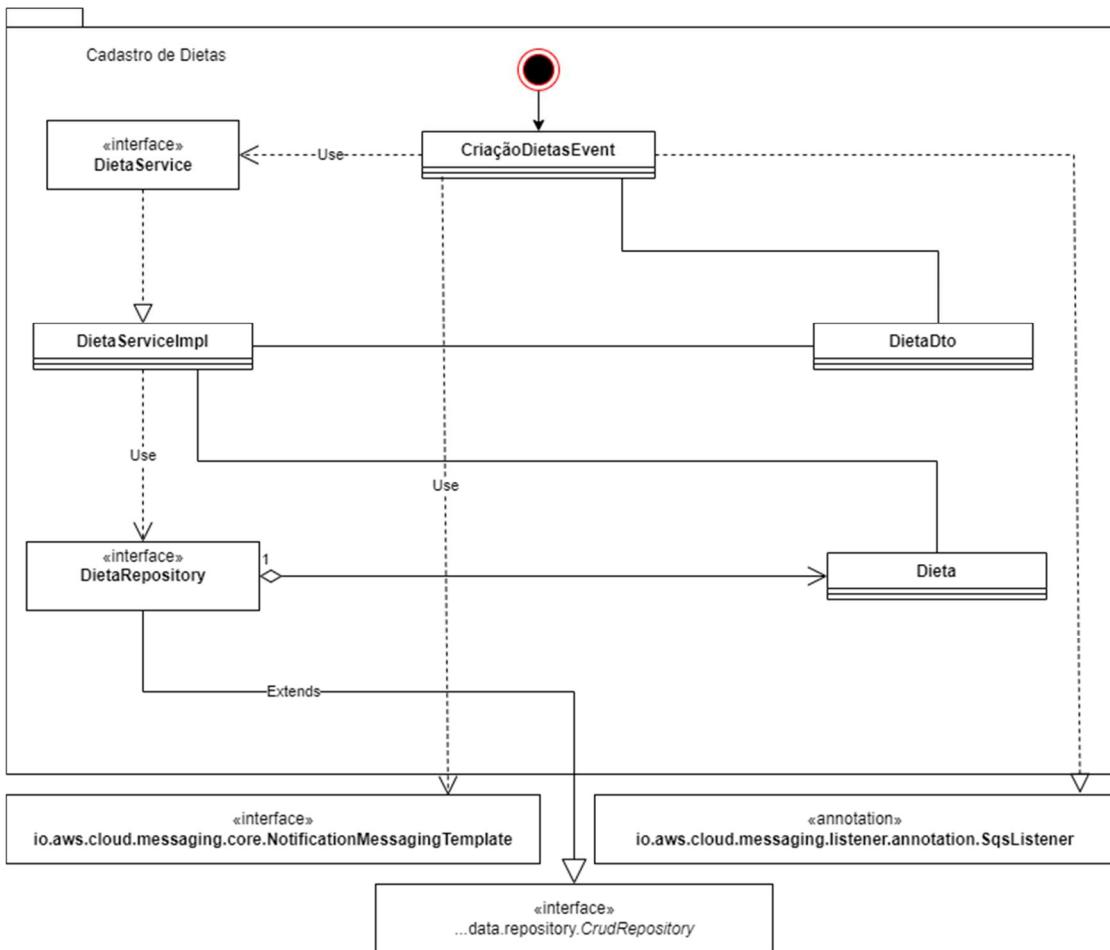


Figura 11- Código do fluxo para realizar Cadastro de Dietas. Fonte: Do Autor.

A classe CriacaoDietasEvent é um listener de uma fila SQS, e é anotada com a annotation SqsListener do Cloud. Essa classe recebe a mensagem no formato JSON, que segue a estrutura do DietaDto. Com o DietaDto preenchido, é utilizada a interface DietaService, e sua implementação, a classe DietaServiceImpl, para iniciar o processo de salvar. O DietaService é a classe de negócios do Cadastro de Dietas. O DietaDto é convertido para a classe de módulo Dieta, e é persistida no banco de dados com a interface que DietaRepository, uma interface de implementação do Entity Framework Core. Após a persistência, o fluxo volta para a CriacaoDietasEvent, onde é enviada a resposta de sucesso ou falha via tópico do Amazon SNS utilizando a classe NotificationMessagingTemplate do Cloud. Com isso é criado (ou não) o Cadastro da Dieta é notificado todos os serviços que são inscritos no tópico SNS correspondente.

6. Avaliação da Arquitetura (ATAM)

Esta seção apresenta a avaliação da arquitetura desenvolvida para a aplicação Inside Nutri, seguindo o método *Architecture Trade-off Analysis Method* (ATAM). O foco do método é identificar padrões arquiteturais que afetam um conjunto de atributos de qualidade predefinidos (DE OLIVEIRA ROSA et al.; 2020).

6.1. Análise das abordagens arquiteturais

A proposta da análise arquitetural tem como característica componentes modulares, cujos atributos de qualidade são apresentados através dos cenários para análise referentes aos requisitos informados na seção 3.3 Requisitos Não-funcionais. Com isso, segue os atributos de qualidade avaliados da aplicação Inside Nutri () .

Atributos de Qualidade	Cenários	Importância	Complexidade
Segurança	Cenário 1: O sistema deve prover segurança no acesso à APIs privadas, exigir devida autenticação e autorização do usuário para acesso a funcionalidades que não são públicas.	A	A
Desempenho	Cenário 2: O sistema deve prove boa performance nas operações funcionais, seguir padrão temporal satisfatório e agradável, não transparecer travamentos e sempre proporcionar feedback ao usuário em relação a consultas.	A	M
Padrão	Cenário 3: O sistema deve seguir padrão modular, possuir baixo acoplamento e alta coesão em suas camadas.	A	A
Confiabilidade	Cenário 4: O sistema deve ser resiliente e possuir rastreio de eventos distribuídos para detecção de gargalos ou indisponibilidade em determinadas integrações.	A	B

Atributos de Qualidade	Cenários	Importância	Complexidade
Usabilidade	Cenário 5: O sistema deve prover interface responsiva, aplicar um design adaptativo para melhorar a disposição do conteúdo na tela do dispositivo melhorando a experiência do usuário.	M	A
Compatibilidade	Cenário 6: O sistema deve funcionar corretamente nos navegadores web modernos mais usados (Google Chrome, Microsoft Edge e Firefox).	B	B
Interoperabilidade	Cenário 7: O sistema deve seguir a restrição Hypermedia As the Engine Of Application State (HATEOAS) nas comunicações baseadas em REST e comunicar com sistemas de outras tecnologias.	B	A

6.2. Cenários

Esta seção apresenta os cenários utilizados na realização dos testes da aplicação Inside Nutri. Os cenários de testes apresentados demonstram os requisitos não funcionais (atributos de qualidade) sendo satisfeitos.

Cenário 1 – Segurança: Ao acessar a página pública inicial do sistema o usuário não precisa estar autenticado, porém para acessar as páginas privadas e interagir com outras funcionalidades do sistema, o usuário obrigatoriamente deve possuir uma conta registrada e estar devidamente autenticado. As APIs privadas devem ser seguras e usar o padrão JSON Web Token (JWT), onde um acesso direto as URL's das mesmas exigirão que um token do tipo Bearer, válido, temporário e com escopo bem definido seja informado no cabeçalho Authorization da requisição HTTP. Caso o token não seja informado ou tenha expirado, a resposta da requisição HTTP deve retornar erro 401 – Unauthorized. Caso o token seja válido, mas o escopo do mesmo não possua o papel de acesso permitido para acesso a uma funcionalidade em específico, a resposta da requisição HTTP deve retornar erro 403 – Forbidden.

Cenário 2 – Desempenho: Ao acessar a funcionalidade para registro de ocorrência e após o preenchimento do formulário de cadastro de uma nova ocorrência, o processo de registro da mesma no sistema deve estar dentro de um padrão temporal satisfatório obedecendo um tempo médio de 3 segundos. Além disso, funcionalidades que exigem consulta devem sempre informar ao usuário um feedback visual de que o sistema está realizando algum processamento, exibindo uma imagem ou texto de carregando.

Cenário 3 – Padrão: Ao haver necessidade de implantar ou escalar um módulo deve ser possível que o procedimento seja feito de forma individual e ao necessitar que ocorra uma manutenção em um módulo somente parte do sistema deve ficar indisponível temporariamente enquanto o restante do mesmo deve estar operacional. Portanto, o padrão arquitetural do sistema deve ser o de microserviços, onde o sistema como um todo deve ser composto por uma coleção de serviços/módulos dividido de forma organizada e coesa em relação às suas capacidades funcionais, sendo independentes e com baixo acoplamento para ser altamente manejável e testável.

Cenário 4 – Confiabilidade: Ao acessar a URL do serviço de rastreio de eventos distribuídos, o mesmo deve retornar métricas com dados de tempo de requisições entre outras informações pertinentes para que seja possível detectar problemas de latência ou indisponibilidade de comunicação.

Cenário 5 – Usabilidade: Ao acessar o sistema em diferentes dispositivos com diferentes dimensões, as telas devem possuir comportamento responsivo se adaptando e melhorando a disposição do conteúdo na tela sem perda de funcionalidades.

Cenário 6 – Compatibilidade: Ao acessar o sistema em diferentes navegadores web como Firefox e Google Chrome, o mesmo deve funcionar corretamente sem nenhum problema grave que comprometa suas funções principais.

Cenário 7 – Interoperabilidade: Ao interagir com qualquer API RESTful de do sistema, ao retorno da mesma deve seguir a restrição *Hypermedia As the Engine Of Application State* (HATEOAS) facilitando e instruindo o cliente a consumir outras

funcionalidades atreladas ao contexto sem a necessidade de um conhecimento mais amplo da API.

6.3. Evidências da Avaliação

Esta seção apresenta as medidas registradas na coleta de dados e as justificativas, baseadas em evidências para comprovar o atendimento aos requisitos não-funcionais da aplicação Inside Nutri. Estes cenários foram apresentados na seção 6.2. Cenários.

Evidência da Avaliação do Cenário 1:

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	O sistema deve possuir um padrão de segurança elevado.
Preocupação:	O sistema deve impedir que recursos privados sejam acessados sem uma devida autenticação.
Cenário(s):	Cenário 1
Ambiente:	Sistema em operação normal
Estímulo:	O usuário tenta acessar uma página privada e/ou API, mas sem estar autenticado no sistema.
Mecanismo:	Criar um serviço para gerir identidades e acessos e prover autorização através de protocolo de autenticação interoperável.
Medida de resposta:	Negar acesso, retornar para a página inicial para novo cadastro ou autenticação.
Considerações sobre a arquitetura:	
Riscos:	A falta ou indisponibilidade do serviço de segurança pode prejudicar a usabilidade e confiabilidade do sistema.

Inside Nutri

	Comprometendo a validação de tokens de acesso e expondo funcionalidade e API privadas, além de informações sensíveis.
Pontos de Sensibilidade:	Serviço de autorização e autenticação, e pontos de extremidades com TLS (HTTPS) ativado.
Tradeoff:	Não há

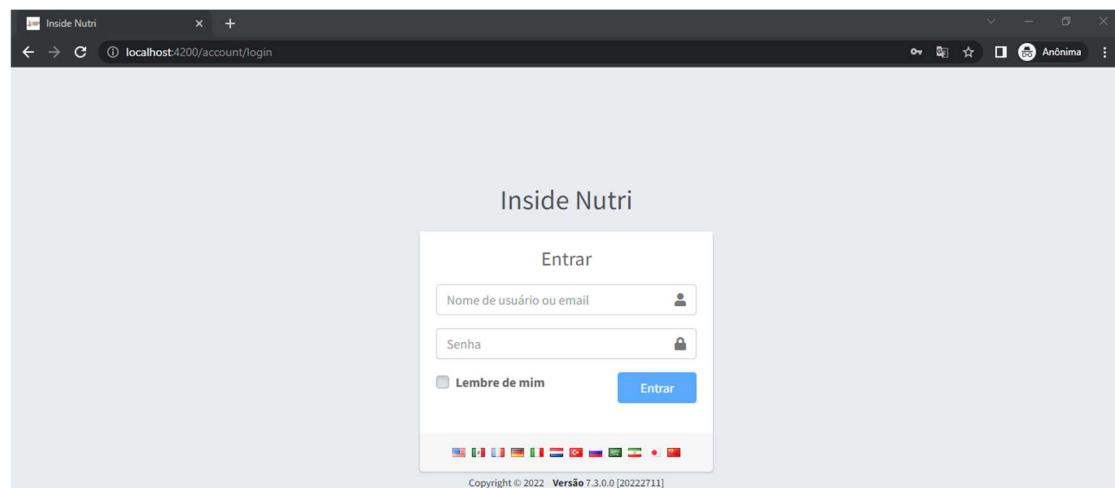


Figura 12 - Evidência 1 (Cenário 1) – Acesso a página de autenticação do sistema após clicar no botão Entrar.

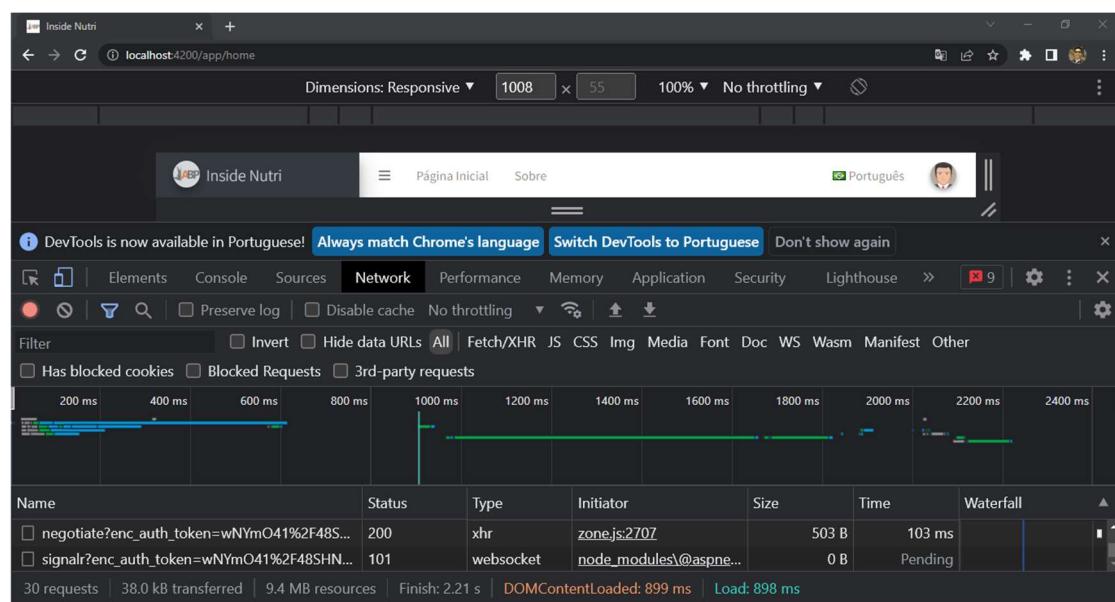


Figura 13 - Evidência 2 (Cenário 1) – Acesso a página inicial do sistema sem estar autenticado, evidenciando a exigência de autenticação inicial para acesso a funcionalidades privadas.

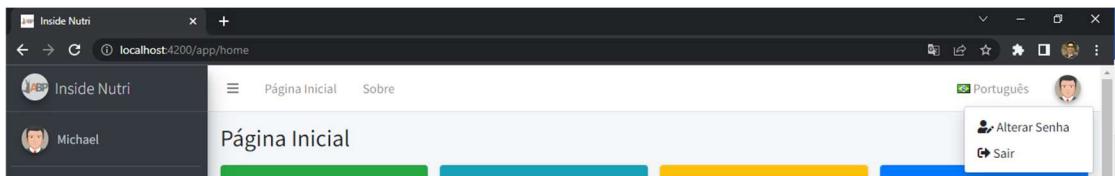


Figura 14 - Evidência 3 (Cenário 1) – Acesso a página inicial do sistema após autenticação. Evidenciando acesso a funcionalidade de sair da aplicação.

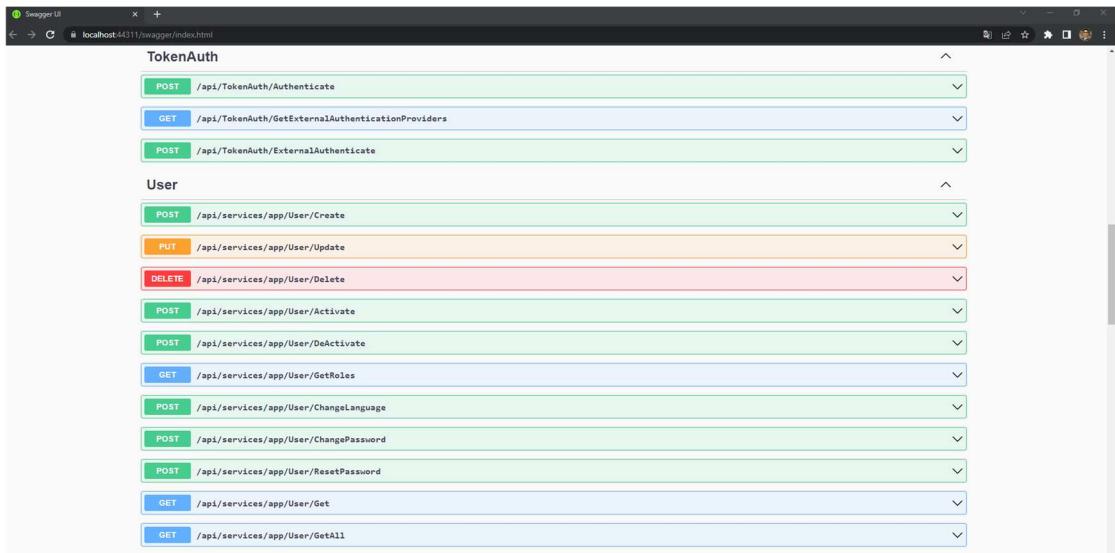


Figura 15 - Evidência 4 (Cenário 1) – Acesso a página do Swagger UI com as APIs de acesso a aplicação.

Evidência da Avaliação do Cenário 2:

Atributo de Qualidade:	Desempenho
Requisito de Qualidade:	O sistema deve possuir performance de processamento satisfatório.
Preocupação:	O sistema deve processar uma ação de forma rápida, desejável que o tempo médio seja menor ou igual a 3 segundos.
Cenário(s):	Cenário 2
Ambiente:	Sistema em operação normal
Estímulo:	O usuário registra um novo paciente ou novo usuário no sistema.
Mecanismo:	Criar um serviço RESTful para atender às requisições de cadastro de pacientes e

Inside Nutri

cadastro de usuário no sistema.	
Medida de resposta:	
Retornar dados que identifique o sucesso no cadastro do paciente e cadastro do usuário.	
Considerações sobre a arquitetura:	
Riscos:	Instabilidade relacionadas a rede podem impactar no tempo de processamento das requisições.
Pontos de Sensibilidade:	Alta disponibilidade através de <i>Replica-Set</i> e balanceamento de carga.
Tradeoff:	Não há

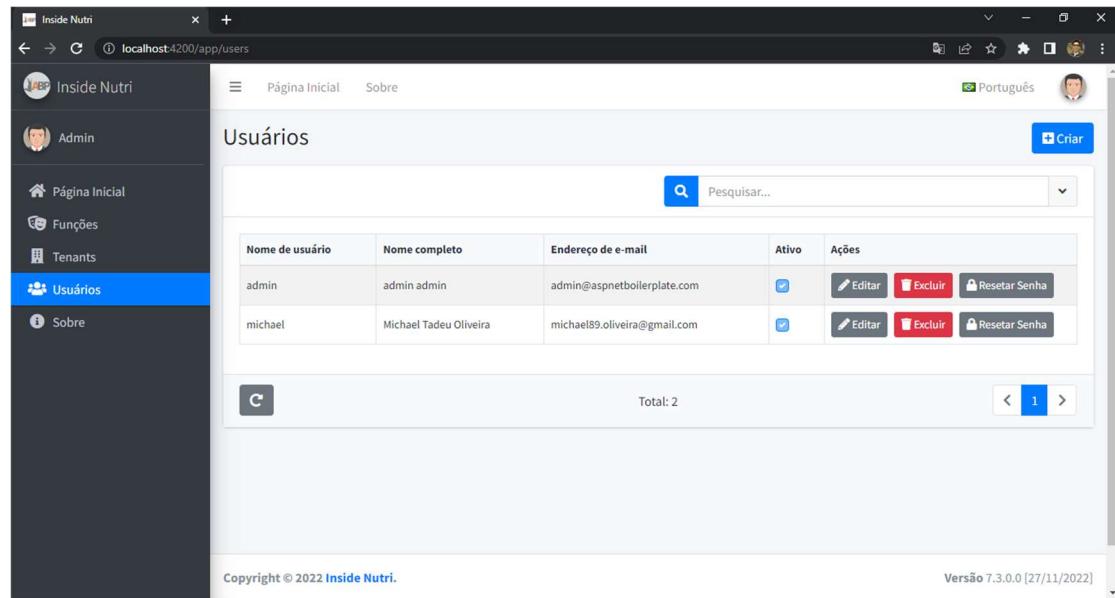


Figura 16 - Evidência 1 (Cenário 2) – Listagem de usuários cadastrados.

Projeto Integrado – Engenharia de Software - PMV

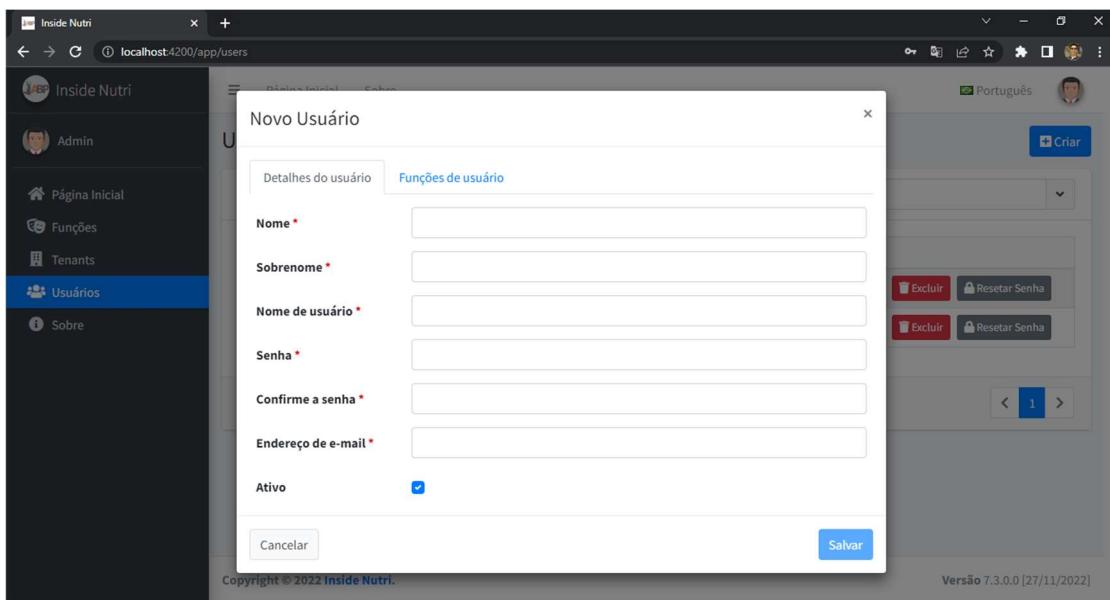


Figura 17 - Evidência 2 (Cenário 2) – Tela de cadastro de um novo usuário.

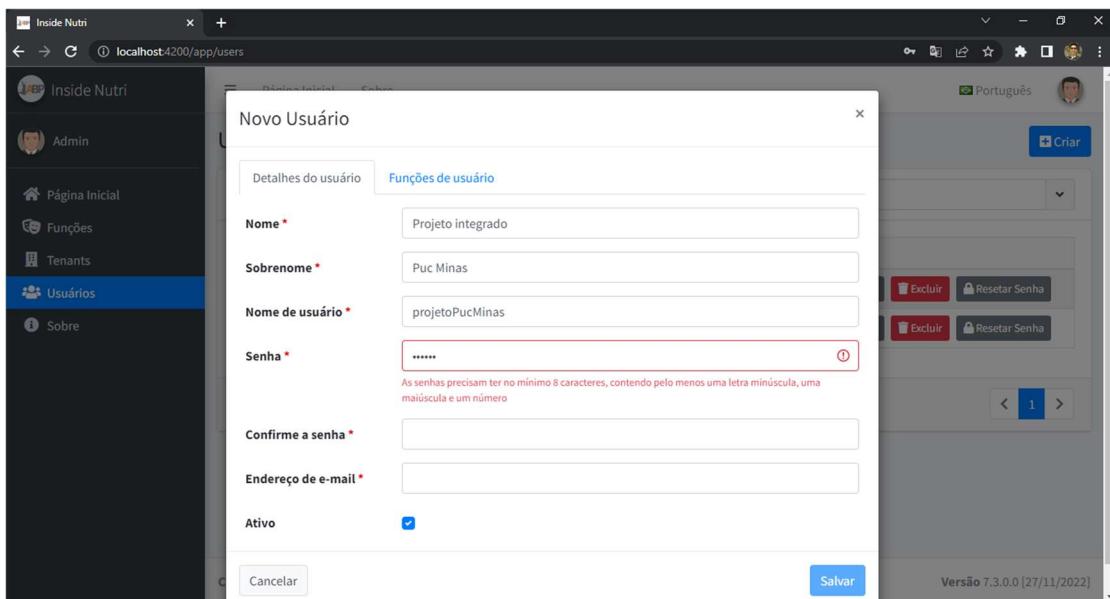


Figura 18 - Evidência 3 (Cenário 2) – Validações do campo senha.

Inside Nutri

The screenshot shows the 'Novo Usuário' (New User) creation form. The 'Senha' (Password) field contains '*****' and the 'Confirme a senha' (Confirm Password) field also contains '*****'. A red error message 'As senhas são diferentes' (The passwords are different) is displayed below the confirm password field.

Nome *	Projeto integrado
Sobrenome *	Puc Minas
Nome de usuário *	projetoPucMinas
Senha *	***** As senhas precisam ter no mínimo 8 caracteres, contendo pelo menos uma letra minúscula, uma maiúscula e um número
Confirme a senha *	***** As senhas são diferentes
Endereço de e-mail *	
Ativo	<input checked="" type="checkbox"/>

Funções de usuário

Salvar

Figura 19 - Evidência 4 (Cenário 2) – Validações do campo confirmar senha.

The screenshot shows the 'Novo Usuário' (New User) creation form with all fields filled. The 'Senha' (Password) field contains '*****' and the 'Confirme a senha' (Confirm Password) field also contains '*****'. The 'Endereço de e-mail' (Email Address) field contains 'projetoPucMinas@gmail.com'.

Nome *	Projeto integrado
Sobrenome *	Puc Minas
Nome de usuário *	projetoPucMinas
Senha *	*****
Confirme a senha *	*****
Endereço de e-mail *	projetoPucMinas@gmail.com
Ativo	<input checked="" type="checkbox"/>

Funções de usuário

Salvar

Figura 20 - Evidência 5 (Cenário 2) – Todos os campos preenchidos do cadastro de usuários.

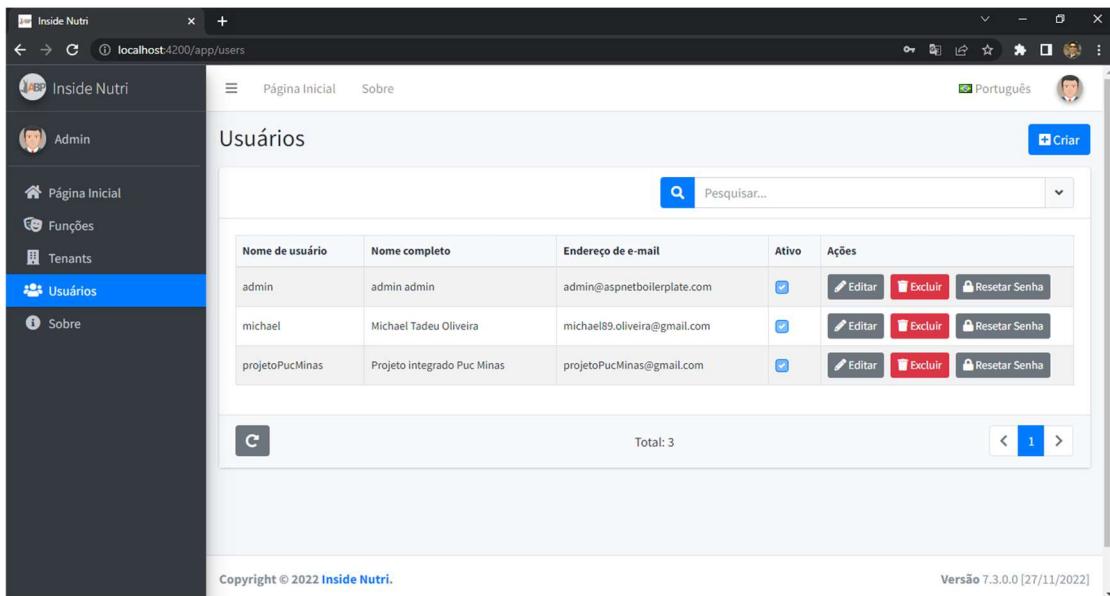


Figura 21 - Evidência 6 (Cenário 2) – Após clicar no botão Salvar, um novo usuário foi registrado com sucesso e em pouco menos que 800 milissegundos, evidenciando o bom desempenho e feedback visual do sucesso da operação através de mensagem específica.

Evidência da Avaliação do Cenário 3:

Atributo de Qualidade:	Padrão
Requisito de Qualidade:	O sistema deve possuir módulos independentes e autocontidos.
Preocupação:	O sistema deve ser composto por um conjunto de serviços coesos e com baixo acoplamento entre si.
Cenário(s):	Cenário 3
Ambiente:	Sistema em operação normal
Estímulo:	O mantenedor realizar a implantação dos módulos separados do sistema em <i>containers Docker</i> .
Mecanismo:	Fragmentar de forma organizada e coesa as funcionalidades do sistema e dividi-las em microserviços separados e com baixo acoplamento expondo suas funções, por exemplo, através de APIs RESTful.
Medida de resposta:	

Indisponibilizar apenas uma parte do sistema quando ocorrer necessidade de manutenção corretiva e/ou evolutiva de algum módulo.	
Considerações sobre a arquitetura:	
Riscos:	Apesar do baixo acoplamento entre os módulos, a indisponibilidade de algum deles pode gerar comportamentos inesperados durante processamento de fluxos complexos com várias integrações e comunicações.
Pontos de Sensibilidade:	Evolução/ Versionamento de API e suporte a API legada até migração completa por todos clientes.
Tradeoff:	Não há

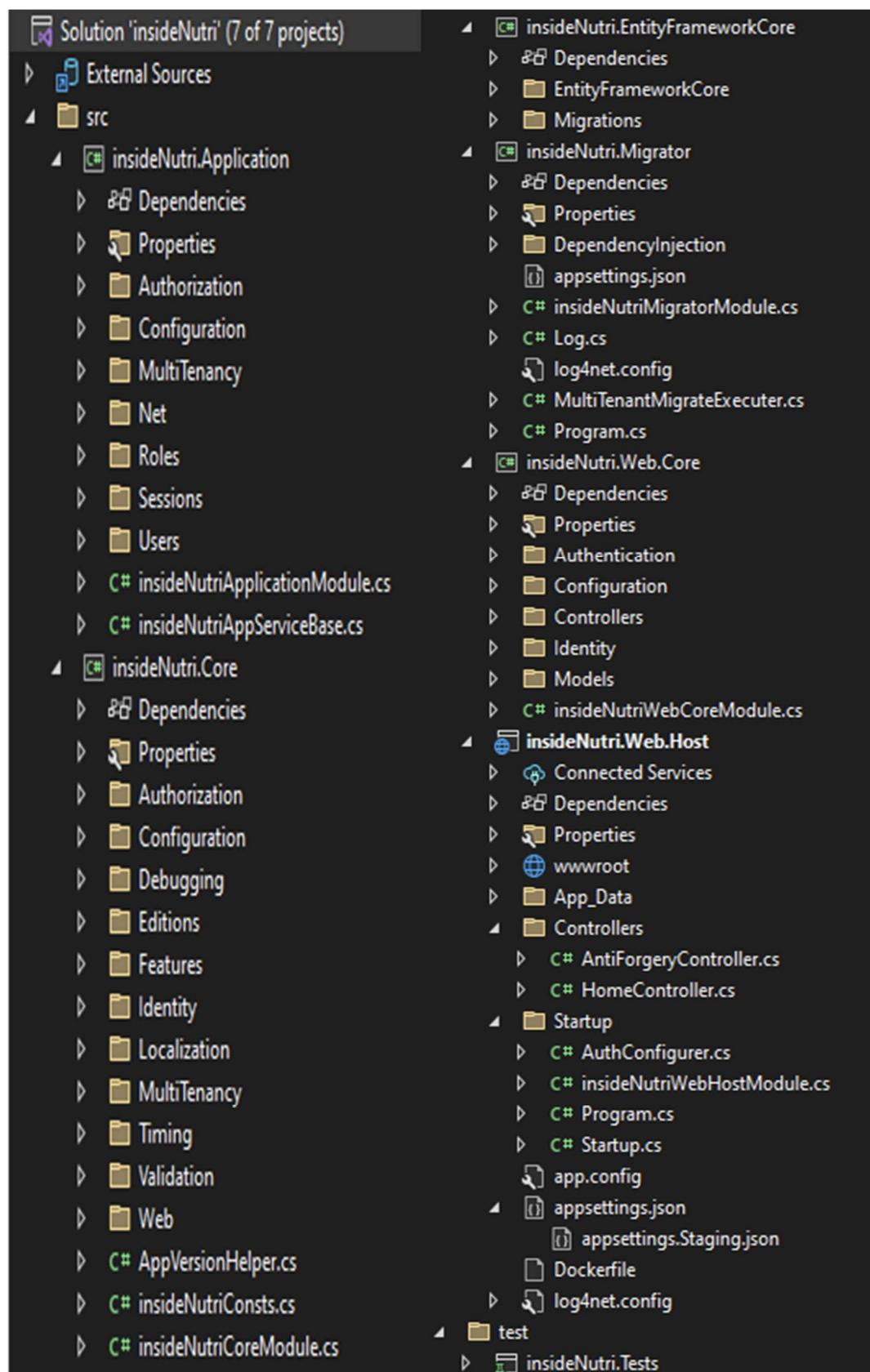


Figura 22 - Evidência 1 (Cenário 3) – Separação e implantação dos módulos específicos que compõem o sistema como um todo.

Evidência da Avaliação do Cenário 4:

Atributo de Qualidade:	Confiabilidade
Requisito de Qualidade:	O sistema deve possuir métricas de comunicações distribuídas.
Preocupação:	O sistema deve possuir um serviço que facilite a análise dos eventos de comunicação entre os serviços distribuídos.
Cenário(s):	Cenário 4
Ambiente:	Sistema em operação normal
Estímulo:	Os módulos do sistema se integram e enviam amostras de métricas das requisições para um serviço específico que centraliza as informações.
Mecanismo:	Criar um serviço para atender às requisições de eventos distribuídos relacionados a métricas de processamento, capturando e armazenando tais eventos.
Medida de resposta:	Disponibilizar de forma visual acesso aos dados através de telas de consulta.
Considerações sobre a arquitetura:	
Riscos:	A indisponibilidade do serviço pode dificultar na investigação de problemas de integração e latência que possam ocorrer entre os serviços distribuídos.
Pontos de Sensibilidade:	Fila de mensageria como ponto de resiliência.
Tradeoff:	Não há

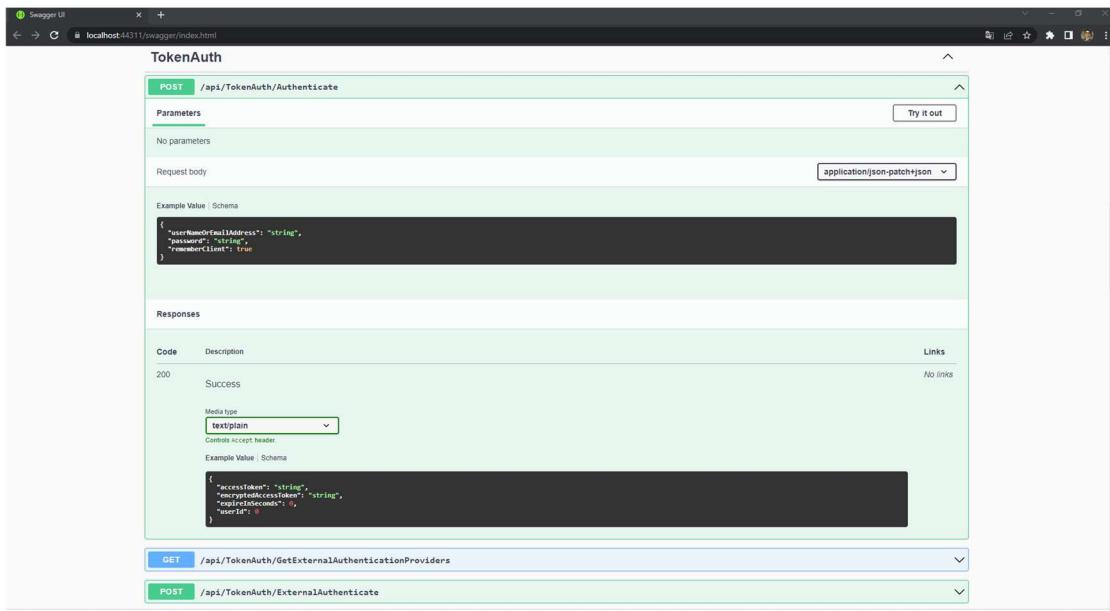


Figura 23 - Evidência 1 (Cenário 4) – Página que detalha um evento específico mostrando mais informações.

Evidência da Avaliação do Cenário 5:

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	O sistema deve se adequar ao ambiente <i>mobile</i> e <i>desktop</i> .
Preocupação:	O sistema deve possuir design adaptativo (responsivo) quando acessado via dispositivos móveis (Celular/Tablets) com resoluções menores e em dispositivos como (PC/Notebook) com resoluções maiores sem perda de funcionalidades.
Cenário(s):	Cenário 5
Ambiente:	Sistema em operação normal
Estímulo:	Usuário acessando o sistema e registrando um novo paciente.
Mecanismo:	Criar telas que tenham suporte ajustável e adaptativo de seus componentes visuais melhorando a disposição do conteúdo na tela.
Medida de resposta:	Ajustar de forma dinâmica os componentes visuais da tela em diversos dispositivos

Inside Nutri

mantendo todas as funcionalidades.	
Considerações sobre a arquitetura:	
Riscos:	Podem ocorrer imprevistos na melhor disposição e adaptação do conteúdo caso o dispositivo tenha tela extremamente pequena.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

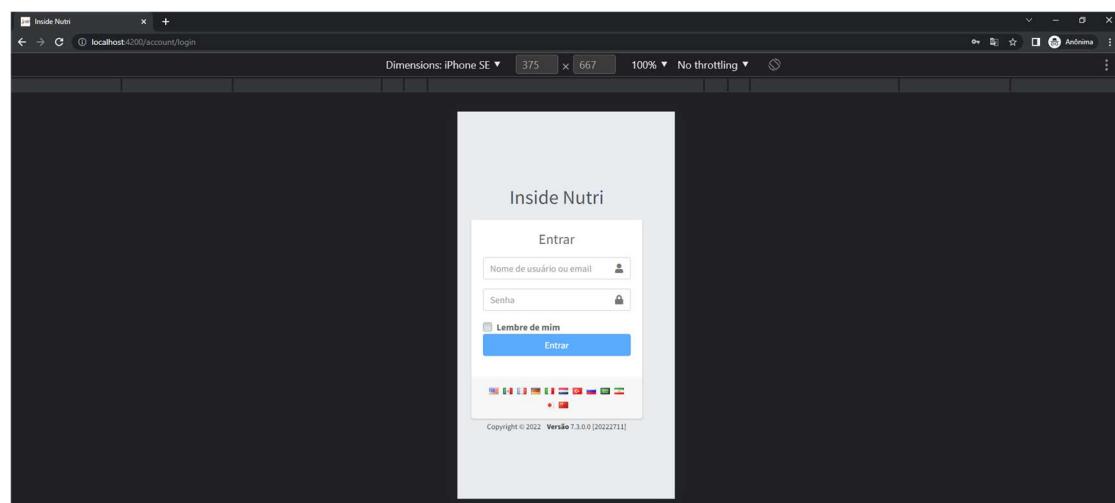


Figura 24 - Evidência 1 (Cenário 5) – Página inicial autenticada acessada via navegador web em ambiente Desktop com janela redimensionada para mostrar os componentes visuais com ajustes padrão.

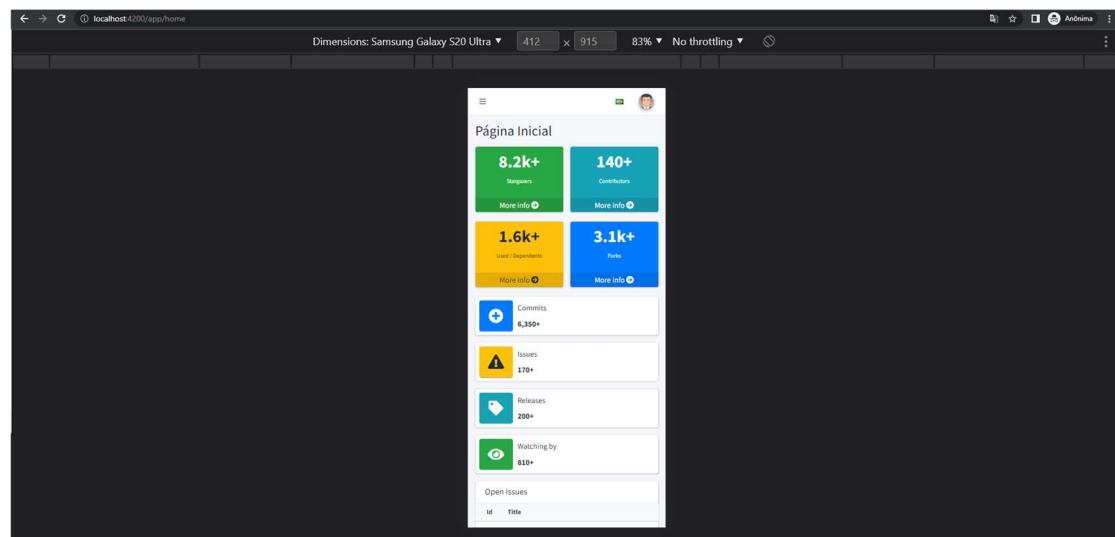


Figura 25 - Evidência 2 (Cenário 5) – Página inicial autenticada acessada via navegador web em ambiente Desktop com janela redimensionada para mostrar os componentes visuais com ajustes, se colapsando e se readequando.

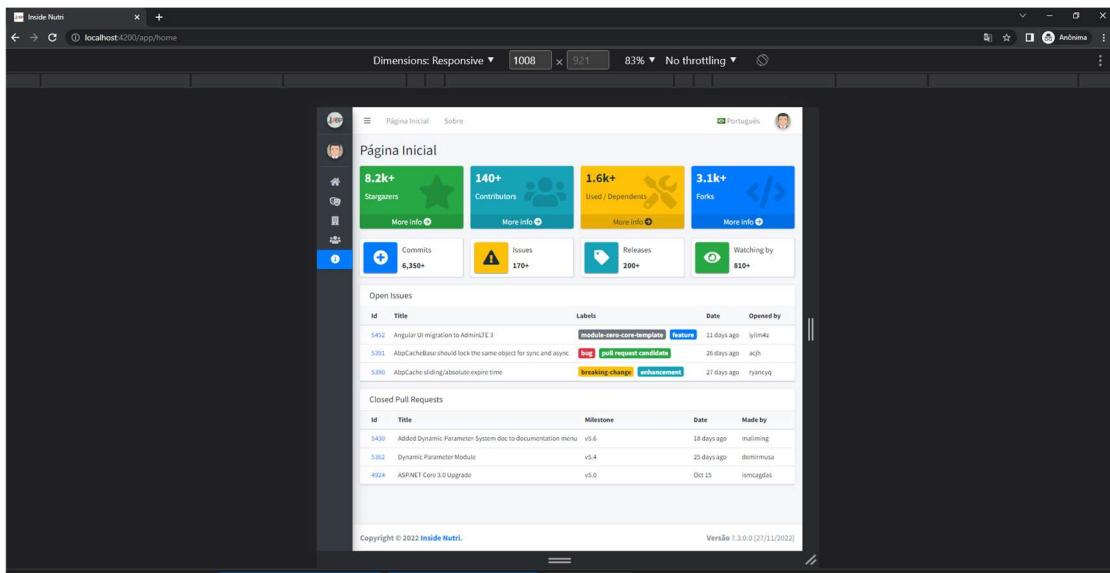


Figura 26 - Evidência 3 (Cenário 5) – Página inicial autenticada acessada via navegador web em ambiente Desktop com janela redimensionada para mostrar os componentes visuais com ajustes, se colapsando e se readequando.

Evidência da Avaliação do Cenário 6:

Atributo de Qualidade:	Compatibilidade
Requisito de Qualidade:	O sistema deve funcionar em qualquer navegador moderno.
Preocupação:	O sistema deve funcionar corretamente em diferentes navegadores web modernos sem comprometer suas funcionalidades.
Cenário(s):	Cenário 6
Ambiente:	Sistema em operação normal
Estímulo:	Usuário acessando o sistema e registrando uma ocorrência em diferentes navegadores web.
Mecanismo:	Criar portal web usando o framework Angular/TypeScript onde após transpilação para código nativo JavaScript/CSS o mesmo seja compatível com a maioria dos navegadores web modernos.
Medida de resposta:	Disponibilizar site com suas capacidades funcionais e visuais funcionando

Inside Nutri

corretamente independentemente do navegador web utilizado para acesso.	
Considerações sobre a arquitetura:	
Riscos:	Podem ocorrer imprevistos funcionais do site caso o mesmo seja acessado por navegadores muito antigos, legados e descontinuados. Por exemplo: Internet Explorer.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

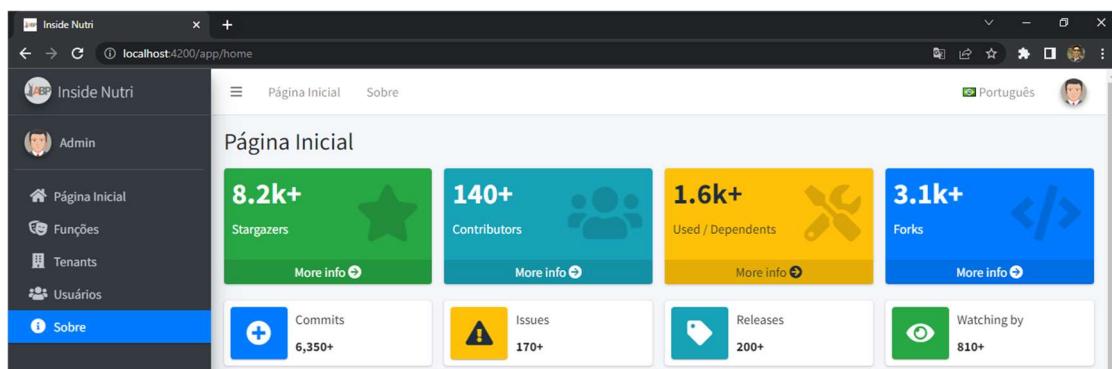


Figura 27 - Evidência 1 (Cenário 6) – Página Inicial acessado via navegador Google Chrome.

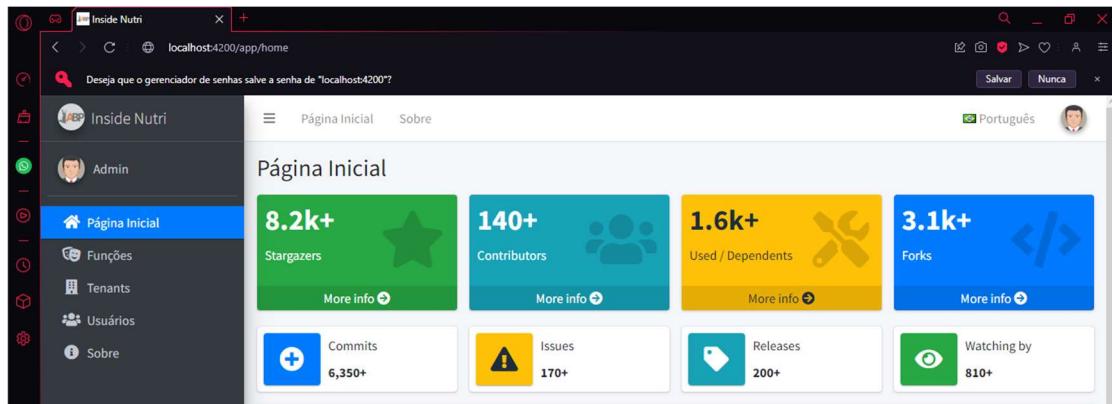


Figura 28 - Evidência 2 (Cenário 6) – Página Inicial acessado via navegador Opera GX.

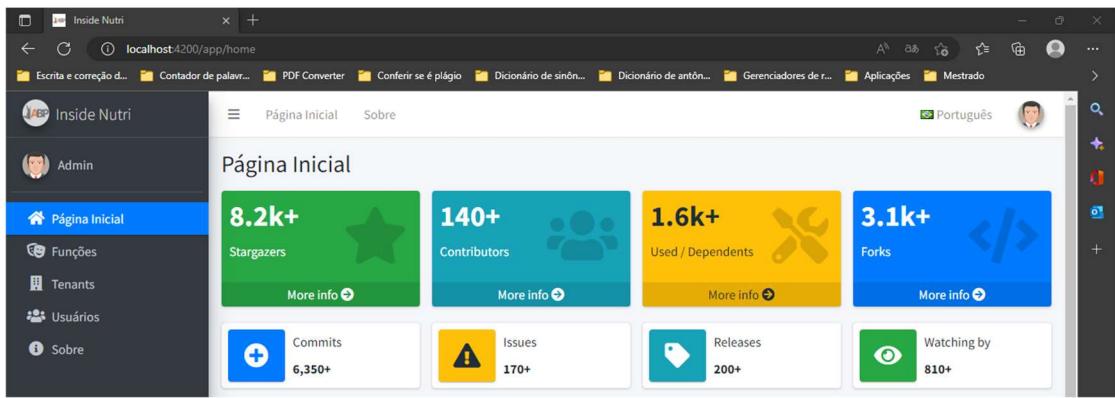


Figura 29 - Evidência 3 (Cenário 6) – Página Inicial acessado via navegador Microsoft Edge.

Evidência da Avaliação do Cenário 7:

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	O sistema deve se comunicar com outros sistemas.
Preocupação:	
O sistema deve utilizar padrão de comunicação RESTful seguindo restrição HATEOAS para que possa interagir com outros sistemas independentemente da tecnologia usada na codificação.	
Cenário(s):	
Cenário 7	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Cliente interage com a API enviando uma solicitação HTTP para listagem de ocorrências.	
Mecanismo:	
Criar um serviço RESTful para atender às requisições do cliente para o contexto de ocorrências.	
Medida de resposta:	
Retornar os dados requisitados no formato HAL+JSON.	
Considerações sobre a arquitetura:	
Riscos:	Podem ocorrer falha de comunicação ou lentidão de resposta caso exista alguma

	instabilidade de rede.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

The screenshot shows a Swagger UI interface for a 'User' endpoint. At the top, there is a green 'POST' button next to the URL '/api/services/app/User/Create'. Below the button, there are sections for 'Parameters' (which says 'No parameters') and 'Request body' (with a dropdown menu set to 'application/json-patch+json'). Under 'Request body', there is a code block containing a JSON schema for user creation:

```
{
  "username": "string",
  "name": "string",
  "surname": "string",
  "emailAddress": "user@example.com",
  "isActive": true,
  "roleNames": [
    "string"
  ],
  "password": "string"
}
```

Below the request body, there is a 'Responses' section. It lists a single response entry for 'Code: 200 Success'. The 'Links' column for this entry says 'No links'. Under the '200 Success' entry, there is a 'Media type' dropdown set to 'text/plain' and a note 'Controls Accept header.' Below this, there is another code block showing a sample response JSON:

```
{
  "id": 0,
  "username": "string",
  "name": "string",
  "surname": "string",
  "emailAddress": "user@example.com",
  "isActive": true,
  "fullName": "string",
  "lastLoginTime": "2022-12-15T19:12:13.572Z",
  "creationTime": "2022-12-15T19:12:13.572Z",
  "roleNames": [
    "string"
  ]
}
```

Figura 30 - Evidência 1 (Cenário 7) – Retorno da API de criação de usuário com dados estruturados seguindo restrição HATEOAS.

6.4. Resultados Obtidos

Esta seção é apresentada através de uma tabela, os resultados da arquitetura produzida para o sistema, onde os requisitos não-funcionais planejados inicialmente foram todos testados e homologados, considerando também os atributos propostos de qualidade verificados após a PoC.

Requisitos Não Funcionais	Teste	Homologação
RNF01: Desempenho — A operação de salvar uma ocorrência tem que ser rápida, não podendo exceder mais do que 3 segundos em média para ser executada.	OK	OK
RNF02: Usabilidade — A interface gráfica deve ser responsiva aplicando um design adaptativo para melhorar a disposição do conteúdo na tela do dispositivo melhorando a experiência do usuário.	OK	OK
RNF03: Compatibilidade — O sistema deverá funcionar corretamente nos navegadores web modernos mais usados (Google Chrome, Edge e Firefox).	OK	OK
RNF04: Segurança — O acesso a APIs privadas precisa ser seguro, exigindo a devida autenticação e autorização do usuário, caso contrário resultará respectivamente nos erros HTTP 401 - (<i>Unauthorized</i>) ou HTTP 403 (<i>Forbidden</i>).	OK	OK
RNF05: Padrão — O software deverá ser orientado a camadas possuindo baixo acoplamento e alta coesão seguindo o princípio de Responsabilidade Única para prover melhor manutenibilidade e evolução.	OK	OK
RNF06: Confiabilidade — Rastreamento distribuído para coleta e pesquisa de dados de tempo de requisições para detectar possíveis problemas de latência na arquitetura de serviços deverá ter um percentual de amostragem de 100%.	OK	OK

7. Avaliação Crítica dos Resultados

Esta seção é apresentada através de um quadro resumo, os principais pontos positivos e negativos da arquitetura proposta, esclarecendo também pontos de melhorias para entendimento das limitações arquiteturais, e os prós e contras das tecnologias utilizadas.

Ponto avaliado	Descrição
Padrão arquitetural	<p>A arquitetura de um sistema deve ser pensada para um propósito, onde não existe uma que seja perfeita o suficiente para todas as situações, ou na qual resolverá todos os desafios. Dito isso e, analisando os requisitos arquiteturais propostos e os requisitos não-funcionais idealizados incialmente para o Sistema de Gestão de Ocorrência, o padrão arquitetural orientado à microserviços foi o que melhor se adequou as necessidades, ciente e levando em consideração alguns pontos abaixo:</p> <p>Prós:</p> <ul style="list-style-type: none"> - Baixo acoplamento a outros serviços, permitindo que a equipe trabalhe independentemente utilizando a tecnologia que mais conhece e sem ser impactado na maior parte do tempo por mudanças em outros serviços; - Maior independência e organização de implantação e de manutenção corretiva/evolutiva pontual; - Eliminação de dependência tecnológica a longo prazo, tendo mais liberdade para desenvolvimento de serviços novos usando tecnologia mais atuais. <p>Contras:</p> <ul style="list-style-type: none"> - Manter a consistência dos dados pode ser mais complexa em uma arquitetura distribuída; - Maior complexidade na gestão e organização das comunicações entre os serviços através de versionamento de APIs; - Identificar e lidar com falhas parciais ou indisponibilidades totais de alguns serviços que pode atrapalhar um fluxo/processamento negocial.

Ponto avaliado	Descrição
API <i>Gateway</i>	Um ponto de melhoria na arquitetura atual seria o uso de um API <i>Gateway</i> . Levando em consideração a arquitetura distribuída de serviços que compõem o sistema como um todo, naturalmente existe uma granularidade das APIs com endereços e portas diferentes para cada API, o que pode dificultar o mapeamento, obtenção e consolidação de informações sobre determinado assunto. A implementação de um API <i>Gateway</i> seria interessante para resolver esse problema, criando um ponto centralizado e único, como um proxy reverso agregando informações para os clientes.
Soluções em nuvem	Nenhuma solução pronta em nuvem foi utilizada na arquitetura proposta digamos que por falta de conhecimento ou não querer ficar acoplado à um fornecedor e também porque não houve necessidade, como por exemplo para armazenamento de dados e serviço de gerenciamento de acesso e identidade, onde foram usadas soluções open-source, mais conhecidas e mais flexíveis, como MySQL e Keycloak respectivamente, mas o fato de não utilizar soluções prontas em nuvem não é necessariamente um ponto negativo, nesse caso sendo neutro, pois computação/soluções em nuvem também têm seus prós e contras.
Conteinerização dos serviços	Um ponto positivo e interessante na arquitetura proposta foi o uso da abordagem de empacotamento dos serviços como uma imagem (Docker) onde os serviços podem ser implantados como container, e de uma vez só facilmente através do Docker Compose. Uma das vantagens disso é que cada serviço é autocontido podendo ser composto de outros serviços exclusivamente necessários para um correto funcionamento operacional do mesmo, melhorando a disponibilidade e <i>throughput</i> , além da escalabilidade, manutenibilidade, etc. E uma pequena desvantagem talvez seria a falta de compatibilidade da imagem entre plataformas no lado servidor (Windows/Linux).

Ponto avaliado	Descrição
<i>Frontend</i>	A camada <i>fronteend</i> feita em Angular 14, foi criada totalmente desacoplada e independente do <i>backend</i> (<i>server-side web application</i>) onde toda comunicação com o servidor é feita através de APIs REST. Isso é interessante e positivo pois flexibilizou o uso de um framework web e mais popular que possuí sua própria organização e modularização das camadas através de componentes. Além disso, essa abordagem facilita mudanças caso necessário codificar um outro <i>frontend</i> usando outra tecnologia.
<i>Backend</i>	Os serviços de <i>backend</i> essencialmente nessa arquitetura foram planejados e implementados para expor suas funcionalidades através de APIs <i>RESTful</i> sendo um ponto positivo pois facilita a integração com clientes web ou mobile. O Spring framework foi muito utilizado pelo fato de ser bem conhecido, com funcionalidades prontas para produção e propiciar fácil desenvolvimento de microserviços, mas essa arquitetura proposta não impede que outras tecnologias sejam utilizadas.

8. Conclusão

Este trabalho apresentou um projeto arquitetural e implementação de uma prova de conceito da aplicação Inside Nutri para cálculos e cadastros de pacientes de dietas enterais. Através deste trabalho comprehende-se com mais clareza as etapas que envolvem a idealização, definição documental, implementação arquitetural e implantação de um software.

Todo o processo foi extremamente importante para o aprendizado, possibilitando desenvolver um senso crítico e ter opinião sólida sobre propostas de soluções arquiteturais. A oportunidade de experimentar e aprender novas tecnologias também foi de grande valia, ajudando a compreender melhor os desafios que envolvem o trabalho com frontend, backend e DevOps.

Avaliando o protótipo elaborado se faz necessário evidenciar possíveis melhorias arquiteturais, tais como a implementação de um API Gateway e a implantação do sistema distribuído em um cluster Kubernetes para um controle mais adequando dos serviços. Além disso, em relação às melhorias funcionais já listadas na seção de requisitos funcionais, vale destacar a implementação de novas funções de cadastro de fornecedores em outros estados, dicas nutricionais, envio de e-mail, criação de dietas entre outros.

Portanto, pelo apresentado conclui-se que os objetivos pretendidos com a elaboração deste trabalho foram alcançados e para que haja aplicabilidade em um contexto organizacional, aconselha-se alterações para adequação ao real contexto e conjuntura em que a companhia esteja inserida.

Referências

- DE OLIVEIRA ROSA, Thatiane et al. A method for architectural trade-off analysis based on patterns: Evaluating microservices structural attributes. In: Proceedings of the European Conference on Pattern Languages of Programs 2020. 2020. p. 1-8.
- OLIVEIRA, L.B.; JUNIOR, P.B.R.; GUIMARÃES, N.M.; DIDONET, M.T. Variáveis relacionadas ao tempo de internação e complicações no pós operatório de pacientes submetidos à cirurgia do trato gastrointestinal. Comunicação em Ciências da Saúde. Vol. 21. Num. 4. 2010. p. 319-330.
- MINISTÉRIO DA SAÚDE. Secretaria de Atenção à Saúde. Manual de Terapia Nutricional na Atenção Especializada Hospitalar. Brasília. Ministério da Saúde. 2016.
- LOPES, D.A.M.; MANZATTI, F.; BRUSTOLIN, A.; PENTEADO, E.G.; FRANCO, S.; BARATTO, I. Perfil nutricional de pacientes em um hospital de Guarapuava-PR. Anais da SIEPE. Semana de Integração Ensino, Pesquisa e Extensão. 26 a 30 de outubro de 2009.
- LEITE, H. P.; CARVALHO, W. B.; MENESSES, J. F. S. Atuação da equipe multidisciplinar na terapia nutricional de pacientes sob cuidados intensivos. Revista de Nutrição, 2005.
- GARCIA, R.W.D.; PADILHA, M.; SANCHES, M. Alimentação hospitalar: proposições para a qualificação do Serviço de Alimentação e Nutrição, avaliadas pela comunidade científica. Revista Ciência & Saúde Coletiva. Vol. 17. Num. 2. 2012. p. 473-480.
- CFN - CONSELHO FEDERAL DE NUTRICIONISTAS. Resolução CFN nº 600, de 25 de fevereiro de 2018. Dispõe sobre a definição das áreas de atuação do nutricionista e suas atribuições, indica parâmetros numéricos mínimos de referência, por área de atuação, para a efetividade dos serviços prestados à sociedade e dá outras providências. Diário Oficial da União, p. 1-55, 2018.
- WAITZBERG, Dan Linetzky; CAIAFFA, Waleska T.; CORREIA, M. I. T. D. Inquérito brasileiro de avaliação nutricional hospitalar (Ibranutri). Revista Brasileira de Nutrição Clínica, v. 14, p. 124-34, 1999.
- SOMMERVILLE, I. Engenharia de Software. 10ª. edição. Pearson Universidades, 768 p., 2019.
- CHITCHYAN, R.; RASHID, A.; SAWYER, P.; GARCIA, A.; ALARCON, M. P.; BAKKER, J.; TEKINERDOGAN, B.; CLARKE, S.; JACKSON, A. Report Synthesizing State-of-the-Art in Aspect-Oriented Requirements Engineering, Architectures and Design. AOSD-Europe Deliverable D, 11, 1-259. 2005.
- RIEHLE, Dirk; ZÜLLIGHOVEN, Heinz. Understanding and using patterns in software development. Tapos, v. 2, n. 1, p. 3-13, 1996.
- BROWN, Simon. The c4 model for visualising software architecture (2020). 2020.