UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

AIRTON DA CRUZ

APLICATIVO PARA ASSESSORIA ESPORTIVA E NUTRICIONAL

TRABALHO DE CONCLUSÃO DE CURSO 2

PATO BRANCO
2016

AIRTON DA CRUZ

APLICATIVO PARA ASSESSORIA ESPORTIVA E NUTRICIONAL

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2016

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Tecnologia em Análise e
Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

APLICATIVO PARA ASSESSORIA ESPORTIVA E NUTRICIONAL

por

AIRTON DA CRUZ

Este trabalho de conclusão de curso foi apresentado no dia 22 de novembro de 2016, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO.

Banca examinadora:

Profª. Drª. Beatriz Terezinha Borsoi
Orientador

Profª. Me. Rúbia Eliza de Oliveira Schultz Ascari

Profª. Me. Soelaine Rodrigues Ascari

Prof. Dr. Edilson Pontarolo
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Profª. Me. Soelaine Rodrigues Ascari
Responsável pela Atividade de Trabalho de Conclusão de Curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

CRUZ, Airton da. Aplicativo para assessoria esportiva e nutricional. 2016. 41f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

A prática de atividade física associada à nutrição é recorrente por diversos fatores. Entre esses fatores estão a otimização na perda de peso, o aumento de massa muscular e obtenção ou manutenção de saúde, com o ganho de resistência física e imunológica. A orientação de profissionais habilitados é bastante importante para que sejam obtidos os resultados esperados e sem prejuízo à saúde, seja por realização de atividade ou nutrição de forma inadequadas. O uso de aplicativos computacionais para o acompanhamento de atividade física e de orientações de nutrição passa a ter função relevante tanto para o profissional quanto para a pessoa que está sendo orientada. O profissional pode melhor acompanhar a evolução da pessoa e a pessoa pode mais facilmente seguir as orientações e instruções. Esse tipo de aplicativo para ambiente *web* faz com que o usuário possa mais facilmente acompanhar o que foi definido pelo profissional e os resultados que vem obtendo e os aspectos que precisa melhorar, por exemplo. Tendo em vista esse contexto, neste trabalho é apresentado o desenvolvimento de um sistema *web* que possibilite a interação entre aluno e profissional durante assessoria esportiva e nutricional. A principal tecnologia utilizada na implementação é o *framework* ASP Net MVC, que é complementado com tecnologias voltadas para o desenvolvimento de aplicações *web* denominadas como ricas, as *Rich Internet Application*.

Palavras-chave: Aplicação *web*. Aplicativo para assessoria esportiva. Aplicativo para acompanhamento nutricional.

ABSTRACT

CRUZ, Airton da. Application to sports and nutritional advice. 2016. 41f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

The practice of physical activity associated with nutrition, it recurrent by several factors. That include an optimization in weight loss, the increase of muscle mass and health care, through physical and immunological resistance gain. The orientation of qualified professionals is very important, so that the expected results will be obtained. In this way there will be no health risks by performing activities or nutrition conducted in inappropriate ways. The use of computer applications to monitor physical activity and guidelines for nutritional diets start having relevant function for the health professionals and for the person being oriented. The professional can monitor more effectively the evolution of the person, the person can easily follow the directions and instructions. This type of application is even easier to handle when its use for web environment. The user can monitor more practical way what was defined by the professional. Given this context, this work proposes a web system that enables interaction between student and professional during sports and nutritional consultancy. The main technology used in implementation is the ASP Net MVC framework which is complemented with technologies for developing web applications termed as rich, the Rich Internet Application.

Keywords: Web application. Application to sports advice. Application to nutritional advice.

LISTA DE FIGURAS

Figura 1 - Diagrama de casos de uso do sistema proposto 21
Figura 2 - Diagrama de classe 23
Figura 3 - Diagrama de entidades e relacionamentos do banco de dados 24
Figura 4 - Tela inicial do sistema..... 25
Figura 5 - Tela de cadastro de usuário..... 25
Figura 6 - Tela de login do usuário..... 26
Figura 7 - Tela inicial do usuário 26
Figura 8 - Tela de listagem de dietas..... 27
Figura 9 - Tela de cadastro de dietas..... 27
Figura 10 - Tela de montagem de dietas 28
Figura 11 - Tela de listagem de treinos 28
Figura 12 - Tela de cadastro de treinos..... 29
Figura 12 - Tela de montagem de treinos..... 29

LISTA DE QUADROS

Quadro 1 - Tecnologias e ferramentas utilizadas na modelagem e na implementação ... 17
Quadro 2 - Detalhamento do requisito funcional Cadastrar Aluno ... 18
Quadro 3 - Detalhamento do requisito funcional Cadastrar Profissional ... 19
Quadro 4 - Detalhamento do requisito funcional Cadastrar Dieta ... 19
Quadro 5 - Detalhamento do requisito funcional Cadastrar Treino ... 20
Quadro 6 - Detalhamento do requisito funcional Solicitar assessoria ... 20
Quadro 7 - Detalhamento do requisito funcional Enviar treino para aluno ... 21
Quadro 8 - Detalhamento do requisito funcional Enviar dieta para o aluno ... 21

LISTAGEM DE CÓDIGO

Listagem 1 – Código do leiaute da tela principal 30
Listagem 2 – Código da view "Index.cshtml 31
Listagem 3 – Código partial view MeuSuperioCadastro.cshtml 31
Listagem 4 – Classe Usuario.cs 32
Listagem 5 – Mapeamento em banco de dados da classe Usuário.cs 32
Listagem 6 – Classe Profissional.cs 33
Listagem 7 – Classe de mapeamento do banco de dados da classe Professional.cs 33
Listagem 8 – Classe "HomeController.cs 34
Listagem 9 – Classe ProfissionalController.cs 35
Listagem 10 – Código da tela de cadastro de profissional 37
Listagem 11 – Classe CadastroProfissional.cs 38

LISTA DE SIGLAS

AJAX	*Asynchronous JavaScript and XML*
CREF	Conselho Federal de Educação Física
CRN	Conselho Regional de Nutricionistas
CRUD	*Create, Retrieve, Update e Delete*
CSS	*Cascading Style Sheets*
GCD	Gasto Calórico Diário
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
IIS	*Internet Information Services*
MVC	*Model-View-Controller*
ORM	*Object-Relational Mapping*
RF	Requisitos Funcionais
RIA	*Rich Internet Applications*
RNF	Requisitos Não Funcionais
SQL	*Structured Query Language*
TMB	Taxa de Metabolismo Basal
W3C	*World Wide Web Consortium*
WHATW	*Web Hypertext Application Technology Working Group*
XHTML	*Extensible Hypertext Markup Language*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO **10**
1.1 CONSIDERAÇÕES INICIAIS 10
1.2 OBJETIVOS 11
1.2.1 Objetivo Geral 11
1.2.2 Objetivos Específicos 11
1.3 JUSTIFICATIVA 11
1.4 ESTRUTURA DO TRABALHO 12
2 RERENCIAL TEÓRICO **13**
2.1 APLICAÇÕES *WEB* 13
2.2 HTML5 14
3 MATERIAIS E MÉTODO **16**
3.1 MATERIAIS 16
3.2 MÉTODO 17
4 RESULTADO **18**
4.1 ESCOPO DO SISTEMA 18
4.2 MODELAGEM DO SISTEMA 18
4.4 IMPLEMENTAÇÃO DO SISTEMA 30
5 CONCLUSÃO **39**
REFERÊNCIAS **40**

1 INTRODUÇÃO

Este capítulo apresenta a introdução do trabalho que abrange as considerações iniciais, os objetivos e a justificativa. O capítulo é finalizado com a apresentação do texto por meio da descrição sumária dos capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

A prática de atividade física regular tem aumentado consideravelmente no Brasil nos últimos anos. Segundo pesquisa realizada pelo Ministério da Saúde (MINISTÉRIO..., 2013), 33,8% dos brasileiros com mais de 18 anos praticam atividade física regularmente. Isso equivale a um aumento de 12,6% nos últimos 5 anos.

A prática de atividade física associada à nutrição é recorrente por diversos fatores, que incluem a otimização na perda de peso, o aumento de massa muscular e melhoria de saúde, pelo ganho de resistência física e imunológica. Freire (2012) destaca que uma correta nutrição ajuda a evitar a fadiga física, otimiza o período de recuperação do organismo após a realização da atividade física, diminui o risco de lesões musculares e assegura a correta reposição dos estoques de energia.

O vínculo entre nutrição e atividade física ocorre em pessoas das mais diversas faixas etárias e finalidades: de pessoas com sobrepeso a atletas de alto desempenho. Assim, a procura por profissionais de educação física e de nutrição tem expandido consideravelmente levando à necessidade da otimização do atendimento.

A alta demanda por serviços exclusivos e personalizados na área de acompanhamento físico e nutricional, tem levado aos profissionais expandirem seu ramo de atuação para fora de academias e consultórios. Essa expansão é realizada por meios eletrônicos que possibilitem um maior número de atendimentos e uma abrangência regional.

De acordo com Oliveira (2010), os sistemas de informação trazem um avanço na qualidade dos serviços realizados e ofertados no atendimento personalizado de nutrição e atividade física. Essas tecnologias também ampliam a capacidade de tomada de decisão por permitirem um acompanhamento mais rigoroso do progresso nutricional e físico de cada pessoa e de fornecer mais dados ao profissional para que ele possa fundamentar uma decisão em termos de recomendações nutricionais e de atividades físicas.

Visando fornecer recursos computacionais, em termos de sistemas, para auxiliar a esses profissionais no acompanhamento e melhor atendimento, este trabalho visa desenvolver uma aplicação *web* que permita fornecer as funcionalidades necessárias para controle de assessorias esportiva e nutricional. E, assim, intermediar o contato entre os envolvidos que são as pessoas que realizam atividades físicas e acompanhamento nutricional e os profissionais dessas áreas.

1.2 OBJETIVOS

A seguir estão o objetivo geral e os objetivos específicos definidos para este trabalho.

1.2.1 Objetivo Geral

Implementar um sistema *web* que possibilite a interação entre aluno e profissional durante uma assessoria esportiva e nutricional.

1.2.2 Objetivos Específicos

- Possibilitar o controle de alunos e atendimentos por parte dos profissionais que prestam atendimento nutricional e na realização de atividades físicas.
- Auxiliar profissionais e alunos no acompanhamento e controle de atividades físicas e recomendações de dietas nutricionais.
- Fornecer uma ferramenta para auxílio na definição de dietas e de treinos físicos personalizados.

1.3 JUSTIFICATIVA

A implementação de uma aplicação para controle e auxílio em assessoria esportiva e nutricional, baseia-se na necessidade de profissionais liberais em fornecer atendimento de forma ágil, controlada e que possa transpor distâncias físicas. Assim, o profissional não necessita estar próximo de quem está sendo acompanhado (o aluno).

É importante que os profissionais desta área possuam uma ferramenta para auxiliar o atendimento, evitar o retrabalho e controlar suas informações e de seus alunos a fim de agilizar e otimizar os processos.

De acordo com Laudon e Laudon (2007), um sistema de informação é uma ferramenta de extrema importância dentro das organizações, uma vez que é composto por diversos fragmentos relacionados entre si, abstraindo dados, processando e fornecendo informações para a gestão. Um profissional liberal também é visto no contexto de uma empresa porque presta serviços e realiza atividades que necessitam de gestão. Profissionais de nutrição e de atividades físicas mesmo que não atuem vinculados a empresas também necessitam de auxílio de sistemas informatizados para a gestão das suas atividades.

O aplicativo desenvolvido como resultado deste trabalho se propõe a ser uma ferramenta de gestão para o treino e especificações nutricionais individuais. Assim, visando atender a essa necessidade foi desenvolvida uma aplicação *web* para controle de assessorias esportiva e nutricional. Esse aplicativo auxiliará tanto os profissionais quanto os alunos (pessoas que praticam atividade física e seguem prescrições nutricionais) desses profissionais.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos. Este é o primeiro e apresenta as considerações iniciais com o contexto do sistema desenvolvido, os seus objetivos e a justificativa. O Capítulo 2 apresenta o referencial teórico centrado em aplicações *web*. No Capítulo 3 estão as ferramentas e as tecnologias utilizada para a modelagem e a implementação do sistema. No Capítulo 4 é apresentado o resultado da realização do trabalho, ou seja, a modelagem e a implementação do sistema. Por fim estão as considerações finais seguidas pelas referências utilizadas no texto.

2 RERENCIAL TEÓRICO

Este capítulo apresenta o referencial do trabalho que está centrado em aplicações *web* que são caracterizadas como de interface rica, embora outros aspectos como o compartilhamento do processamento tradicionalmente realizado pelo servidor com o cliente, que reduz o tráfego de rede, também sejam relevantes nessa categoria de aplicações *web*.

2.1 APLICAÇÕES *WEB*

Nos últimos anos as aplicações *web* moveram-se das tradicionais (*web* 1.0) - nas quais a maior parte do processamento das informações é realizado no servidor enquanto o cliente tem a responsabilidade de apresentar conteúdo estático – para as aplicações denominadas *Rich Internet Applications* (RIA) (BERNARDI; DI LUCCA; DISTANTE, 2014). Tramontana, Amalfitano e Fasolino (2013) colocam 2005 como marco de evolução das aplicações *web* para as RIAs. Esses autores ressaltam que nas RIAs uma parte relevante da lógica de negócio é mantida no lado cliente, tomando vantagens das potencialidades do *Asynchronous JavaScript and XML* (AJAX).

AJAX é uma combinação de tecnologias *web* relacionadas que são usadas no lado cliente para criar aplicações *web* assíncronas (PAVLIĆ; PAVLIĆ; JOVANOVIĆ, 2012). Com a ajuda de AJAX, aplicações *web* podem enviar e receber dados do servidor de maneira assíncrona e sem interferir na apresentação da página *web*. De acordo com esses autores, AJAX é frequentemente utilizado em combinação com as tecnologias *Cascading Style Sheets* (CSS), *HyperText Markup Language* (HTML) e *eXtensible Markup Language* (XML), embora esse uso combinado não seja obrigatório.

De acordo com Bernardi, di Lucca e Distante (2014), as RIAs são caracterizadas por uma interação mais ampla e melhor com os usuários que não possuem um papel passivo (isto é, de ser somente consumidor de conteúdo armazenado e processando no lado servidor da aplicação), mas tem assumido um papel ativo que contribui, pela seleção que faz, da definição ou processamento de conteúdo a ser provido no lado cliente.

As RIAs, para mostrar usabilidade melhor, permitem melhor desempenho porque elas reduzem significativamente o número de requisições ao servidor pelo processamento no cliente de muitas operações e a manipulação de dados requeridos pelo usuário, reduzindo a

atualização da página e o tráfego de dados na Internet, que ocorre entre cliente e servidor (BERNARDI; DI LUCCA; DISTANTE, 2014).

As RIAs são baseadas em um novo estilo de arquitetura cliente-servidor que usa requisições assíncronas compostas de pequenos blocos de dados (BERNARDI; DI LUCCA; DISTANTE, 2014). Assim, as RIAs são mais similares às aplicações *desktop* porque elas combinam a arquitetura de distribuição da *web* caracterizada pela redução de tráfego com os recursos de interação (interface) e capacidade de processamento das aplicações *desktop* (DI LUCCA et al., 2002). Para esses autores, a combinação resultante é a melhoria de todos os elementos (dados, lógica de negócio, comunicação e apresentação) envolvidos em uma aplicação *web*.

Peintner, Kosch e Heuer (2009) ressaltam a similaridade de características das RIAs com as aplicações *desktop* quando se referem a elas como aplicações *web* que tendem a ter características e funcionalidades semelhantes às aplicações *desktop* tradicionais. Para esses autores, aplicações *web* tradicionais tendem a ser centradas no servidor, o que significa que a maioria do trabalho e do processamento é realizado nos servidores e os clientes somente apresentam conteúdo estático na forma de *Extensible Hypertext Markup Language* (XHTML). Para reduzir a carga do servidor, evitar interações lentas e acelerar o *feedback* do usuário, as aplicações RIA atuais movem processamento do servidor para os clientes.

Sumariamente, uma RIA pode ser vista como uma aplicação *web* na qual a interface com o usuário é processada no lado cliente e a lógica de negócio é definida por um serviço de *backend* (HASSAN; HOLT, 2001).

Como AJAX, HTML é um tipo de tecnologia cliente não *plugin* (LI-LI; ZHENG-LONG, 2012). Essas tecnologias que fornecem as características das RIAs, podem não somente melhorar a usabilidade das aplicações, mas, também, reduzir dados extra transmitidos em cada requisição e armazenar dados no cliente, reduzindo o tempo de conexão entre cliente e servidor. Isso significa que a frequência de interação em HTML5 é menor que AJAX (LI-LI; ZHENG-LONG, 2012).

2.2 HTML5

HTML5 é um padrão proposto pela *Web Hypertext Application Technology Working Group* (WHATW) e aceito pelo *World Wide Web Consortium* (W3C) após 10 anos do HTML4 (QI, 2011 *apud* LI-LI; ZHENG-LONG, 2012). HTML5 não somente provê novos

elementos semânticos como <header>, <nav>, <article> e <footer>, mas também simplifica e padroniza configurações de interface permitindo inserir áudio e vídeo por meio de *tags* <audio> e <video>, respectivamente, sem a necessidade de *plugins* de terceiros.

Apesar da padronização estabelecida para HTML5 e da popularidade das RIAs, para Fraternali, Rossi e Sánchez-Figueroa (2010), HTML5 assumiu um papel central no desenvolvimento das RIAs. A comunidade de pesquisadores deve conduzir uma revisão focada em esforços para assegurar que um conjunto complexo de características incorporadas nessa linguagem tenha coerência interna e atendam os requisitos dos desenvolvedores de RIAs.

Sharma, Kumar e Trivedi (2013) ressaltam que HTML5 é o desenvolvimento de *tags* de mídias ricas como áudio e vídeo e que são programáveis por meio de JavaScript. Para esses autores essas *tags* são de uso muito simples. CSS3 trabalha com HTML5 para especificar como os elementos da página devem ser atualizados. Uma especificação CSS define para o navegador *web* como apresentar os elementos.

O advento de HTML5 e AJAX tornou o navegador *web* um ambiente de desenvolvimento de aplicações sofisticado (CHOLIA; SKINNER; BOVERHOF, 2011). HTML5 é uma revisão dos padrões de HTML que adiciona recursos do lado do cliente no ambiente do navegador *web* (HTML5 E CSS3, 2015).

3 MATERIAIS E MÉTODO

A seguir estão os materiais e o método utilizados para a modelagem e a implementação do sistema obtido como resultado da realização deste trabalho.

3.1 MATERIAIS

O Quadro 1 apresenta as tecnologias e as ferramentas utilizadas para o desenvolvimento do trabalho.

Ferramentas	Versão	Disponível em	Aplicação no projeto
StarUML		http://sourceforge.net/projects/staruml/	Modelagem do sistema
ASP Net MVC	5	http://www.asp.net/mvc/mvc5	*Framework* para criação de aplicações *web* no padrão *Model-View-Controller* (MVC), com a possibilidade da utilização de Helpers html, para diversas linguagens de backend como C#, F#, Visual Basic e C.
JavaScript		https://www.javascript.com/	Linguagem orientada a objetos para aplicações *web*.
HTML5		http://www.w3.org/TR/html5/	Linguagem de marcação de textos para páginas *web*.
Bootstrap	3	http://getbootstrap.com/	Framework de estilizações de páginas por meio de CSS.
C#		https://msdn.microsoft.com/pt-br/library/kx37x362.aspx	Linguagem de programação de alto nível orientada a objetos pertencente à plataforma .NET.
Nhibernate	4	http://nhibernate.info/	NHibernate é uma das soluções de Mapeamento Objeto-Relacional, do inglês *Object-Relational Mapping* (ORM) para a plataforma de desenvolvimento Microsoft .NET, um *framework* que fornece o mapeamento do modelo relacional para a orientação a objeto e persistência de dados.
Fluent Nhibernate	4.0.3	http://www.fluentnhibernate.org/	*Framework* de mapeamento de objeto relacional que tem por finalidade facilitar o mapeamento de classes no banco de dados.
Jquery	1.10.2	https://jquery.com/	Biblioteca JavaScript.
*Internet Information Services (*IIS)	7.0	https://www.iis.net/	Servidor *web* para aplicações ASP.Net.
.Net framework	4.0	http://www.microsoft.com/net	Plataforma de execução de aplicações .NET.
Visual Studio Express for *web*	2013	https://www.visualstudio.com/en-us/products/visual-studio-express-	*Integrated Development Environment* (IDE) para

		vs.aspx	desenvolvimento de aplicações .NET.
ASP.NET Identity	2.0	https://www.asp.net/identity	Biblioteca para manipulação e controle de usuários e níveis de acesso.

Quadro 1 - Tecnologias e ferramentas utilizadas na modelagem e na implementação

3.2 MÉTODO

Neste trabalho de conclusão de curso o enfoque foi na análise, desenvolvimento e estudo das tecnologias com a finalidade de desenvolver um sistema *web* que atendesse aos objetivos estipulados. Inicialmente foram implementados os cadastros básicos e posteriormente as funcionalidades específicas e mais complexas. Para a realização do trabalho foram definidas as seguintes etapas:

a) **Levantamento dos requisitos**

   Os requisitos foram definidos com base na necessidade de profissionais das referidas áreas em assessorar seus alunos utilizando a Internet como meio de comunicação. Neste trabalho de conclusão de curso esses requisitos foram avaliados e mensurados por meio de diagramas.

b) **Análise e projeto**

   A modelagem foi desenvolvida com o intuito de facilitar a compreensão dos requisitos do sistema.

c) **Implementação**

   A implementação foi realizada utilizando as tecnologias apresentadas na Seção 3.1.

4 RESULTADO

Este capítulo apresenta o resultado da realização deste trabalho. O resultado está centrado na modelagem e na implementação do sistema.

4.1 ESCOPO DO SISTEMA

O sistema desenvolvido tem como finalidade controlar o atendimento de assessoria esportiva e nutricional, o qual deverá possibilitar o cadastro de profissionais e de alunos e permitir a interação entre os mesmos. Os usuários poderão controlar suas dietas e treinamentos como desejarem ou solicitar um atendimento especializado de um profissional que esteja cadastrado. Os profissionais poderão controlar a evolução dos seus alunos, bem como realizar o atendimento individual dos mesmos.

4.2 MODELAGEM DO SISTEMA

A seguir são apresentados os requisitos e os diagramas desenvolvidos a fim de apresentar os processos e a estrutura da aplicação. Os requisitos estão dispostos como funcionais e não funcionais fazendo uso do identificador RF para os funcionais e RNF para os não funcionais.

O Quadro 2 apresenta o requisito funcional cadastrar aluno e os requisitos não funcionais relacionados.

F1 Cadastrar aluno		Oculto ( )		
Descrição: O sistema deverá oferecer a possibilidade ao usuário de realizar seu cadastro bem como alterar suas informações cadastrais.				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 1.1 Enviar e-mail de confirmação	O sistema deverá enviar *e-mail* com *link* para confirmação do cadastro.	Usabilidade	( )	(x)
NF 1.2 Registro de ficha clinica	O sistema deverá manter o cadastro de uma ficha clínica que conterá as informações de atendimentos anteriores, e evolução do aluno.	Usabilidade	(x)	(x)

Quadro 2 - Detalhamento do requisito funcional Cadastrar Aluno

O Quadro 3 apresenta o requisito funcional cadastrar profissional e os requisitos não funcionais relacionados.

F2 Cadastrar profissional	Oculto ( )			
Descrição: O sistema deverá oferecer a possibilidade ao profissional de realizar seu cadastro bem como alterar suas informações cadastrais.				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 2.1 Enviar e-mail de confirmação	O sistema deverá enviar um *e-mail* com um *link* para confirmação do cadastro	Usabilidade	( )	(x)
NF 2.2 Manter negada a autorização de contratação de alunos	O sistema deverá manter negada a autorização de contratação de alunos durante o período de confirmação dos documentos pessoais e número do registro no Conselho Federal de Educação Física (CREF)/ Conselho Regional de Nutricionista (CRN).	Usabilidade	(x)	(x)

Quadro 3 - Detalhamento do requisito funcional Cadastrar Profissional

O Quadro 4 apresenta o requisito funcional cadastrar dieta e os requisitos não funcionais relacionados.

F3 Cadastrar dieta	Oculto ( )			
Descrição: O sistema deverá oferecer a possibilidade ao usuário de cadastrar e alterar suas dietas.				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 3.1 Permitir o usuário adicionar alimentos.	O sistema deverá oferecer uma tela de consulta de alimentos com suas respectivas informações nutricionais para que o usuário escolha o que melhor se adapta ao seu objetivo	Usabilidade	( )	(x)
NF 3.2 Permitir o usuário adicionar novas refeições	O sistema possibilitará o usuário adicionar novas refeições e horários.	Usabilidade	(x)	(x)
NF 3.3 Calcular macro nutrientes	O sistema calculará instantaneamente o número de proteínas, carboidratos e gorduras em cada refeição e o total do dia.	Usabilidade	(x)	(x)
NF 3.4 Impressão da dieta	O sistema permitirá a impressão da dieta.	Usabilidade	(x)	(x)

Quadro 4 - Detalhamento do requisito funcional Cadastrar Dieta

O Quadro 5 apresenta o requisito funcional cadastrar treino e os requisitos não funcionais relacionados.

F4 Cadastrar treino		Oculto ( )		
Descrição: O sistema deverá oferecer a possibilidade ao usuário aluno cadastrar e alterar seus treinamentos.				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 4.1 Permitir o usuário adicionar exercícios.	O sistema deverá oferecer uma tela de consulta de exercícios com suas respectivas informações para que o usuário escolha o que melhor se adapta ao seu objetivo.	Usabilidade	( )	(x)
NF 4.2 Impressão do treino	O sistema permitirá a impressão do treino cadastrado.	Usabilidade	(x)	(x)

Quadro 5 - Detalhamento do requisito funcional Cadastrar Treino

O Quadro 6 apresenta o requisito funcional solicitar assessoria e os requisitos não funcionais relacionados.

F5 Solicitar assessoria		Oculto ( )		
Descrição: O sistema deverá oferecer a possibilidade ao usuário de buscar um profissional e solicitar assessoria.				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 5.1 Enviar e-mail ao profissional selecionado.	O sistema deve enviar um *e-mail* para o profissional selecionado com as informações do solicitante avisando sobre a solicitação.	Usabilidade	(x)	(x)
NF 5.2 Liberação da assessoria	Após a aprovação por parte do profissional o sistema deverá exibir a ficha clínica do aluno.	Interface	(x)	(x)
NF 5.3 Validar duração da assessoria.	Ao aprovar a assessoria o profissional deverá informar a data de início e término do período de assessoria.	Usabilidade	(x)	(x)

Quadro 6 - Detalhamento do requisito funcional Solicitar assessoria

O Quadro 7 apresenta o requisito funcional enviar treino para o aluno e os requisitos não funcionais relacionados.

F6 Enviar treino para o aluno		Oculto ( )		
Descrição: O sistema deverá oferecer a possibilidade ao profissional de enviar um treino para o aluno.				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 6.1 Permitir o profissional alterar o treino.	O sistema deverá permitir que o profissional realize modificações especificas para	Usabilidade	(x)	(x)

	o aluno antes do envio do treino.			

Quadro 7 - Detalhamento do requisito funcional Enviar treino para aluno

O Quadro 8 apresenta o requisito funcional enviar dieta para o aluno e os requisitos não funcionais relacionados.

F7 Enviar dieta para o aluno		Oculto ( )		
Descrição: O sistema deverá oferecer a possibilidade ao profissional de enviar a especificação de um treino para o aluno.				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 7.1 Permitir o profissional alterar a dieta.	O sistema deverá permitir que o profissional realize modificações especificas para o aluno antes do envio da dieta.	Usabilidade	(x)	(x)

Quadro 8 - Detalhamento do requisito funcional Enviar dieta para o aluno

Na Figura 1 está representado o diagrama de casos de uso do sistema. Nela, os casos de uso são relacionados com os atores: aluno e profissional.

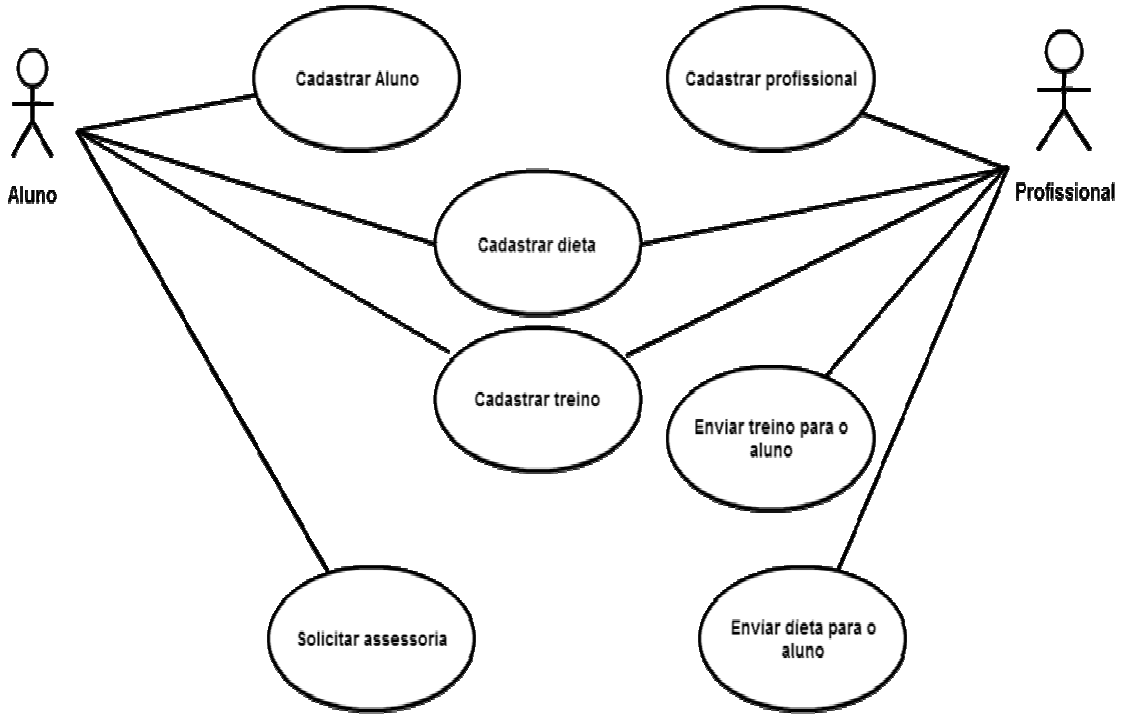

Figura 1 - Diagrama de casos de uso do sistema proposto

Na Figura 2 está representado o diagrama de classes do sistema proposto. Nesse diagrama está representada a estrutura do software por meio de suas classes. Essas classes foram definidas de maneira a tornar o desenvolvimento mais simples e flexível, pela

reutilização de código e evitando a criação de tabelas desnecessárias na base de dados. A flexibilidade do modelo é evidente pela chave primária da classe 'ApplicationUser' que é uma *string* 'GUID' única, fornecendo, assim, a possibilidade de migração e interação entre outras bases de dados sem prejudicar a integridade dos dados por haver um identificador único.

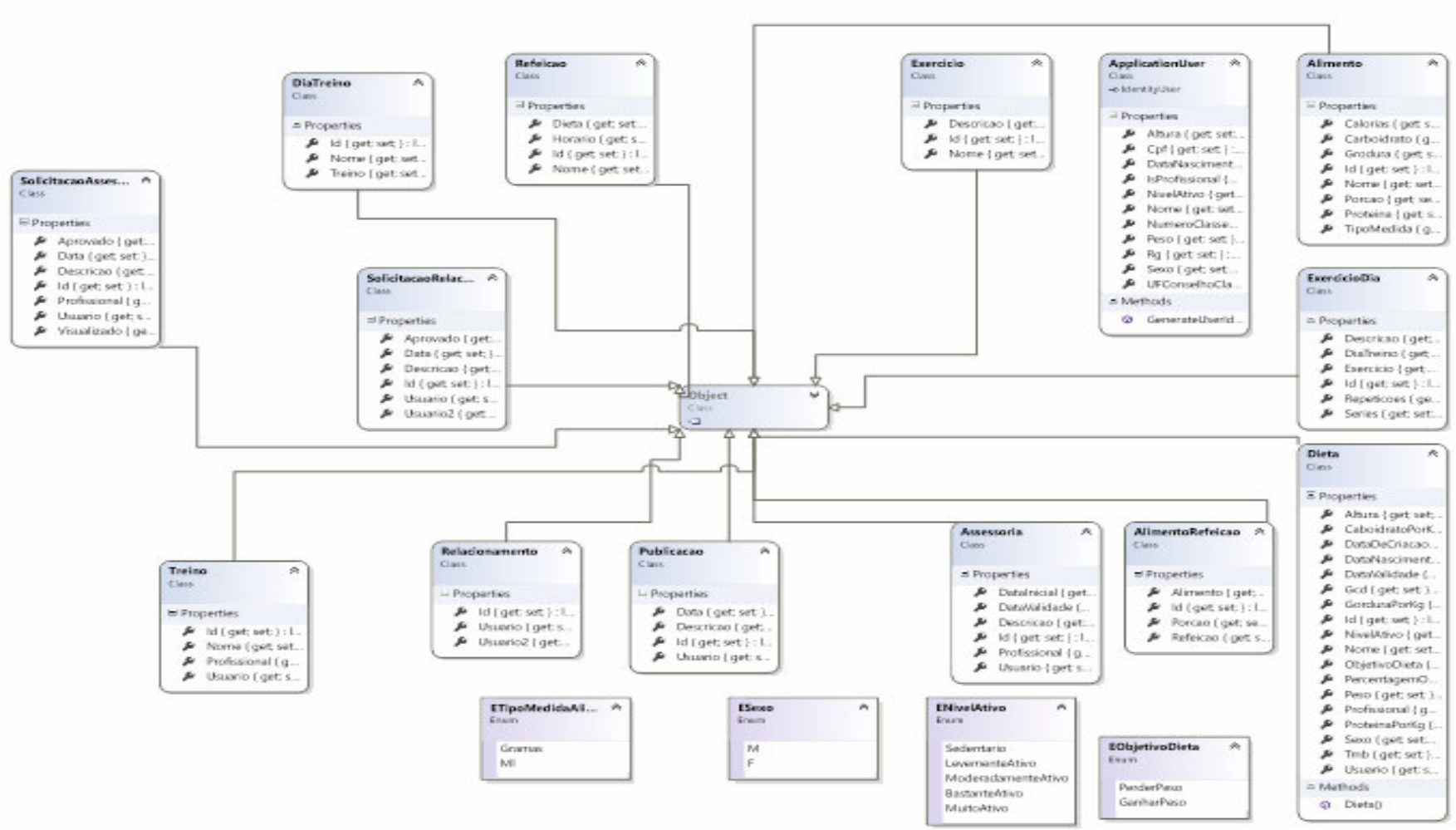

Figura 2 - Diagrama de classe

O diagrama de entidades e relacionamentos do banco de dados é apresentado na Figura 3.

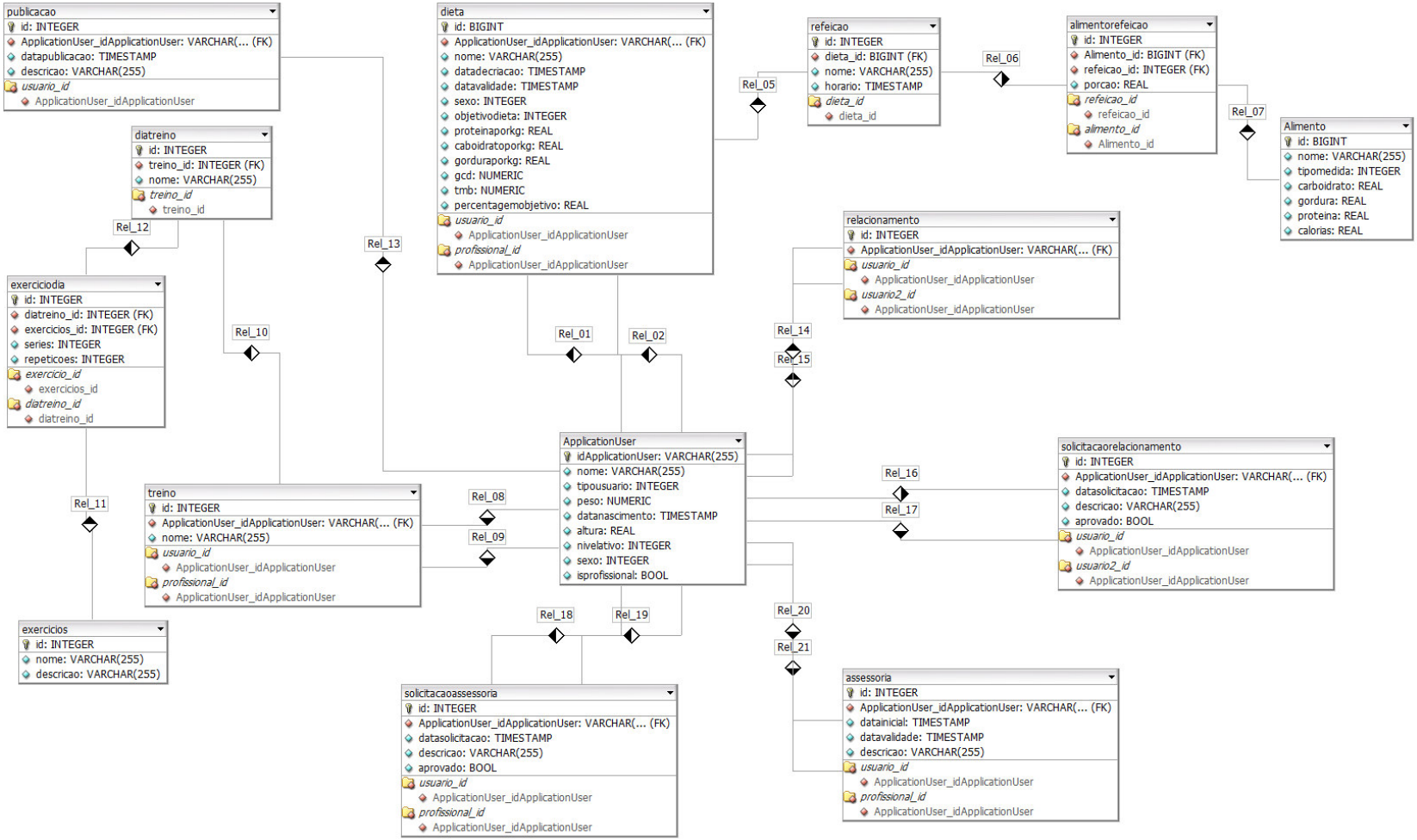

Figura 3 - Diagrama de entidades e relacionamentos do banco de dados

4.3 APRESENTAÇÃO DO SISTEMA

A seguir são apresentados os leiautes e as telas da aplicação.

Na Figura 4 está representada a tela inicial do sistema, na qual estão os campos de *login* e de cadastro no sistema.

Figura 4 - Tela inicial do sistema

Na Figura 5 está representada a tela de cadastro de usuário.

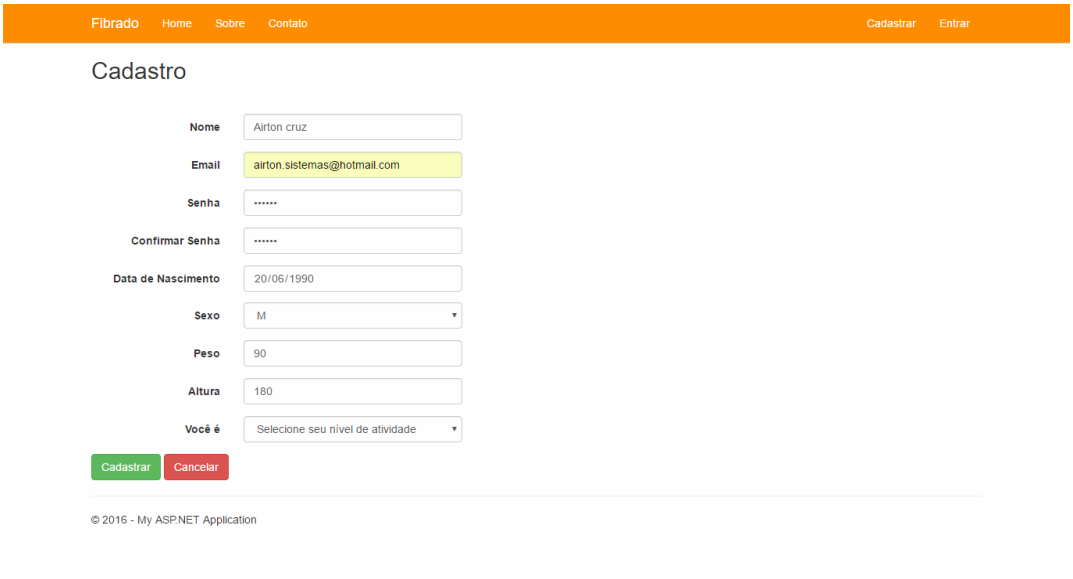

Figura 5 - Tela de cadastro de usuário

Na Figura 6 está representada a tela de *login* do usuário.

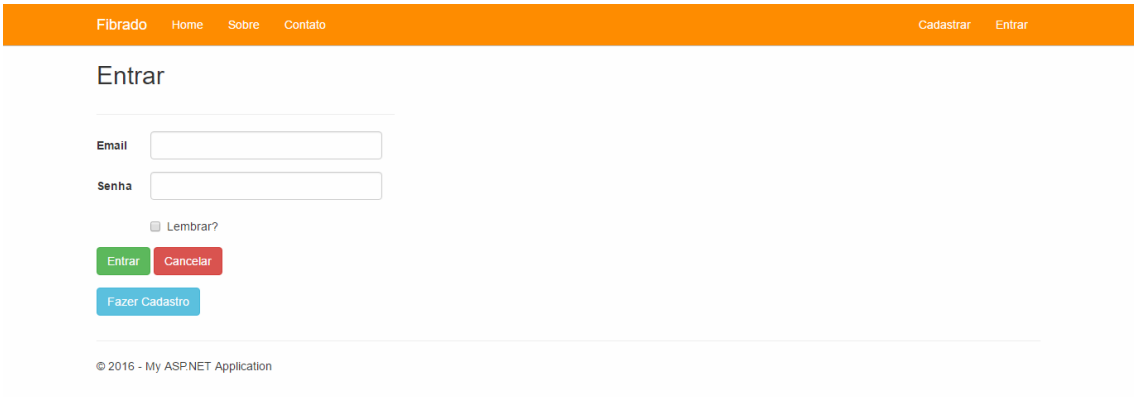

Figura 6 - Tela de login do usuário

Na Figura 7 está representada a tela inicial do usuário.

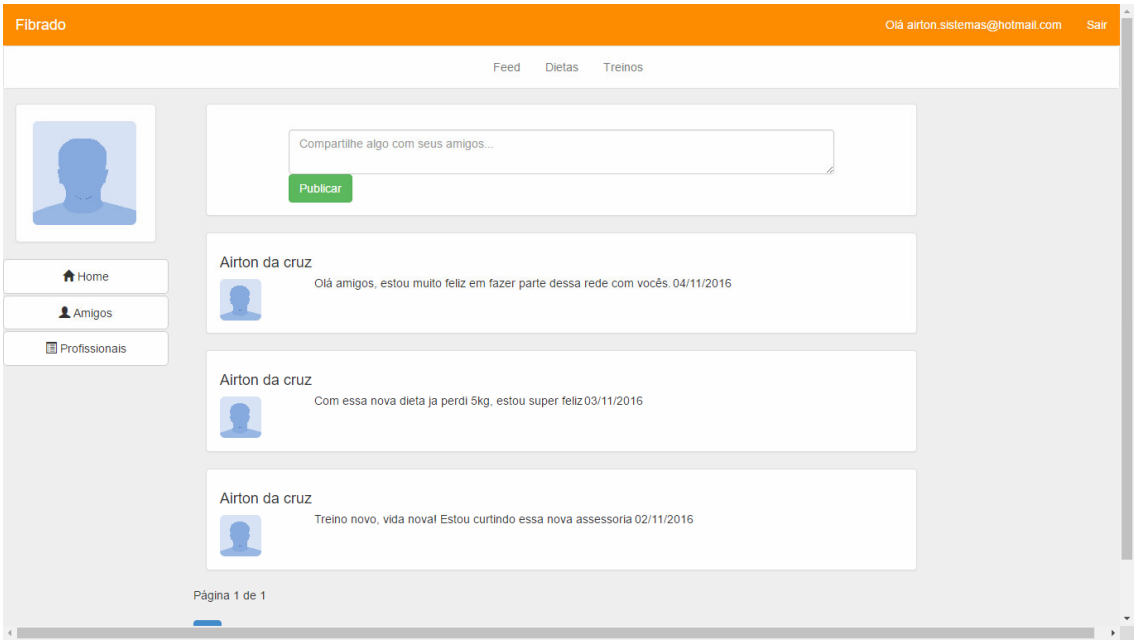

Figura 7 - Tela inicial do usuário

Na Figura 8 está representada a tela de listagem de dietas.

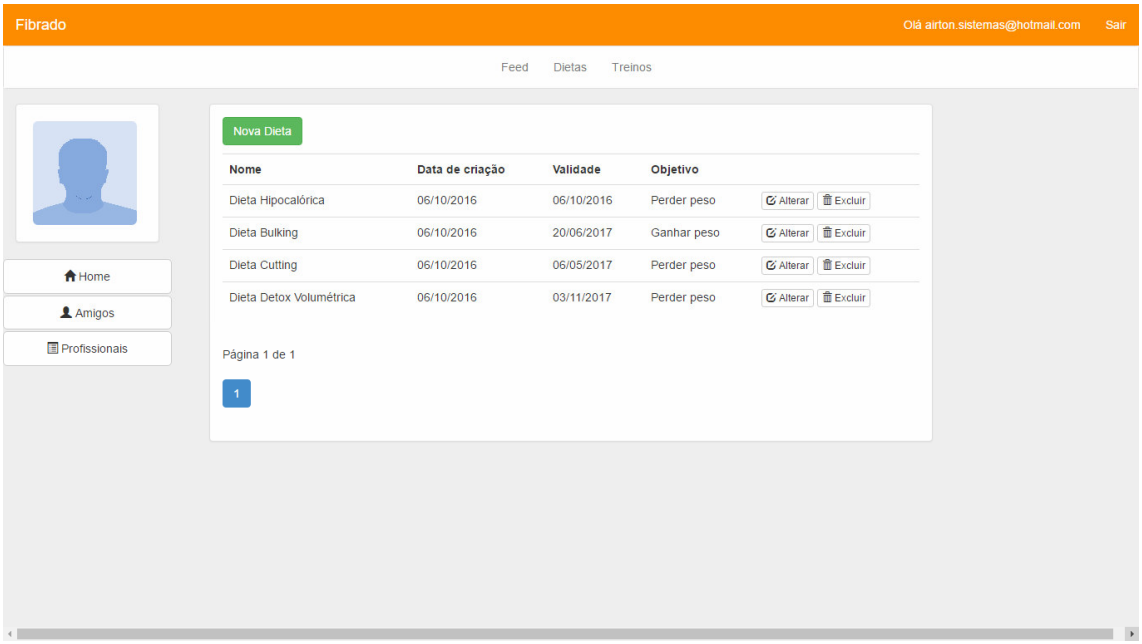

Figura 8 - Tela de listagem de dietas

Na Figura 9 está representada a tela de cadastro de dieta. Nessa tela, TMB significa Taxa de Metabolismo Basal e GCD significa Gasto Calórico Diário.

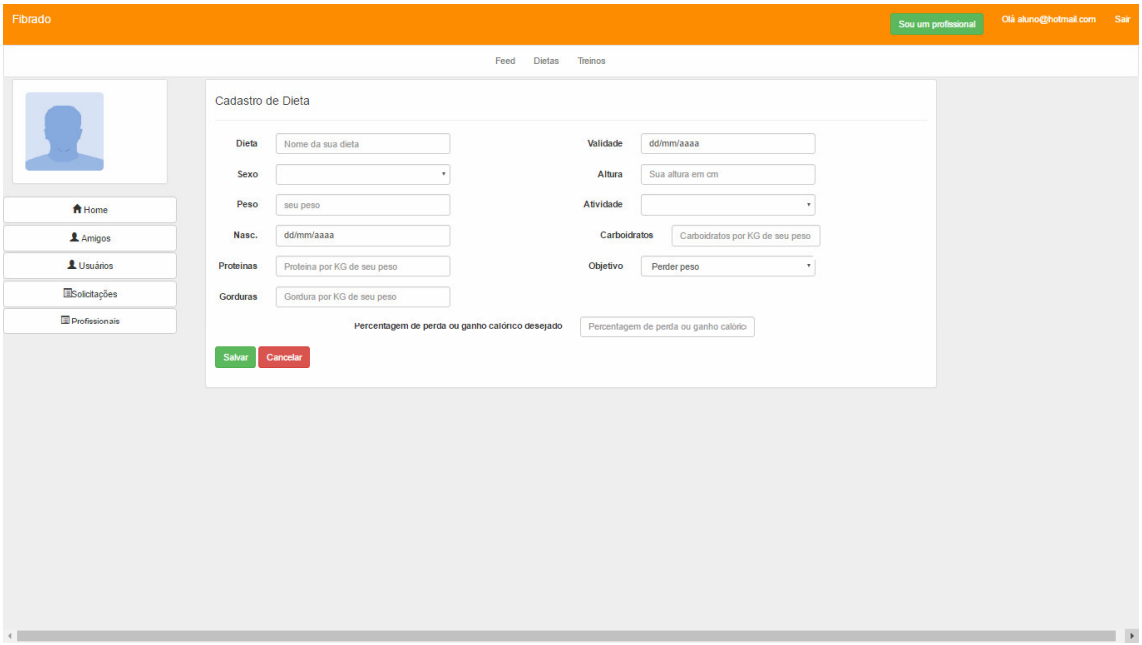

Figura 9 - Tela de cadastro de dietas

Na Figura 10 está representada a tela alimentos dentro de uma dieta.

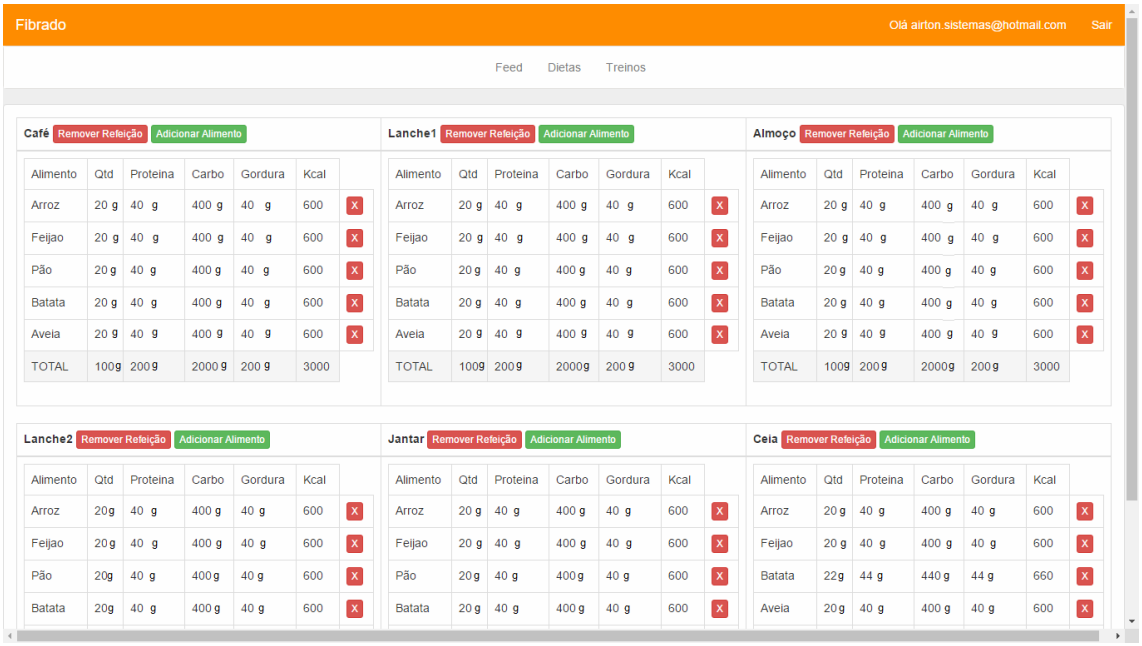

Figura 10 - Tela de montagem de dietas

Na Figura 11 está representada a tela de listagem de treinos.

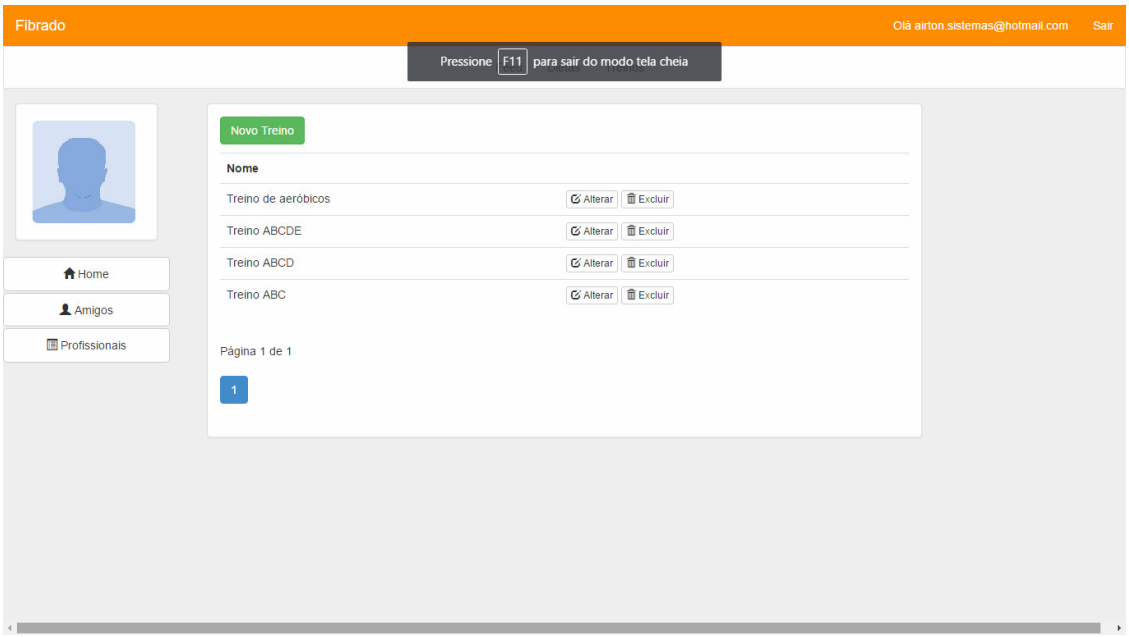

Figura 11 - Tela de listagem de treinos

Na Figura 12 está representada a tela de cadastro de treino.

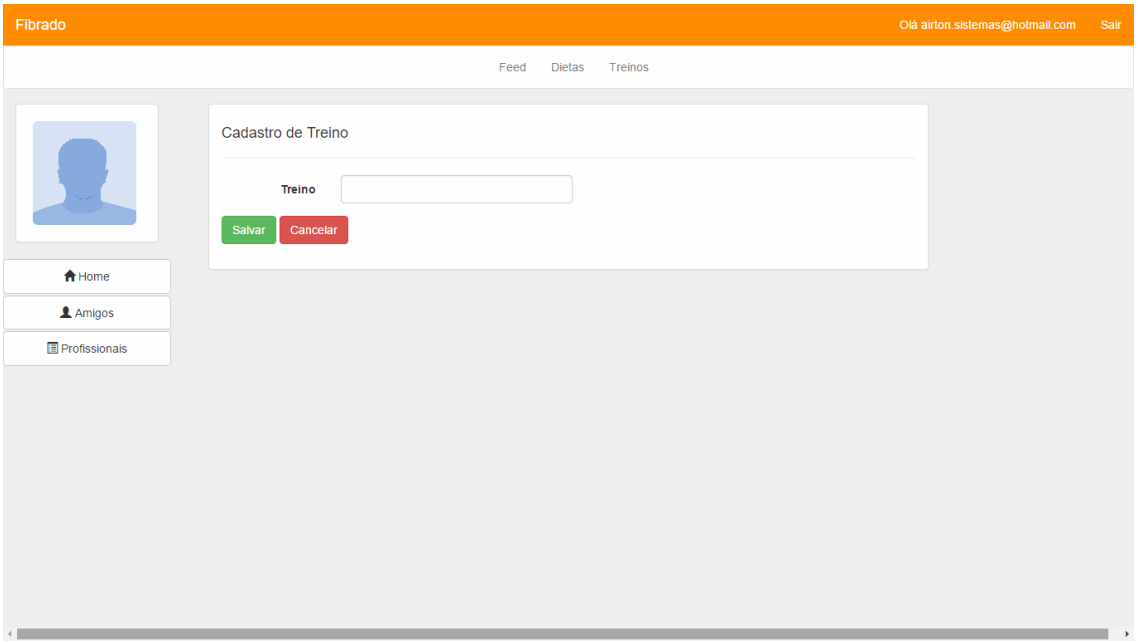

Figura 12 - Tela de cadastro de treinos

Na Figura 13 está representada a tela da relação de exercícios de um treino.

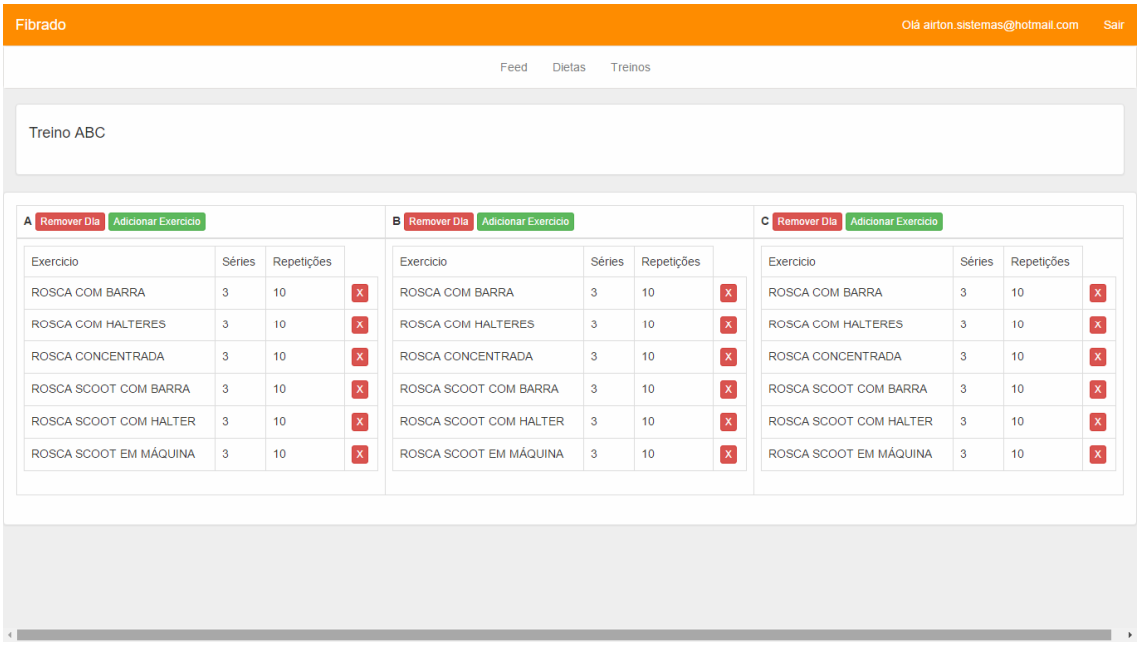

Figura 13 - Tela de montagem de treinos

4.4 IMPLEMENTAÇÃO DO SISTEMA

O sistema usa leiautes que dividem cada tela entre menu superior, lateral, rodapé e conteúdo central. A aplicação proposta possui três leiautes que são referenciados em determinadas telas. Na Listagem 1 está exposto o leiaute utilizado na tela principal "_Layout.cshtml". Este leiaute é responsável por agrupar cada parte da tela por meio de "@RencerSection" e também é o responsável por renderizar o conteúdo da página dentro da tag "@RenderBody()". Ao final é feita a referência ao JQuery e ao Bootstrap.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

</head>
<body>
    @if (IsSectionDefined("menusuperior"))
    {
        @RenderSection("menusuperior")
    }
    else
    {
        @Html.Partial("_MenuSuperiorCadastro")
    }

    @RenderBody()

    @if (IsSectionDefined("rodape"))
    {
        @RenderSection("rodape")
    }
    else
    {
        @Html.Partial("_Rodape")
    }

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

Listagem 1 – Código do leiaute da tela principal

Na Listagem 2 está o código da *view* "Index.cshtml" e a referência ao leiaute utilizado. O conteúdo contido na *view* será adicionado à tag "@RenderBody()" do leiaute "_Layout.cshtml" como mostrado na Listagem1.

```
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<img src="~/Imagens/Home2.jpg" />
```

Listagem 2 – Código da view "Index.cshtml

Na Listagem 3 está o código da partial view "_MeuSuperioCadastro.cshtml", que é um curto trecho de código que pode ser usado em diversas outras *views* dentro do projeto apenas realizando uma chamada. No caso em questão ela é usada como menu superior do leiaute "_Layout.cshtml".

```
class="navbar navbar-default navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <a class="navbar-brand" href="~/Home/Index">ProFit</a>
        </div>
        <center>
            <div class="navbar-inner" id="navbar-main">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Como Funciona", "Index", "Sobre")</li>
                </ul>
                @Html.Partial("_Login")
                <form class="navbar-form navbar-right" role="search">
                    <div class="form-group">
                        <a href="~/Profissional/CadastroProfissional"><input
type="button" value="Registrar Profissional" class="btn btn-warning"></a>
                        <a href="~/Aluno/CadastroAluno"><input type="button"
value="Registrar Aluno/Paciente" class="btn btn-warning"></a>
                    </div>
                </form>
            </div>
        </center>
    </div>
</div>
```

Listagem 3 – Código partial view MeuSuperioCadastro.cshtml

Na Listagem 4 está representada a implementação da classe "Usuario.cs" que por sua vez herda da classe padrão "BaseEntity.cs" na qual é implementada a propriedade ID. Como a classe representada será persistida no banco de dados, todas as propriedades que serão salvas em banco devem possuir "virtual". Os "getters" e "setters" em aplicações "ASP.NET" não precisam ser implementados, somente representados. O método construtor alimenta a propriedade "DataCadastro" com a data atual e a propriedade "GuidEmail" com uma nova *guid* para verificação futura do *e-mail*.

```
public class Usuario : BaseEntity
    {
        public virtual string Nome { get; set; }
        public virtual string SobreNome { get; set; }
        public virtual string EMail { get; set; }
        public virtual string Senha { get; set; }
        public virtual DateTime DataCadastro { get; set; }
        public virtual string GuidEmail { get; set; }
        public virtual bool IsEmailValido { get; set; }
        public virtual byte[] FotoPerfil { get; set; }
        public virtual Cidade Cidade { get; set; }
        public virtual string Telefone { get; set; }
        public virtual string Sobre { get; set; }

        public Usuario()
        {
            this.DataCadastro = DateTime.Now;
            this.GuidEmail = Guid.NewGuid().ToString();
        }

        #region Metodos

        public override bool Equals(object obj)
        {
            if (obj == null || !(obj is Usuario))
                return false;

            return this.GetHashCode() == obj.GetHashCode();
        }
        public override int GetHashCode()
        {
            return Id.GetHashCode();
        }

        #endregion
    }
```

Listagem 4 – Classe Usuario.cs

Na Listagem 5 está representada a classe de mapeamento em banco de dados referente a classe “Usuario.cs”.

```
public sealed class UsuarioMap : ClassMap<Usuario>
    {
        public UsuarioMap()
        {
            Table("usuario");
            Id(x => x.Id);
            Map(x => x.Nome).Length(70).Not.Nullable();
            Map(x => x.SobreNome).Length(50).Not.Nullable();
            Map(x => x.EMail).Length(30).Not.Nullable();
            Map(x => x.Senha).Length(8).Not.Nullable();
            Map(x => x.DataCadastro).Not.Nullable();
            Map(x => x.GuidEmail).Not.Nullable();
            Map(x => x.FotoPerfil);
            Map(x => x.Telefone).Length(12);
            Map(x => x.Sobre).Length(300);
            References(x => x.Cidade);
        }
    }
```

Listagem 5 – Mapeamento em banco de dados da classe Usuário.cs

Na Listagem 6 está representada a implementação da classe “Profissional.cs” que herda de “Usuario.cs”.

```
public class Profissional : Usuario
    {
        public virtual string Cpf { get; set; }
        public virtual string Rg { get; set; }
        public virtual string NumeroCref { get; set; }
        public virtual string NumeroCrn { get; set; }
        public virtual EUF? UFCref { get; set; }
        public virtual EUF? UfCrn { get; set; }
        public virtual bool IsConselhoValidado { get; set; }
        public virtual ETitulacao? Titulacao { get; set; }

        #region Metodos

        public override bool Equals(object obj)
        {
            if (obj == null || !(obj is Profissional))
                return false;

            return this.GetHashCode() == obj.GetHashCode();
        }
        public override int GetHashCode()
        {
            return Id.GetHashCode();
        }

        #endregion
    }
```

Listagem 6 – Classe Profissional.cs

Na Listagem 7 está implementada a classe de mapeamento de banco de dados referente a classe “Profissional.cs”.

```
public sealed class ProfissionalMap : SubclassMap<Profissional>
    {
        public ProfissionalMap()
        {
            Table("profissional");
            KeyColumn("ID");
            Map(x => x.Cpf).Length(11).Not.Nullable();
            Map(x => x.Rg).Length(11).Not.Nullable();
            Map(x => x.NumeroCref).Length(6);
            Map(x => x.NumeroCrn).Length(6);
            Map(x => x.UFCref).CustomType<EUF>();
            Map(x => x.UfCrn).CustomType<EUF>();
            Map(x => x.IsConselhoValidado);
            Map(x => x.Titulacao).CustomType<ETitulacao>();
        }
    }
```

Listagem 7 – Classe de mapeamento do banco de dados da classe Professional.cs

Em aplicações “ASP.NET” todas as requisições devem passar por um *controller*, seja para retornar dados ou uma *view*. Por padrão a aplicação buscará um *controller* chamado “Home” e uma *view* chamada “Index.cshtml” na primeira requisição. Na Listagem 8 está

representada a classe “HomeController.cs” com o método que retorna a *view* “Index.cshtml”. Esse *controller* herda de “BaseController” na qual estarão implementados todos os métodos genéricos e utilizados em diversos *controllers*, como acesso a sessão, usuário logado e operações de *Create, Retrieve, Update e Delete* (CRUD) básicos.

```
public class HomeController : BaseController
    {
        #region Index
        public ActionResult Index()
        {
            return View();
        }
        #endregion
    }
```

Listagem 8 – Classe “HomeController.cs

Na Listagem 9 está representada a implementação da classe “ProfissionalController.cs” que herda de “BaseController.cs” e realiza as operações de cadastro de profissional e validações de regra de negocio.

```
public class ProfissionalController : BaseController
    {
        // GET: Profissional
        public ActionResult CadastroProfissional()
        {
            CadastroProfissional profissional = new CadastroProfissional();

            return View(profissional);
        }

        [HttpPost]
        public ActionResult CadastroProfissional(CadastroProfissional model)
        {
            if (string.IsNullOrEmpty(model.NumeroCref) &&
string.IsNullOrEmpty(model.NumeroCrn))
            {
                ModelState.AddModelError("", "É obrigatório informar ao menos um
número de CREF ou CRN");
            }

            if (!string.IsNullOrEmpty(model.NumeroCref) && model.UFCref == null)
            {
                ModelState.AddModelError("UFCref", "Informe o UF");
            }

            if (!string.IsNullOrEmpty(model.NumeroCrn) && model.UfCrn == null)
            {
                ModelState.AddModelError("UfCrn", "Informe o UF");
            }

            if (ModelState.IsValid)
            {
                CadastroProfissionalSave(model);

                return View("CadastroProfissionalSucesso", model);
            }
```

```
            return View(model);
        }

        public ActionResult CadastroProfissionalSucesso(CadastroProfissional model)
        {
            return View(model);
        }

        private void CadastroProfissionalSave(CadastroProfissional model)
        {
            var profissional = new Profissional();

            profissional.Nome = model.Nome;
            profissional.SobreNome = model.SobreNome;
            profissional.Cpf = model.Cpf;
            profissional.Rg = model.Rg;
            profissional.NumeroCref = model.NumeroCref;
            profissional.NumeroCrn = model.NumeroCrn;
            profissional.UFCref = model.UFCref;
            profissional.UfCrn = model.UfCrn;
            profissional.EMail = model.EMail;
            profissional.Senha = model.Senha;

            //EmailHelper.SendEmailCadastroProfissional(profissional);
            SaveHandling(profissional);
        }
    }
```

Listagem 9 – Classe ProfissionalController.cs

Na Listagem 10 está representada a tela de cadastro do profissional. A referência ao "jqueryval" é responsável pelos *scripts* de validação de campos e de regra de negócios. Todos os campos possuem a propriedade @Html.ValidationMessageFor() que será responsável por exibir a mensagem de validação. E no topo da tela fica a propriedade @Html.ValidationSummary() que será responsável por exibir a mensagem de erro em validações de regra de negócio.

```
@model Portal.Models.CadastroProfissional

@{
    ViewBag.Title = "Cadastro de Profissional";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Cadastro de Profissional</h2>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Fique mais próximo de seus alunos</h4>
        <hr />
        <div class="row">
            @Html.ValidationSummary(true, "", new { @class = "text-danger" })
            <div class="col-md-4">
                <div class="form-group">
                    @Html.LabelFor(model => model.Nome, htmlAttributes: new { @class =
"control-label col-md-4" })
```

```
                    <div class="col-md-8">
                        @Html.EditorFor(model => model.Nome, new { htmlAttributes =
new { @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.Nome, "", new {
@class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.Cpf, htmlAttributes: new { @class =
"control-label col-md-4" })
                    <div class="col-md-8">
                        @Html.EditorFor(model => model.Cpf, new { htmlAttributes = new
{ @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.Cpf, "", new {
@class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.NumeroCref, htmlAttributes: new {
@class = "control-label col-md-4" })
                    <div class="col-md-8">
                        @Html.EditorFor(model => model.NumeroCref, new {
htmlAttributes = new { @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.NumeroCref, "", new
{ @class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.NumeroCrn, htmlAttributes: new {
@class = "control-label col-md-4" })
                    <div class="col-md-8">
                        @Html.EditorFor(model => model.NumeroCrn, new { htmlAttributes
= new { @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.NumeroCrn, "", new {
@class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.EMail, htmlAttributes: new { @class
= "control-label col-md-4" })
                    <div class="col-md-8">
                        @Html.EditorFor(model => model.EMail, new { htmlAttributes =
new { @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.EMail, "", new {
@class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.ConfirmPassword, htmlAttributes: new
{ @class = "control-label col-md-4" })
                    <div class="col-md-8">
                        @Html.EditorFor(model => model.ConfirmPassword, new {
htmlAttributes = new { @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.ConfirmPassword, "",
new { @class = "text-danger" })
                    </div>
                </div>
            </div>
            <div class="col-md-8">
                <div class="form-group">
                    @Html.LabelFor(model => model.SobreNome, htmlAttributes: new {
@class = "control-label col-md-2" })
```

```
                    <div class="col-md-10">
                        @Html.EditorFor(model => model.SobreNome, new { htmlAttributes
= new { @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.SobreNome, "", new {
@class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.Rg, htmlAttributes: new { @class =
"control-label col-md-2" })
                    <div class="col-md-10">
                        @Html.EditorFor(model => model.Rg, new { htmlAttributes = new
{ @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.Rg, "", new { @class
= "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.UFCref, htmlAttributes: new { @class
= "control-label col-md-2" })
                    <div class="col-md-10">
                        @Html.EnumDropDownListFor(model => model.UFCref, "Selecione o
UF", new { @class = "form-control", type = "text" })
                        @Html.ValidationMessageFor(model => model.UFCref, "", new {
@class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.UfCrn, htmlAttributes: new { @class
= "control-label col-md-2" })
                    <div class="col-md-10">
                        @Html.EnumDropDownListFor(model => model.UfCrn, "Selecione o
UF", new { @class = "form-control", type = "text" })
                        @Html.ValidationMessageFor(model => model.UfCrn, "", new {
@class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.Senha, htmlAttributes: new { @class
= "control-label col-md-2" })
                    <div class="col-md-10">
                        @Html.EditorFor(model => model.Senha, new { htmlAttributes =
new { @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.Senha, "", new {
@class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <input type="submit" value="Cadastrar" class="btn btn-success" />
                    </div>
                </div>
            </div>
        </div>
    </div>
}

@section Scripts{
    @Scripts.Render("~/bundles/jqueryval")
}
```

Listagem 10 – Código da tela de cadastro de profissional

Na Listagem 11 está representada a implementação da classe "CadastroProfissional.cs" que faz o papel de "Model" e possui todas as propriedades que possuem necessidade de exibição na tela de cadastro e suas validações por meio da classe "DataAnnotations".

```
public class CadastroProfissional : CadastroUsuario
    {
        public CadastroProfissional()
        {
            //UfList = new List<SelectListItem>();
        }

        [Required(ErrorMessage = "Digite o CPF.")]
        [Display(Name = "CPF")]
        [MinLength(11, ErrorMessage = "O tamanho mínimo do CPF são 11 caracteres.")]
        [StringLength(11, ErrorMessage = "O tamanho máximo do CPF são 11
caracteres.")]
        [Remote("ValidaCpfUnico", "Util", ErrorMessage = "CPF já cadastrado.")]
        public string Cpf { get; set; }

        [Required(ErrorMessage = "Digite o RG.")]
        [Display(Name = "RG")]
        [MinLength(8, ErrorMessage = "O tamanho mínimo do RG são 8 caracteres.")]
        [StringLength(11, ErrorMessage = "O tamanho máximo do RG são 8 caracteres.")]
        public string Rg { get; set; }

        [Display(Name = "Numero CREF")]
        [MinLength(6, ErrorMessage = "O tamanho mínimo do CREF são de 6 caracteres.")]
        public string NumeroCref { get; set; }

        [Display(Name = "UF CREF")]
        public EUF? UFCref { get; set; }

        [Display(Name = "Numero CRN")]
        [MinLength(6, ErrorMessage = "O tamanho mínimo do CRN são de 6 caracteres.")]
        public string NumeroCrn { get; set; }

        [Display(Name = "UF CRN")]
        public EUF? UfCrn { get; set; }
    }
```

Listagem 11 – Classe CadastroProfissional.cs

5 CONCLUSÃO

O objetivo deste trabalho foi a implementação de um sistema *web* para intermediar assessorias esportivas e nutricionais. A aplicação foi projetada para atender as necessidades de profissionais da área de educação física, nutrição e seus respectivos clientes, com intuito de facilitar e agilizar o atendimento entre as partes envolvidas.

No desenvolvimento foram utilizadas tecnologias para plataforma *web* bem como boas práticas de engenharia de software visando um desenvolvimento de código limpo e de fácil manutenção. A parte visual foi projetada para ser totalmente responsiva e simples, evitando assim a necessidade de grande capacidade de processamento por parte do navegador e tornando o tráfego de dados mais leve e ágil.

Dentre as tecnologias utilizadas destacam-se a biblioteca CSS Bootstrap que vem sendo bastante utilizada para o desenvolvimento de interfaces visuais para *web*. Outra ferramenta que tornou a aplicação mais flexível foi o NHibernate, um *framework* ORM que tem por finalidade o mapeamento de classes C# em objeto relacional, evitando a codificação de códigos *Structured Query Language* (SQL) e possibilitando a alteração do banco de dados sem a necessidade de manutenção nos códigos.

Uma dificuldade encontrada durante o desenvolvimento do projeto foi em relação à interação entre o *ASP.NET Identity* que originalmente interage com o *Entity Framework* ao invés do *NHibernate*. Portanto, foi necessário realizar alterações estruturais na persistência de dados, permitindo assim realizar o controle de usuários e níveis de acesso com o *Identity* e o *NHibernate*. Devido a essas alterações foi necessário realizar uma vasta pesquisa em documentações, sendo constatada a falta de materiais para consulta na *web* que pudessem efetivamente auxiliar na resolução de problemas e nas dificuldades. Essa dificuldade é ampla se considerados materiais em português, mas a dificuldade se mantém, ainda que menor, para conteúdo em inglês.

REFERÊNCIAS

BERNARDI, Mario Luca; DI LUCCA, Giuseppe Antonio; DISTANTE, Damiano. **Model-driven fast prototyping of RIAs: from conceptual models to running applications**. IEEE International Workshop on Software Engineering for Web Application Development (SEWAD-2014), IEEE Press, p. 250-258, 2014.

CHOLIA, Shreyas; SKINNER, David; BOVERHOF, Joshua. **NEWT: a RESTful service for building high performance computing web applications**. Gateway Computing Environments Workshop (GCE), 2011, p. 1-11.

DI LUCCA, Giuseppe Antonio et al. **WARE: a tool for the reverse engineering of web applications**. In: 6th European Conference on Software Maintenance and Reengineering, 2002, p. 241–250.

FRATERNALI, Piero; ROSSI, Gustavo; SÁNCHEZ-FIGUEROA, Fernando. Rich internet applications. **IEEE Computer Society**, may/june 2010, p. 9-12

FREIRE, Tiago Onofre. **A importância da alimentação na prática de atividade física**. 2012. Disponível em: <http://www.isaudebahia.com.br/noticias/detalhe/noticia/a-importancia-da-alimentacao-na-pratica-de-atividade-fisica/>. Acesso em: 17 set. 2015.

HASSAN, Ahamed E.; HOLT, Richard C. **Towards a better understanding of web applications**. In: 3rd International Workshop on Web Site Evolution, 2001, p. 112–116.

HTML5 E CSS3. **HTML5 and CSS3 support**. Disponível em:<http://www.findmebyip.com/litmus>. Acesso em: 31 jul. 2015.

LAUDON, Kenneth C.; LAUDON, Jane Price. **Sistemas de informação gerenciais**. 7 ed. São Paulo: Pearson Prentice Hall, 2007.

LI-LI, Chen; ZHENG-LONG, Liu. **Design of rich client web architecture based on HTML5**. In: Fourth International Conference on Computational and Information Sciences, 2012, p. 1009-1012.

MINISTÉRIO DA SAÚDE. SECRETARIA DE VIGILÂNCIA EM SAÚDE. **Vigitel Brasil 2013: vigilância de fatores de risco e proteção para doenças crônicas por inquérito telefônico.** Disponível em: <http://www.prefeitura.sp.gov.br/cidade/secretarias/upload/saude/arquivos/morbidade/Vigitel-2013.pdf.> Acesso em: 15 set. 2015.

OLIVEIRA, Djalma de P. R. de. **Sistemas de informações gerenciais: estratégicas, táticas e operacionais**. 13 ed. São Paulo: Atlas, 2010.

PAVLIĆ, Daniel; PAVLIĆ, Mile; JOVANOVIĆ, Vladan. **Future of internet technologies**. International Convention on Information and Communication Technology, Electronics and Microelectronics, 2012, p. 1366-1371.

PEINTNER, Daniel; KOSCH, Harald; HEUER, Jörg. **Efficient XML interchange for rich internet applications**. In: IEEE International Conference on Multimedia and Expo, 2009, p. 149 – 152.

QI, Long. Study on HTML5 standard of a new generation network technology. **Science and technology information**, n. 10, 2011.

SHARMA, Rishabh; KUMAR, Sanjay; TRIVEDI, Munesh Chandra. **Mobile cloud computing: a needed shift from cloud to mobile cloud**. In: 5th International Conference on Computational Intelligence and Communication Networks, 2013, p. 536-539.

TRAMONTANA, Porfirio; AMALFITANO, Domenico; FASOLINO, Anna Rita. **Reverse engineering techniques: from web applications to rich internet applications**. In: 15th IEEE International Symposium on Web Systems Evolution, 2013, p. 83-86.