FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

VINÍCIUS DE SOUZA AVANSINI

SOCIETY AGENDOR:

Aplicativo para agendar campo de futebol society

AMERICANA, SP

2017

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

VINÍCIUS DE SOUZA AVANSINI

SOCIETY AGENDOR:

Aplicativo para agendar campo de futebol society

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Prof. Dr. Kleber de Oliveira Andrade.

Área de concentração: Aplicativos móveis.

AMERICANA, SP

2017

A964s AVANSINI, Vinícius de Souza.

Society agendor: aplicativo para agendar campo de futebol society. / Vinícius de Souza Avansini. – Americana, 2017.

88f.

Monografia (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Dr. Kleber de Oliveira Andrade

1. Dispositivos móveis – aplicativos. I. ANDRADE, Kleber de Oliveira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.519

Vinícius de Souza Avansini

SOCIETY AGENDOR

Aplicativo para agendar campo de futebol society

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana.
Área de concentração: Aplicativos móveis.

Americana, 12 de dezembro de 2017.

Banca Examinadora:

Prof. Dr. Kleber de Oliveira Andrade (Presidente)
Doutor
FATEC Americana

Prof. Ms. Diógenes de Oliveira (Membro)
Mestre
FATEC Americana

Prof. Ms. Eduardo Antonio Vicentini (Membro)
Mestre
FATEC Americana

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus, pois sem ele eu não teria forças para a realização deste trabalho.

Aos meus pais, que me deram forças para me empenhar e não desistir no meio do caminho.

Ao meu orientador, Prof. Dr. Kleber de Oliveira Andrade por ter me guiado durante o processo de desenvolvimento deste trabalho.

E a todos que me ajudaram direta ou indiretamente com o projeto.

“*Podemos progredir todos os dias. Cada passo pode ser frutífero. Ainda assim, haverá diante de nós caminhos longos e infinitos. Sabemos que nunca chegaremos ao fim da jornada. Mas isso, longe de nos desencorajar, só aumentará a alegria e a glória da subida*”.

Winston Churchill.

RESUMO

Nos últimos anos a sociedade percebeu como a tecnologia está crescendo no espaço de suas vidas e os ajudando, tanto com os pequenos afazeres do lar ou como atividades complexas, como editores de imagens, programas de transações bancárias e etc. Através da detecção de que poucos estabelecimentos fornecem a possibilidade de locação de campo de *society*, através de aplicativos móveis, foi desenvolvido o aplicativo Society Agendor utilizando o *framework* Xamarin em conjunto com a linguagem de programação C#, sendo o principal enfoque do trabalho, além de uma API, que é responsável por algumas transações do banco de dados na nuvem da Microsoft, o Microsoft Azure. Durante os capítulos foram apresentados os processos de desenvolvimento do aplicativo, detalhando e documentando desde a metodologia de desenvolvimento incremental, os requisitos, o planejamento, os recursos e as tecnologias utilizadas, os diagramas e por fim o então desenvolvimento, dando origem ao aplicativo construído.

Palavras Chave: Dispositivos Móveis – aplicativos; Desenvolvimento de Software; Engenharia de Software.

ABSTRACT

In the past few years society has realized how technology is increasing in their lives and helping them, both with simple housework and complex activities, such as image editors, banking software, etc. Through the detection that few establishments provide the possibility of renting a society football, through mobile applications, the application Society Agendor, the principal focus of this project, was developed using the framework Xamarin along with C# programming language, in addition to an API, which is responsible for some transactions in the Microsoft cloud database, Microsoft Azure. During the chapters, the application development processes were presented, detailing and documenting from incremental development methodology, requirements, planning, the resources and technologies used, diagrams and finally the development, giving rise to the built mobile application.

Keywords: *Mobile – applications; Software development; Software Engineering.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Nos últimos anos a sociedade percebeu como a tecnologia está crescendo no espaço de suas vidas, como ela está os ajudando, tanto com os pequenos afazeres do lar como: aspiradores inteligentes, geladeiras inteligentes, telefones inteligentes etc.; até as mais complexas ações que facilitam e automatizam, como: *softwares* para renderização[1] de imagens, *softwares* de agências bancárias, *softwares* para comunicação *online*.

A ideia do desenvolvimento de um aplicativo móvel neste trabalho, surge a partir da palavra *cross-platform*. Podendo ser definida como desenvolvimento de um *software* que tem a capacidade de compilar em vários sistemas operacionais móveis díspares, além do conceito "*write once, run anywhere*" (para o português, "escrever uma vez, rodar em qualquer lugar"), criado pelos "pais" da linguagem C, Ritchie e Thompson. Mas esta capacidade pode ter uma desvantagem, sendo que, quanto mais plataformas e SOs móveis, menor vai ser o número de recursos nativos disponíveis ao desenvolvedor, dificultando a criação e manipulação destes.

O autor deste trabalho percebeu que, há uma certa falta de inovação por parte de aluguel de campos de futebol *society* em sua cidade. Poderia haver um meio que facilitasse esta locação em qualquer hora do dia. Chegou então a uma ideia para este trabalho, onde se desenvolverá um aplicativo que facilite o agendamento do campo de futebol *society*.

O trabalho está organizado da forma que o leitor possa compreender desde as referências para a fundamentação deste, até a apresentação do aplicativo. No Capítulo 2, é apresentada a metodologia de desenvolvimento utilizada (subcapítulo 2.1), o processo de engenharia de *software,* que abrange os subcapítulos 2.2, 2.3, 2.5 2.6 e 2.7. O subcapítulo 2.4 é apresentado os recursos e ferramentas utilizadas no desenvolvimento do aplicativo, abrangendo desde a linguagem de programação até os serviços de nuvem que foram utilizados. E por fim, no Capítulo 3 é apresentado as considerações finais que o autor chegou ao desenvolver este aplicativo.

[1] Renderizar é o processo pelo qual pode-se obter o produto final de um processamento digital qualquer. Este processo aplica-se essencialmente em programas de modelagem 2D e 3D, bem como áudio e vídeo.

2 PROJETO DO APLICATIVO

Este capítulo abordará alguns dos tópicos mais importantes da Engenharia de Software, como: o tipo de método de desenvolvimento, engenharia de requisitos, planejamento do desenvolvimento, ferramentas utilizadas, modelagem – estabelecendo a diagramação do projeto utilizando a UML – e desenvolvimento, que irá abranger os diagramas de classe e sua explicação.

2.1. Metodologia de Desenvolvimento

A metodologia utilizada para o desenvolvimento do aplicativo, foi a metodologia incremental. Segundo SOMMERVILLE (2011) baseia-se “na ideia de desenvolver uma implementação inicial, expô-la aos comentários dos usuários e continuar por meio da criação de várias versões até que um sistema adequado seja desenvolvido”.

2.2. Engenharia de Requisitos

De acordo com SOMMERVILLE (2011), a *Requirements Engineering* ou para o português, Engenharia de Requisitos, pode ser definida como o processo de descobrir, analisar, documentar e verificar os requisitos de um sistema. Estes requisitos, segundo SOMMERVILLE (2011), podem ser definidos como as descrições, serviços e restrições que um sistema possui. Refletindo diretamente na compreensão de como o sistema irá funcionar. Normalmente os requisitos são classificados em dois tipos: Requisitos Funcionais e Requisitos Não Funcionais.

2.2.1. Requisitos Funcionais

Os requisitos funcionais refletem diretamente nos processos do sistema, ou seja, as funcionalidades e serviços do *software*. A Tabela 1 apresenta os requisitos funcionais do projeto.

Tabela 1 - Requisitos funcionais do projeto.

Identificação	Requisito Funcional	Categoria	Prioridade
RF001	O sistema deve possuir um botão que possibilite o login com o Facebook, possibilitando o resgate de informações do usuário e facilite seu cadastro.	Design e Programação	Média

RF002	O sistema deve exibir apenas os agendamentos do usuário “logado”.	Programação	Muito alta
RF003	Para se utilizar o sistema, o usuário deve entrar com sua conta do Facebook.	Programação	Alta
RF004	O sistema automaticamente deve excluir, do banco de dados, os agendamentos que já ocorreram.	Programação	Muito alta
RF005	O sistema deve apontar os estabelecimentos através do Google Maps.	Design e Programação	Alta
RF006	O sistema deve emitir uma mensagem de alerta caso o dispositivo não possua conexão com a internet.	Programação	Média
RF007	Apenas o usuário que agendou o local pode editar ou cancelar o agendamento.	Programação	Muito alta
RF008	O sistema deve emitir uma notificação uma hora antes do evento acontecer.	Programação	Média

Fonte: Elaborado pelo autor.

2.2.2. Requisitos Não Funcionais

Diferente dos requisitos funcionais, os requisitos não funcionais, não estão ligados diretamente aos processos e métodos, mas sim nas características do sistema. Normalmente especificam ou restringem o desempenho, a proteção ou a disponibilidade do sistema (SOMMERVILLE, 2011).

Tabela 2 - Requisitos não funcionais do projeto.

Identificação	Requisito Não Funcional	Categoria	Prioridade
RNF001	O aplicativo deve executar em celulares a partir da API [2] 19 do Android.	Programação	Muito alta
RNF002	O aplicativo não irá funcionar sem conexão com a internet.	Programação	Muito alta

[2] API, do inglês, *Application Programming Interface* – Interface de Programa de Aplicação (Ver explicação na Seção 2.4.4).

RNF003	Se o usuário deixar algum campo de digitação em branco, o sistema deve referenciá-lo com uma cor avermelhada.	Design e Programação	Média
RNF004	Minimizar a quantidade de campos de entrada de texto, tornando menos cansativo para o usuário.	Design	Alta
RNF005	Todos os métodos que possuem entrada de dados, devem possuir validação para saber se os campos estão nulos ou com espaço em branco.	Programação	Média
RNF006	Exibir mensagens de alerta caso algum campo de texto esteja vazio.	Programação	Baixa

Fonte: Elaborado pelo autor.

2.3. Planejamento

Este subcapítulo tem propósito de apresentar o planejamento do projeto através da elaboração de um cronograma.

Antes da apresentação do cronograma, deve-se levar em conta que o projeto foi iniciado pelo desenvolvimento do aplicativo e não pela documentação (sendo mais recomendado por todos na literatura da área). Finalizado o desenvolvimento do aplicativo, foi realizado o processo inverso do tradicional ciclo de vida de um sistema, que se dá início na análise de viabilidade, levantamento de requisitos e que vai até sua manutenção / “morte” do sistema.

A Figura 1 apresenta o cronograma do projeto.

Figura 1 - Cronograma do desenvolvimento.

	Modo da Tarefa	Nome da tarefa	% concluída	Início	Término	Predecessoras
1		**Início do Projeto**	**100%**	**Qua 05/07/17**	**Seg 06/11/17**	
2		**Planejamento e Desenvolvimento**	**100%**	**Qua 05/07/17**	**Sáb 04/11/17**	
3		Mock-up das telas do aplicativo (versão 1)	100%	Qua 05/07/17	Qui 06/07/17	
4		Desenvolvimento Tela de Login (versão 1)	100%	Qui 06/07/17	Seg 10/07/17	
5		Desenvolvimento Tela Inicial (versão 1)	100%	Qui 13/07/17	Qui 13/07/17	4
6		Desenvolvimento Tela de Perfil (versão 1)	100%	Sex 14/07/17	Sex 14/07/17	5
7		Desenvolvimento Tela de Splash Screen (versão 1)	100%	Qua 02/08/17	Sex 04/08/17	
8		Mock-up das telas do aplicativo (versão 2)	100%	Sex 11/08/17	Sex 11/08/17	22;23
9		Desenvolvimento Tela de Login (versão 2)	100%	Seg 04/09/17	Qua 06/09/17	
10		Desenvolvimento Cadastrar usuário	100%	Qui 07/09/17	Sáb 16/09/17	9
11		Desenvolvimento Login usuário	100%	Dom 17/09/17	Ter 19/09/17	10
12		Desenvolvimento Visualizar perfil de usuário	100%	Qua 20/09/17	Qui 21/09/17	11
13		Desenvolvimento Atualizar perfil de usuário	100%	Sex 22/09/17	Seg 25/09/17	12
14		Desenvolvimento de API	100%	Ter 03/10/17	Qua 04/10/17	
15		Desenvolvimento de Cadastrar agendamento	100%	Qui 05/10/17	Qua 18/10/17	14
16		Desenvolvimento Listar Agendamentos	100%	Qui 19/10/17	Sex 20/10/17	15
17		Desenvolvimento Tela de Edição de agendamento	100%	Sáb 21/10/17	Dom 22/10/17	16
18		Desenvolvimento Atualizar agendamento	100%	Seg 23/10/17	Qua 25/10/17	17
19		Desenvolvimento Apagar agendamento	100%	Qui 26/10/17	Seg 30/10/17	18
20		Desenvolvimento Notificação de jogo	100%	Qua 01/11/17	Sáb 04/11/17	
21		**Levantamento de Requisitos**	**100%**	**Seg 07/08/17**	**Qui 10/08/17**	
22		Requisito Funcional	100%	Seg 07/08/17	Ter 08/08/17	
23		Requisito Não Funcional	100%	Qua 09/08/17	Qui 10/08/17	22
24		Documentação	100%	Qua 25/10/17	Seg 06/11/17	
25		Diagrama de Casos de Uso	100%	Qua 25/10/17	Qui 26/10/17	
26		Diagrama de Classes	100%	Sex 03/11/17	Sex 03/11/17	
27		Cronograma do Projeto	100%	Seg 06/11/17	Seg 06/11/17	

Fonte: Elaborado pelo autor.

2.4. Recursos e Ferramentas

Esta Seção abordará os modelos, ferramentas e conceitos de programação que foram utilizados para o desenvolvimento do aplicativo.

O aplicativo foi desenvolvido para celulares que possuem o sistema operacional Android utilizando o Visual Studio como ambiente de desenvolvimento. Em conjunto à plataforma da Microsoft, foi utilizado a plataforma Microsoft Azure para a hospedagem de serviços de banco de dados e APIs na nuvem.

2.4.1. Xamarin

Em meados de junho do ano de 2000, a Microsoft anunciava o .NET, após este evento, uma companhia chamada Ximian - fundada por Miguel de Icaza e Nat Friedman - deram o primeiro passo em criar um projeto de código aberto (*open-source project*) chamado Mono, este que era uma implementação alternativa do compilador da linguagem C# e do .NET Framework para o sistema operacional Linux, ou seja, ao fazer esta implementação da linguagem e do *framework*, os desenvolvedores poderiam utilizá-la em ambientes Linux. No ano de 2011, a empresa já tinha sido

adquirida por uma outra empresa, a Novell, além disso, os fundadores da Ximian fundaram uma nova empresa chamada Xamarin. Esta adaptou o projeto Mono na base para formar soluções de desenvolvimento de aplicativos móveis *cross-platform* (PETZOLD, 2016).

Com o um crescimento constante do uso de ambientes de plataforma livre, foi anunciado e publicada no ano de 2014 uma versão *open-source* do compilador C#. Xamarin desempenhou um grande papel quando a .NET Foundation anunciou estar à disposição em relação a algumas de suas tecnologias para soluções de código aberto. Em março do ano de 2016, a Microsoft anunciava a compra da empresa Xamarin, com isso, levando a área de desenvolvimento *cross-platform* à comunidade de desenvolvedores Microsoft. Até então, antes de anunciar a aquisição, o desenvolvimento em Xamarin era pago, após a compra, a Microsoft cedeu a oportunidade de desenvolver aplicativos móveis, de graça, utilizando Xamarin a todos (PETZOLD, 2016).

Ao utilizar deste *framework*, o programador se beneficiará com a plataforma e suas soluções, sendo algumas:

- Desenvolver aplicativos utilizando a biblioteca .NET;
- Desenvolver aplicativos utilizando a linguagem C# para cinco plataformas móveis distintas (iOS, Android, Universal Windows Platform, o Windows 8.1 e Windows Phone);
- Desenvolver aplicativos nativos para cada plataforma (Xamarin.iOS, Xamarin.Android, Xamarin.UWP, Xamarin.Windows, Xamarin.WinPhone) utilizando as APIs nativas;
- Desenvolver aplicativos utilizando Xamarin.Forms com PCL (*Portable Class Library*, para o português, Biblioteca de Classes Portáteis) ou SAP (*Shared Asset Project*, para o português, Projeto de Recursos Partilhados);

Xamarin é dividido em duas vertentes de programação, o desenvolvimento nativo para cada plataforma ou o desenvolvimento utilizando Xamarin.Forms, que utiliza um projeto PCL (*Portable Class Library*) ou SAP (*Shared Asset Project*) para compartilhar o código entre as plataformas específicas.

- **SAP (*Shared Asset Project*):** Este projeto contém arquivos de código e outros arquivos que são compartilhados com cada projeto de plataforma, tornando-se essencialmente parte de cada projeto em cada plataforma.
- **PCL (*Portable Class Library*):** Todo o código comum do aplicativo torna-se uma DLL (*dynamic-link library*) que é referenciada por todos os projetos de plataformas individuais.

Mas esses dois projetos possuem benefícios ao se utilizar, quanto diferenças, que podem causar frustrações ao programador. A tabela abaixo deixa claro quais são as vantagens e desvantagens da utilização projetos.

Tabela 3 - Vantagens e desvantagens entre PCL e SAP.

Projeto	Vantagens	Desvantagens
PCL	**1.** Compartilhamento de código centralizado - escreva e teste o código em um único projeto que pode ser utilizado por outras bibliotecas e aplicações.	**1.** Como a mesma PCL é compartilhada por mais de uma plataforma, bibliotecas nativas específicas desta não podem ser referenciadas.
	2. Operação de "*refatoração*"[3] de código irão afetar todo o código incluído na solução (a PCL e o projeto específico de uma plataforma).	**2**. O subconjunto PCL pode não incluir as classes que estarão disponíveis no MonoTouch e Mono para Android.
	3. O projeto PCL pode ser referenciado facilmente por outros projetos na mesma solução, ou seu *assembly* de saída poderá ser compartilhado por outros e referenciados em suas soluções.	
SAP	**1.** Permite compartilhamento de código entre vários projetos.	**1.** O projeto não possui um *assembly* de saída. Durante a compilação os arquivos são tratados como parte do projeto de referência e compilados apenas nesse *assembly*.

[3] *Refatoração*, palavra de origem inglesa (*refactoring*), é o processo de melhoria de código de um sistema, onde visa retirar ou reduzir redundâncias.

	2. O código compartilhado pode ser derivado com base na plataforma usando diretrizes do compilador.	**2.** As refatorações que afetam o código dentro das diretrizes "inativas" do compilador não irão atualizar o código.
	3. O projeto pode referenciar bibliotecas nativas específicas da plataforma utilizando o código compartilhado.	

Fonte: Elaborado pelo autor.

O autor deste trabalho irá desenvolver um aplicativo para Android utilizando Xamarin.Forms com PCL. A escolha é devido a sua familiaridade com esta plataforma e sua facilidade de programar. A PCL oferece também o recurso de compartilhar a interface de usuário através da linguagem de marcação XAML[4], criada pela Microsoft, que é baseada em XML[5]. Xamarin.Forms em conjunto com uma PCL, são ideais para o desenvolvimento de protótipos e aplicativos profissionais fazendo com que a organização não precise contratar um time especializado para cada plataforma que for desenvolver.

2.4.2. MVVM

É um tipo de padrão de arquitetura, onde visa seguir boas práticas na organização do projeto (SOMMERVILLE, 2011). Aderindo a essas boas práticas, o desenvolvedor torna o projeto legível para futuros desenvolvedores, possibilitando também, o uso de testes, a distinção dos conceitos de cada classe (lógica de negócio e interface de usuário) etc. Com este princípio, podemos lembrar do padrão MVC[6], mas o padrão de arquitetura mais recomendado para o desenvolvimento de aplicações móveis, é o MVVM (*Model-View-ViewModel*), que atualizou o modo de como o MVC trata as interfaces de usuário, além de ter características que ajudam a lidar com a interface de usuário criada com XAML (PETZOLD, 2016).

[4] XAML, do inglês, *Extensible Application Markup Language*, para o português, Linguagem Extensível de Marcação de Aplicativos. Permite desenvolver em Xamarin.Forms, a interface de usuário utilizando uma linguagem de marcação ao invés de código.
[5] XML, do inglês, *Extensible Markup Language*, para o português, Linguagem de Marcação Extensível. Foi desenvolvida para a troca ou transporte de dados estruturados (SOMMERVILLE, 2011). Saiba mais em: <https://www.w3schools.com/xml/xml_whatis.asp>.
[6] MVC, sigla referente à *Model-View-Controller*, para o português, Modelo-Visão-Controlador. É um padrão de arquitetura muito utilizado em desenvolvimento para a Web (SOMMERVILLE, 2011).

A Tabela 4 detalha as funções de cada componente do padrão MVVM.

Tabela 4 - Funções dos Componentes MVVM.

Componentes	Funções
Model	A *Model* é responsável por prover os dados da aplicação, sendo classes que são usadas em conjunto com serviços ou repositórios que protegem, encapsulam o acesso aos dados. As classes modelos não têm acesso às propriedades públicas nem aos métodos da *View Model*.
View	Responsável pela camada de apresentação dos dados ao usuário, ou seja, é a interface de usuário. Na maioria das vezes é implementada por XAML. No padrão MVVM, a *View* não tem acesso à *Model*, os dados são repassados a ela através de propriedades ou eventos que estão implementados na *View Model*.
View Model	A *View Model* faz uma conexão entre a *Model* e a *View*. Como a *View*, a *View Model* não tem acesso às funcionalidades internas ligadas a *View*. A conexão entre *View* e *View Model* é dada através de chamadas ou acessos às propriedades da *View Model*, feita a partir de requisições da *View*. Então a *View Model* faz a mesma conexão com a *Model*, que é dada a partir da *View Model*, que faz chamadas a classe modelo ou acessa suas propriedades.

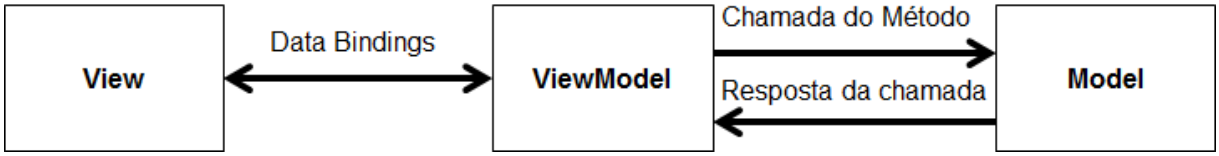
Fonte: Elaborado pelo autor.

Não se pode esquecer que o MVVM foi desenvolvido para tirar vantagem da utilização do XAML, contido na *View*. A conexão explicada na Tabela 4, não cita que a *View* faz suas requisições à *View Model* a partir de *Data Bindings* (para o português, ligação de dados), onde estes se ligam às propriedades que estão na *View Model* para recuperar um valor. Além do mais, ao utilizar deste recurso, o programador não irá precisar escrever código na classe anexada à *View*, que para o inglês, chame-se *code-behind*[7].

A partir deste adicional as funções citadas na tabela, a Figura 2 mostra o uso do recurso de *Data Binding*.

[7] *code-behind*, para o português, código por trás. É um termo associado ao código criado pelo processador XAML.

Figura 2 - Esquema do modelo MVVM.



Fonte: PETZOLD, 2016, p. 492. ***Modificado pelo autor.***

2.4.3. Linguagem C#

A linguagem C# foi criada pelo time de desenvolvedores da Microsoft, liderados por Anders Hejlsberg. Seguindo as mesmas inspirações do .NET Framework, onde as próximas tendências futuras seriam movidas pela tecnologia, o time de desenvolvimento teve muita influência e inspiração nas linguagens de programação C, C++ e Java, em relação a popularidade delas. Um dos propósitos da linguagem, era que ela teria que ser destinada tanto para sistemas hospedados e embutidos, independente se fosse um sistema grande e sofisticado ou um sistema comum e pequeno. Sua primeira versão (1.0) foi lançada em janeiro de 2002, em conjunto com o lançamento do Visual Studio 2002, o que atraiu a comunidade de desenvolvedores (NOVÁK et al, 2010).

C# é atualmente uma linguagem de programação orientada a objeto e fortemente “tipada”, sendo que suas variáveis e / ou objetos tem um tipo bem definido e que precisa ser informado no momento da sua declaração e que está em sua versão 7.1, lançada em setembro do ano de 2017 juntamente com o Visual Studio 2017 versão 15.3 (MICROSOFT, 2017). Contêm muitas características que lhe torna uma linguagem simples, muito poderosa e produtiva. Algumas delas são: a facilidade no entendimento de sua sintaxe, tornando o dia a dia do programador mais simples; programação assíncrona; expressões *LINQ* (*Language Integrated Query* - Consulta Integrada à Linguagem); expressões lambda[8] etc.

Em conjunto com a programação utilizando a linguagem C#, o projeto foi desenvolvido no ambiente muito bem conhecido, o Visual Studio 2017 Enterprise. Versão licenciada através do convênio que a Faculdade de Tecnologia de Americana possui com a Microsoft.

[8] Expressão lambda, em C#, são funções anônimas que são usadas para criar funções locais que podem ou não retornar valores e passar argumentos. Saiba mais em: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions>.

A Figura 3 mostra o ambiente de desenvolvimento utilizado no projeto.

Figura 3 - IDE utilizada no desenvolvimento do projeto.

Fonte: Elaborado pelo autor.

A IDE[9] possui diversas ferramentas que podem ser instaladas, facilitando a vida do programador. Umas dessas ferramentas utilizadas foram a de controle de versão e de gerenciamento do código fonte, o Git[10] e Visual Studio Team Services[11].

2.4.4. ASP.NET Core Web API

> É uma interface normalmente especificada como um conjunto de operações, que permite acesso a uma funcionalidade da aplicação. Isso significa que esta funcionalidade pode ser chamada diretamente por outros programas e não apenas acessada através da Interface de usuário. (SOMMERVILLE, 2011).

Foi também desenvolvido uma API Web utilizando o ASP.NET Core, que “é uma estrutura de software livre, de multiplataforma e alto desempenho para a criação de aplicativos modernos conectados à Internet e baseados em nuvem” (MICROSOFT, 2017), com o padrão MVC.

---

[9] IDE, do inglês, *Integrated Development Environment*, para o português, Ambiente de desenvolvimento integrado.

[10] Git é um sistema de controle de versão distribuído gratuito que tem o objetivo de registrar as mudanças feitas em um ou em um conjunto de arquivos (GIT, 2017).

[11] Visual Studio Team Services é uma plataforma que oferece suporte a times de desenvolvedores. Contendo várias ferramentas que ajudam no desenvolvimento, nos testes de software, na colaboração, acompanhamento do trabalho, compartilhamento de código etc. (MICROSOFT, 2017).

Antes de abordar o que a API faz, deve se entender o termo REST, pois esta API utiliza deste estilo de desenvolvimento. REST, do inglês, *Representational State Transfer*, para o português, Transferência de Estado Representacional, segundo a definição de SOMMERVILLE (2011), REST é um estilo de desenvolvimento com base na transferência de uma representação de documentos entre cliente/servidor que utiliza o protocolo de transferência de dados, ou HTTP[12]. Sendo que essas transferências ou interações tem como base os recursos em HTTP POST (cria algum estado no servidor), GET (lê o estado do documento), PUT (atualiza o estado do documento) e DELETE (exclui o estado do servidor) (SOMMERVILLE, 2011).

E por fim, o modelo REST se base em recursos identificáveis, sendo as URIs[13], que é o meio que diferencia os recursos um dos outros. São compostas pelo "protocolo/ host/ caminho", sendo que por elas que os recursos são manipulados.

Esta API é de grande importância para o projeto, pois ela gerencia a parte de Estabelecimentos e Horários disponíveis. No quesito gerenciar os Estabelecimentos, a API controla a parte de retornar (GET) todos os estabelecimentos cadastrados ou algum específico, em formato JSON[14]. Já a parte de gerencia de Horários, a API controla os recursos REST comuns (GET, POST, UPDATE e DELETE).

2.4.5. Microsoft Azure

Segundo a Microsoft, "computação em nuvem é o fornecimento de serviços de computação" através da internet (a "nuvem"). Os serviços oferecidos são: servidores, armazenamento, banco de dados, rede, software, análise etc. As empresas fornecedoras destes serviços são chamadas de "provedores de nuvem" e costumam cobrar por este serviço com base no uso. Esses serviços acabam gerando benefícios a partir dos serviços da nuvem, sendo alguns deles: velocidade, escalabilidade (fornece a quantidade correta de recursos de TI), produtividade, desempenho, confiabilidade.

---

[12] HTTP, do inglês, *Hypertext Transfer Protocol*, para o português, Protocolo de Transferência de Hipertexto.
[13] URI, do inglês, *Uniform Resource Identifiers*, para o português, Indentificador Uniforme de Recursos.
[14] JSON, do inglês, *JavaScript Object Notation*, em português, Notação de Objetos JavaScript, é uma formatação para troca de dados. Saiba mais em: <http://www.json.org/json-pt.html>.

Segundo a Microsoft, os serviços de nuvem se dividem em três amplas categorias, sendo eles:

IaaS (*Infrastructure as a Service*, para o português, Infraestrutura como Serviço): Além de ser a base da nuvem, é uma infraestrutura de computação instantânea gerenciada pela internet, ou seja, um ambiente de computador virtualizado. Constitui de bens tangíveis, como servidores, equipamentos de rede e de software, discos de armazenamento. É um serviço onde ajuda o cliente a evitar gastos em configurar e gerenciar sua infraestrutura, como exemplo, datacenters. Na IaaS o cliente obterá controle de uma máquina virtual localizada na nuvem, nesta, o cliente terá o dever de gerenciar toda a parte da infraestrutura da máquina, por exemplo, a segurança/firewalls de rede, atualizações do sistema operacional. Já a parte de hardware, o provedor do serviço tem a responsabilidade de gerenciá-la.

PaaS (*Platform as a Service*, para o português, Plataforma como Serviço): Como a IaaS, a PaaS também possui os recursos de infraestrutura, mas são acrescentadas algumas ferramentas para o desenvolvimento de aplicativos, gerenciamento de banco de dados etc. Esta plataforma é ideal para desenvolvedores, pois não há necessidade de se preocupar com a infraestrutura, a plataforma terá a responsabilidade de gerenciá-la. O dever do cliente será apenas gerenciar o seu aplicativo e os dados que irão trafegar por sua aplicação.

SaaS (*Software as a Service*, para o português, Software como Serviço): Diferente dos dois outros modelos, a SaaS é um tipo de serviço que fornece ao cliente toda a parte de infraestrutura até o próprio sistema, sendo que todo o ambiente é gerenciado pelo provedor do serviço. O cliente não precisa se preocupar com nada. Exemplos de SaaS mais conhecidos nos dias de hoje são os serviços de e-mail baseados na Web, como Outlook, Hotmail e Yahoo! Mail.

Durante o trabalho, o autor pensou em como tornar o seu desenvolvimento mais ágil e prático, chegando à conclusão que *cloud computing* (computação em nuvem) é essencial quando se trata de programação Web e móvel. Existem diversas plataformas que oferecem diversos serviços relacionados a *cloud computing*. A plataforma provedora de nuvem escolhida pertence a empresa Microsoft e se chama Microsoft Azure, e a razão pelo autor ter a escolhido foi a familiaridade que ele já

possui em desenvolver projetos utilizando esta plataforma. Além de possuir recursos que são fácil e rapidamente configuráveis.

Os serviços utilizados na nuvem da Microsoft, são os listados abaixo:

- **Banco de dados SQL do Azure** é um serviço de banco de dados relacional inteligente de nuvem (MICROSOFT, 2017). O projeto possui todo o banco de dados integrado na nuvem, podendo ser utilizado por qualquer aplicativo de API, móvel ou de Web.
- **Serviço de aplicativo** é uma plataforma dentro do Azure que gerencia a parte de hospedagem de aplicativos de API, móveis e de Web (MICROSOFT, 2017).
    - **Aplicativo Web** consiste na hospedagem em si de sites e de aplicativos Web, que no final se torna uma oferta de PaaS (MICROSOFT, 2017). O projeto usufrui deste serviço com um aplicativo Web (API desenvolvida).
    - **Aplicativo Móvel** é um serviço onde se oferece serviços avançados para cenários Web, móveis e de integração que é altamente dimensionável e global. Com essas características acaba se tornando um PaaS (MICROSOFT, 2017). O projeto utiliza desse serviço, hospedando dois aplicativos móveis e aproveita da integração com a autenticação com o Facebook e sincronização de dados utilizando o Tabelas fáceis.
        - **Tabelas fáceis ou *Easy Tables*** consiste em um serviço que tem o objetivo de ser um *back-end* que a própria plataforma cria para o gerenciamento fácil e rápido do banco de dados que está armazenando as informações do usuário e de agendamento do sistema.

Para saber mais sobre as configurações do Azure, confira o Apêndice A.

2.4.6. Facebook Graph API

> A Graph API é a principal forma de incluir e excluir dados na plataforma do Facebook. É uma API baseada em HTTP de nível inferior que você pode usar para consultar dados de forma programada, publicar novas histórias, gerenciar anúncios, carregar fotos e realizar várias outras tarefas que um aplicativo pode implementar (FACEBOOK, 2017).

Esta API do Facebook foi de muita utilidade no desenvolvimento do projeto, pois de maneira fácil, consegue-se os dados de qualquer cliente. Mas para conseguir acesso à essas informações o usuário necessita permitir o acesso, sem essa autorização, não há como.

Um exemplo de um pedido à Graph API utilizado no projeto, pode ser conferido na Figura 4:

Figura 4 - URI utilizada para pedido da Graph API.

```
var requestUrl =
    "https://graph.facebook.com/v2.10/me?fields=id,name,email,birthday,gender,picture.height(720){url},cover&access_token=" + userToken;
```

Fonte: Elaborado pelo autor.

2.4.7. Google Maps API

O Google Maps API é recurso que o Google fornece aos desenvolvedores que possibilita adicionar e personalizar mapas baseados nos dados do Google Maps para aplicativos de dispositivos móveis (GOOGLE DEVELOPERS, 2017). Foi utilizado este recurso para facilitar a visualização de estabelecimentos cadastrados de exemplo para a demonstração do sistema.

2.5. Modelagem

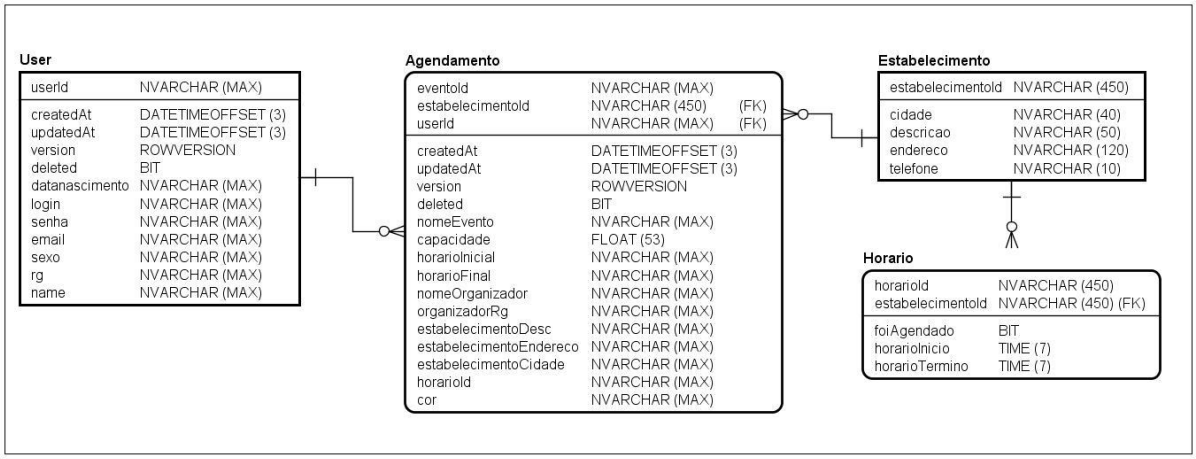
Na fase de modelagem é feita a documentação do aplicativo, onde, de forma padronizada, a compreensão será feita a partir de diagramas.

Sendo que a documentação irá utilizar-se da Linguagem de Modelagem Unificada, ou do inglês, *Unified Modeling Language* (UML), muito utilizada para modelar *softwares* orientados a objetos. Para se obter uma visão diferenciada do sistema, o projeto utilizará dois tipos de diagramas da UML, o diagrama de casos de uso, diagrama de classes e o modelo de entidade e relacionamento do banco de dados.

2.5.1. Diagrama Entidade Relacionamento

O DER foi desenvolvido para facilitar a compreensão da estrutura do banco de dados do projeto, fazendo com que o programador obtenha uma maior facilidade de visualizar e entender como o banco deverá se comportar, além de fornecer as relações entre as entidades e seus atributos.

Figura 5 - DER do projeto de banco de dados do sistema.



Fonte: Elaborado pelo autor.

Ao visualizar a Figura 5, o leitor compreende que a entidade *User* (usuário, para o português) pode agendar um ou no máximo vários estabelecimentos. O relacionamento se torna 1 para N quando a entidade Estabelecimento pode ser agendada por no mínimo um ou no máximo por vários usuários, tornando assim um relacionamento de muitos para muitos e que, obrigatoriamente cria uma outra tabela que contém as chaves primárias das tabelas *User* e Estabelecimento como chave estrangeira, além de outros atributos.

Outro relacionamento perceptível na imagem, é entre a entidade Estabelecimento, onde esta possui no mínimo um ou muitos horários cadastrados.

2.5.2. Casos de Uso

Os diagramas de casos de uso, descrevem, através de uma linguagem informal, as funcionalidades do sistema, o seu comportamento geral, através da perspectiva do usuário. Tendo como principais itens, os atores (representados por bonecos palito), as funcionalidades do sistema (representadas por elipses contendo a descrição desta) e as relações (representadas por linhas) (BERNARDO, 2017).

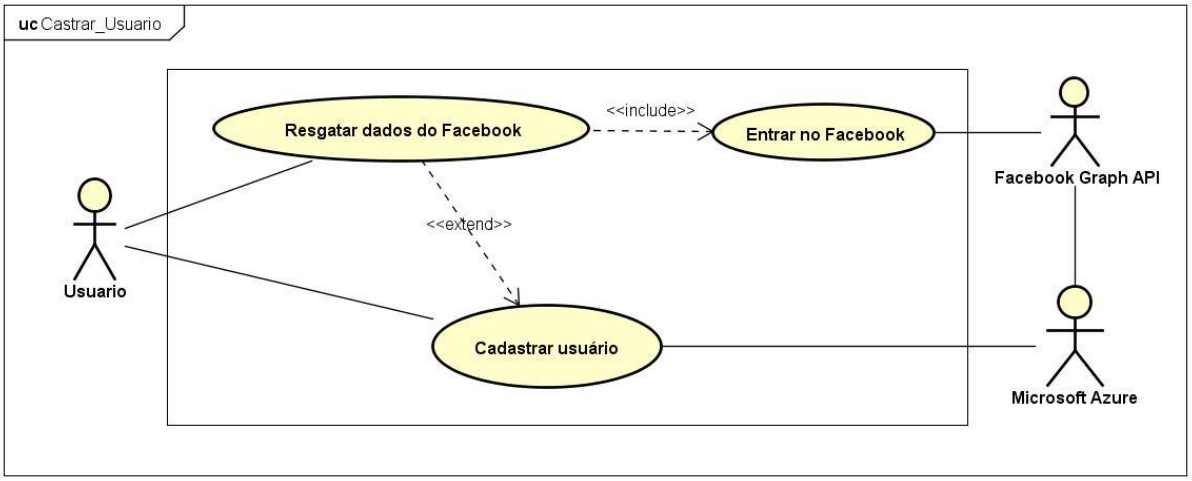
A seguir, será definido quem são os atores e algumas de suas funções.

- **Usuário:** é o ator que representa o cliente que está utilizando do aplicativo. Este ator, pode, por exemplo, cadastrar-se no sistema, editar seu perfil, agendar um local etc.

- **Facebook Graph API:** representa a API que interage com o sistema, e que fornece informações do usuário, que concede autorização através do login com sua conta do Facebook.
- **Microsoft Azure:** este ator representa a nuvem da Microsoft. Nele é onde está localizado o banco de dados de toda a aplicação.
- **API Desenvolvida:** representa a API que faz operações REST (HTTP DELETE, GET, POST e UPDATE) e que se comunica com o banco de dados localizado na nuvem da Microsoft. Possuindo apenas informações sobre os estabelecimentos e seus horários.

A Figura 6 apresenta o caso de uso do cadastro de usuário no sistema.

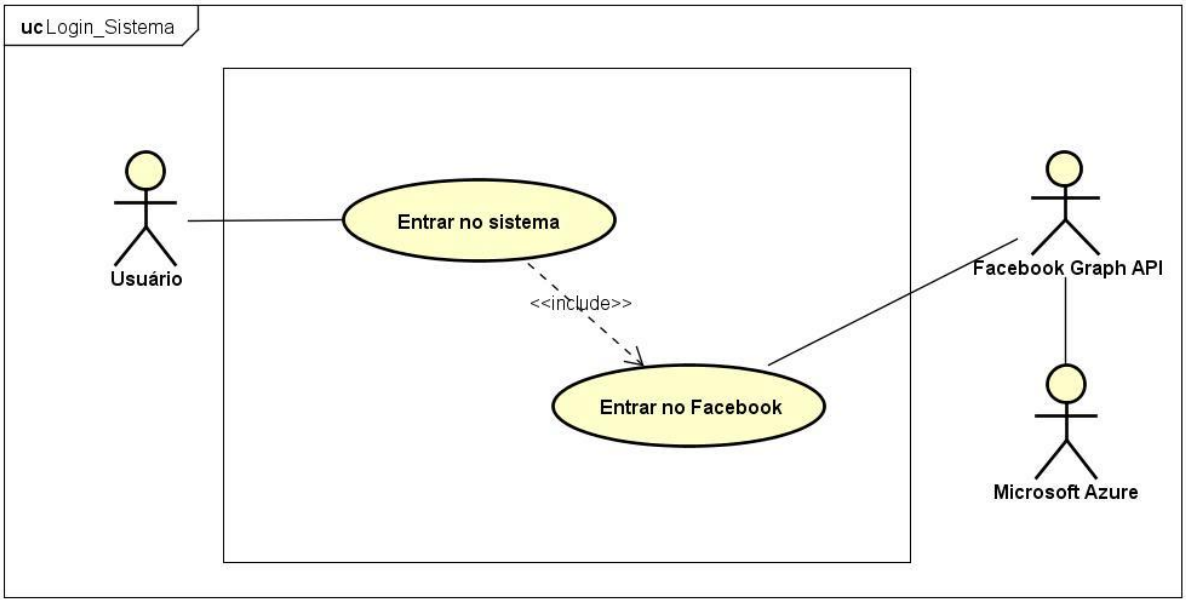
Figura 6 - Diagrama de Caso de Uso de Cadastro de Usuário.



Fonte: Elaborado pelo autor.

A Figura 7 representa apresenta o caso de uso para a entrada do usuário no aplicativo.

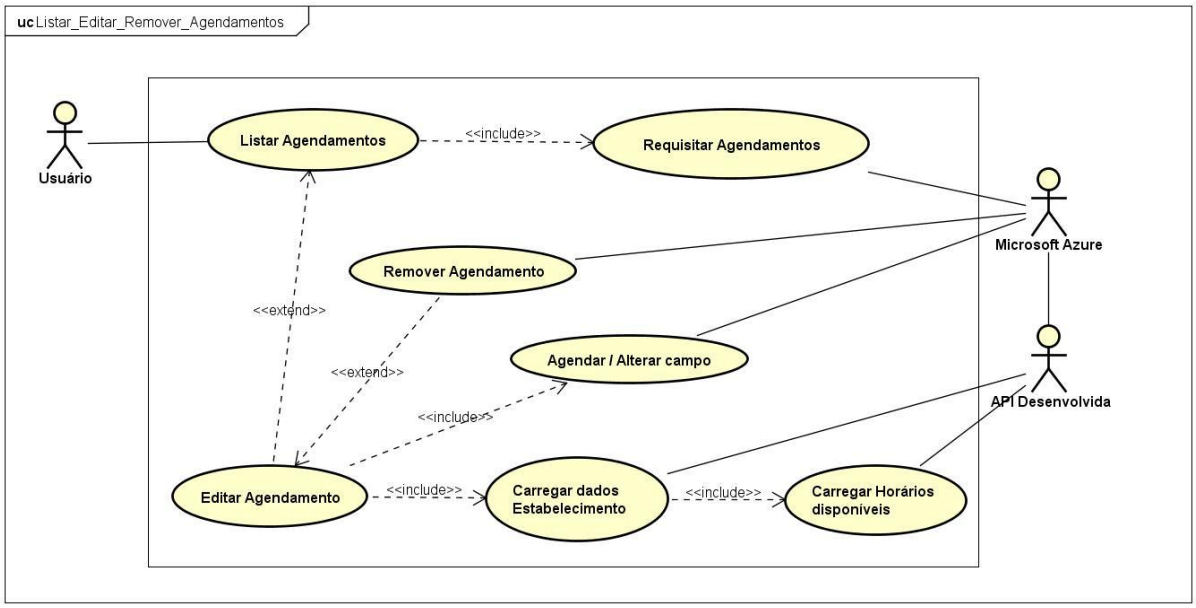
Figura 7 - Diagrama de Caso de Uso Login de Usuário.



Fonte: Elaborado pelo autor.

A Figura 8 apresenta o caso de uso que representa as ações de listar, editar e remover o agendamento.

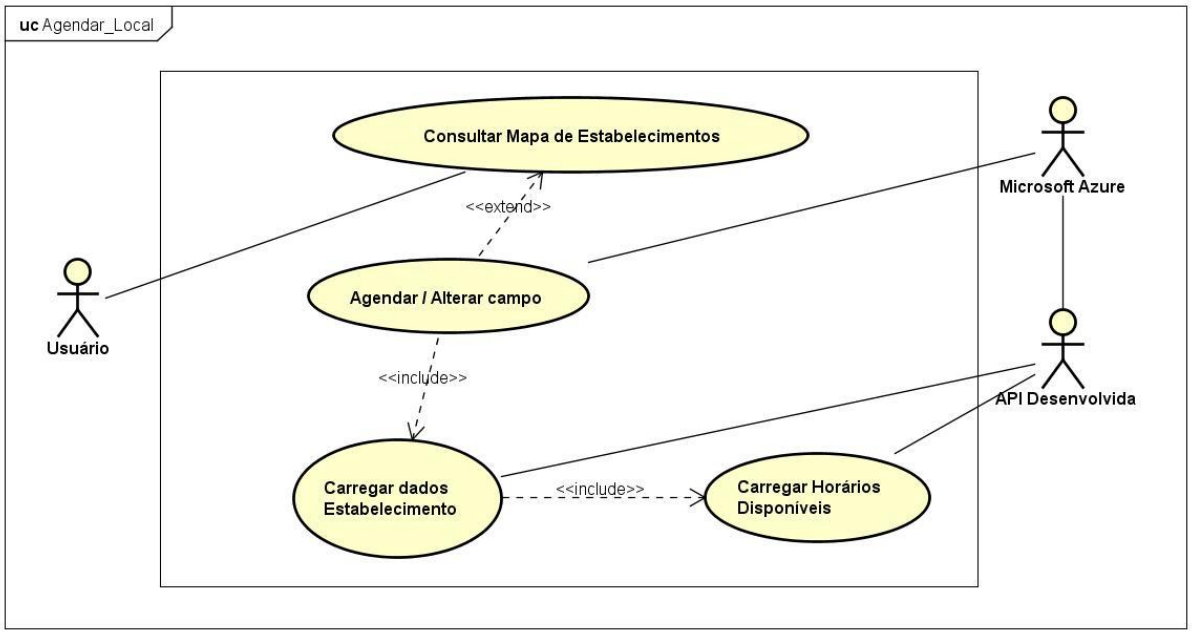
Figura 8 - Diagrama de Caso de Uso de Listar / Editar / Remover agendamento.



Fonte: Elaborado pelo autor.

A Figura 9 se refere ao caso de uso para agendar o campo de futebol.

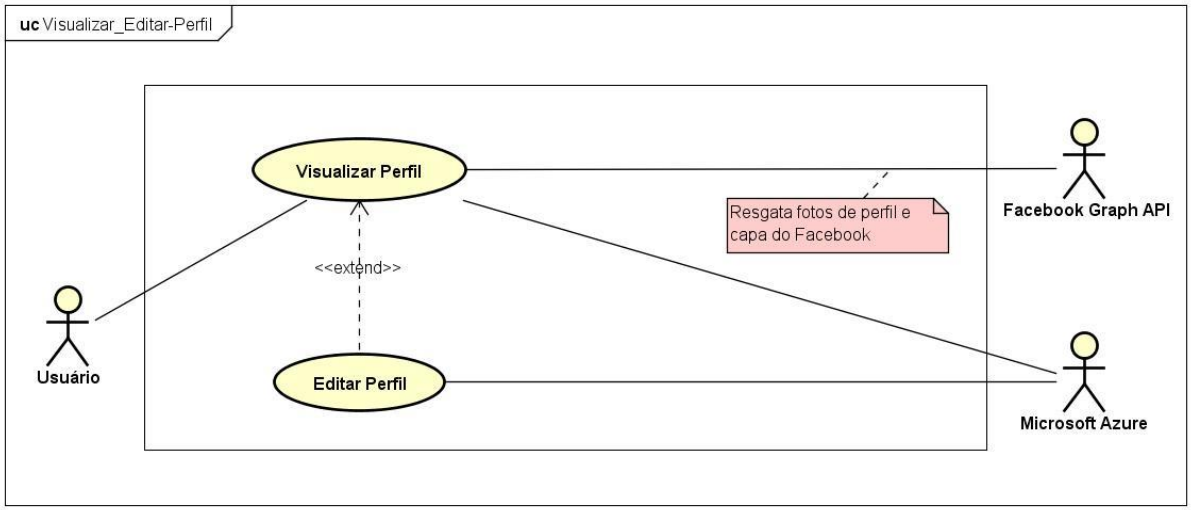
Figura 9 - Diagrama de Caso de Uso Agendar Local.



Fonte: Elaborado pelo autor.

A Figura 10 apresenta o caso de uso de visualização e edição de perfil do usuário.

Figura 10 - Diagrama de Caso de Uso Visualizar / Editar Perfil.



Fonte: Elaborado pelo autor.

2.5.3. Documentação dos Casos de Uso

As funcionalidades dos casos de uso já apresentados serão descritas nos quadros a seguir, onde antes de cada tabela, será abordado uma explicação mais detalhada do caso de uso em questão.

Para que o login do usuário seja efetuado, o sistema valida as informações fornecidas pelo usuário nos campos e se forem válidas ele prossegue para a autenticação da conta do Facebook, onde a Graph API retornará com as informações do usuário e seu *Access Token*[15], onde este será enviado para o serviço utilizado do Microsoft Azure e o autenticará, possibilitando sua entrada no aplicativo. A Tabela 5 detalha o passo a passo da execução deste caso de uso.

Tabela 5 - Documentação do Caso de Uso Entrar no Sistema.

Nome do Caso de Uso	Entrar no sistema
Atores Envolvidos	Usuário, Facebook Graph API, Microsoft Azure.
Objetivo	Este caso de uso descreve os passos do login de um usuário no sistema.
Prioridade de Desenvolvimento	Essencial
Ações do Ator	**Ações do Sistema**
1. O usuário clica no botão entrar.	
	2. Verifica se o usuário existe e se existir abre a tela de autenticação do Facebook e retorna com as informações.
	3. Ao retornar as informações do usuário, o Microsoft Azure autentica o usuário.
	4. Após o usuário ser autenticado, o sistema redireciona para a página inicial do aplicativo.

[15] *Access Token*, para o português, *Token* de Acesso, é uma cadeia de caracteres que identifica um usuário. Ele concede o acesso temporário às APIs do Facebook, possibilitando o acesso às informações deste, como nome, fotos, data de nascimento etc.

Validações	1. Para que o login seja efetuado, o usuário deve inserir o nome de usuário e senha corretos.
	2. Para efetuar o login, o usuário deve inserir uma conta já existente.
	3. Para efetuar login, o usuário deverá entrar com sua conta do Facebook.
	4. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

O caso de uso “Entrar no Facebook” que é detalhado na Tabela 6, tem o propósito principal de resgatar o *Token* de Acesso do usuário, para que mais tarde o sistema possa resgatar os dados do usuário através da Graph API do Facebook. Para que isso aconteça, o sistema ativa um serviço que irá invocar uma tela de autenticação do Facebook, onde o usuário entrará com e-mail e senha de seu Facebook. Em seguida será exibido uma tela de conceder permissões, se autorizado o acesso aos seus dados, o serviço retornará um ID de usuário único e o mandará para autenticar no serviço do Azure.

Tabela 6 - Documentação do Caso de Uso Entrar no Facebook.

Nome do Caso de Uso	Entrar no Facebook
Atores Envolvidos	Facebook Graph API, Microsoft Azure, Usuário.
Objetivo	Este caso de uso descreve os passos de login com o Facebook.
Prioridade de Desenvolvimento	Essencial
Ações do Ator	**Ações do Sistema**
	1. Abre uma tela de autenticação do Facebook.
2. Entra com e-mail e senha.	
3. Concede autorização para acessar seus dados.	

	4. Valida informações de login e retorna com seu *Token* de Acesso.
	5. Manda para o *Token* de Acesso para o Microsoft Azure autenticar o usuário.
	6. Autentica o usuário e manda um *token* de usuário.
Validações	1. Usuário precisa aceitar o acesso aos seus dados.
	2. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

O aplicativo oferece ao usuário a opção de entrar com a conta de seu Facebook, possibilitando o resgate de suas informações e tornando o preenchimento do formulário de cadastro menos cansativo. Se ele escolher a opção de resgatar suas informações, o aplicativo irá aplicar as mesmas funções de login de sistema (ver Tabela 5), a diferença é que, ao fechar a janela de autenticação do Facebook, alguns campos do formulário estarão preenchidos com as informações obtidas através do login com o Facebook. Ao clicar no botão “Cadastrar”, o sistema irá validar primeiro se existe alguma informação em branco, se existir, o aplicativo exibirá um alerta. Após a validação dos campos, o sistema irá mandar para a API do Azure o campo de login do usuário, retornando se existe ou não um usuário com o mesmo login. Se não existir, o sistema manda para a API do Azure as informações do usuário, onde ela irá inserir no banco de dados esses dados, cadastrando o usuário. A Tabela 7 detalha passo a passo do caso de uso “Cadastrar Usuário”.

Tabela 7 - Documentação do Caso de Uso Cadastrar Usuário.

Nome do Caso de Uso	Cadastrar Usuário
Atores Envolvidos	Usuário, Facebook Graph API, Microsoft Azure.
Objetivo	Este caso de uso descreve os passos do cadastro de um usuário no sistema.
Prioridade de Desenvolvimento	Essencial

Ações do Ator	Ações do Sistema
1. O usuário clica no texto “Cadastre aqui”.	
	2. Abre uma nova tela de cadastro.
3. Usuário clica no botão “Entrar com Facebook”.	
	4. Uma tela de autenticação do Facebook é aberta.
5. Usuário digita seu e-mail e senha do Facebook.	
	6. A Graph API autentica o usuário e retorna com suas informações.
	7. Sistema preenche os campos de texto com as informações retornadas da Graph API.
8. Usuário edita ou preenche os campos com suas informações.	
	9. O sistema valida se existe algum campo em branco.
	10. Checa se existe no banco de dados no Azure, um Usuário com o mesmo nome de login. Se não existir, insere no banco de dados, cadastrando o Usuário.
Validações	1. Para que o cadastro seja efetuado, o usuário deve inserir todos os campos requeridos.
	2. Para efetuar o cadastro, o usuário deve inserir um nome de login não existente.
	3. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

Como já foi dito na explicação do caso de uso “Cadastrar Usuário”, o usuário pode clicar no botão de entrar com o Facebook e resgatar suas informações, onde o sistema preencherá os campos com os dados recebidos da Graph API. Este

procedimento consiste em receber o *Token* de Acesso do usuário, após o login (ver Tabela 6 para o procedimento de login), em seguida enviá-lo para a Graph API, junto com os campos pré-definidos pelo desenvolvedor (como data de nascimento, sexo, nome completo etc.), em um formato de URI. Em seguida, a API do Facebook devolve uma URI contendo as informações do usuário. O sistema separa esses dados e os envia para os campos de texto. A Tabela 8 detalha o caso de uso “Resgatar dados do Facebook”.

Tabela 8 - Documentação do Caso de Uso Cadastrar Usuário.

Nome do Caso de Uso	Resgatar dados do Facebook
Atores Envolvidos	Usuário, Facebook Graph API, Microsoft Azure.
Objetivo	Este caso de uso descreve os passos de resgate de informações do usuário.
Prioridade de Desenvolvimento	Essencial
Ações do Ator	**Ações do Sistema**
	1. Recebe o *Token* de Acesso do usuário e o envia junto com os campos requeridos em um pedido à API em formato de URI.
	2. Recebe a resposta da API em formato de URI contendo os valores. Separa esses valores e os envia para os campos de texto.
Validações	1. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

Após procedimento de entrada no sistema, o aplicativo irá executar automaticamente o caso de uso “Listar Agendamentos” ou o usuário pode ativá-lo ao executar o comando “puxe para atualizar”. Que tem como objetivo chamar o caso de uso “Requisitar agendamentos” (ver Tabela 10), que retorna todos os agendamentos. Ao receber estes, o sistema irá filtrar aqueles que possuem o identificador do usuário e ao final, exibi-los em tela para o usuário. Esta ação de “Listar Agendamentos” tem a mesma finalidade de atualizar os itens que são exibidos na tela.

A Tabela 9 apresenta o passo a passo que o sistema utiliza para o caso de uso “Listar Agendamentos”.

Tabela 9 - Documentação do Caso de Uso Listar Agendamentos.

Nome do Caso de Uso	Listar Agendamentos
Atores Envolvidos	Usuário, Microsoft Azure
Objetivo	Este caso de uso descreve os passos da filtragem e exibição dos agendamentos ao cliente.
Prioridade de Desenvolvimento	Essencial
Ações do Ator	**Ações do Sistema**
1. Usuário ativa o comando “puxe para atualizar”.	
	2. Envia uma requisição de agendamento para o banco de dados do Azure que então recebe a resposta contendo todos os agendamentos.
	3. Filtra os agendamentos a partir do identificador do usuário corrente e os exibe em tela.
	4. Uma tela de autenticação do Facebook é aberta.
Validações	1. Identificar os agendamentos do usuário “logado” no sistema a partir de seu identificador.
	2. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

Como já foi comentado na breve explicação da Tabela 9, o caso de uso “Listar Agendamentos” ativa o “Requisitar Agendamentos”. Neste, é realizado a operação de pedido à API do Azure e que ao receber este pedido, buscará no banco de dados deste mesmo provedor, todos os agendamentos e ao retornar com os valores, manda para o caso de uso “Listar Agendamentos”.

A Tabela 10 mostra com mais detalhes este processo.

Tabela 10 - Documentação do Caso de Uso Requisitar Agendamentos.

Nome do Caso de Uso	Requisitar Agendamentos
Atores Envolvidos	Microsoft Azure, Sistema.
Objetivo	Este caso de uso descreve os passos da requisição à API do Azure.
Prioridade de Desenvolvimento	Essencial
Ações do Ator	**Ações do Sistema**
	1. Manda uma requisição à API do Microsoft Azure com identificador do usuário “logado”, com isso busca no banco de dados todos os agendamentos e os envia de volta, como reposta.
	2. Recebe a resposta contendo todos os agendamentos e os entrega para o caso de uso “Listar Agendamento”.
Validações	1. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

Com os agendamentos do usuário carregados e listados em tela, o cliente ao selecionar um item da lista, pode editar ou apenas visualizar os detalhes do agendamento. Este caso de uso é de grande importância, pois, a partir dele o usuário tem as opções de excluir e/ou alterar um agendamento (campo *society*).

A Tabela 11 apresenta com mais detalhes o passo a passo do caso de uso Editar Agendamento.

Tabela 11 - Documentação do Caso de Uso Editar Agendamento.

Nome do Caso de Uso	Editar Agendamentos
Atores Envolvidos	Usuário, Microsoft Azure, Sistema.
Objetivo	Este caso de uso descreve os passos da edição de um agendamento.
Prioridade de Desenvolvimento	Essencial

Ações do Ator	Ações do Sistema
1. Usuário clica em um item da lista de agendamentos na tela inicial.	
	2. Ao perceber que o item foi clicado, o sistema recebe o identificador do agendamento selecionado e manda um pedido para a API desenvolvida para que selecione os dados do estabelecimento e os horários disponíveis deste (casos de uso “Carregar dados Estabelecimento” e “Carregar Horários Disponíveis”). Logo então, o sistema recebe como resposta as informações do agendamento e exibe nos campos da tela. Ao final destas ações, o usuário está disponível para remover um agendamento ou alterá-lo.
Validações	1. Identificador do agendamento não pode ser nulo.
	2. Usuário deve clicar nos botões para que sistema possa realizar alguma ação.
	3. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

Ao editar um agendamento o usuário pode optar por remover este, efetuando a ação de desmarcar o local e deixar o horário livre para que algum outro cliente possa vir à agenda-lo. Esta ação consiste em remover do banco de dados este agendamento (que está no Microsoft Azure) e também modificar o campo que estabelece se o horário está ou não agendado para falso na API desenvolvida.

A Tabela 12 apresenta o passo a passo do caso de uso Remover Agendamento.

Tabela 12 - Documentação do Caso de Uso Remover Agendamento.

Nome do Caso de Uso	Remover Agendamento
Atores Envolvidos	Usuário, Microsoft Azure, API Desenvolvida.
Objetivo	Este caso de uso descreve os passos da remoção de um agendamento.
Prioridade de Desenvolvimento	Essencial

Ações do Ator	Ações do Sistema
1. Usuário clica no botão de exclusão.	
	2. A entidade “agendamento” é repassada para o serviço do Azure que irá excluir do banco de dados os dados do agendamento.
	3. Ao mesmo tempo que a etapa 2 acontece, o serviço manda uma requisição do tipo HTTP PUT para a API desenvolvida, que irá atualizar o valor do campo “FoiAgendado” para falso, tornando o horário disponível para futuros agendamentos.
	4. Por fim, o sistema fecha a tela automaticamente e levando o usuário para a tela inicial mostrando os cadastros atualizados do usuário “logado”.
Validações	1. Usuário deve clicar no botão de exclusão para que sistema possa realizar a determinada ação.
	2. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

Para que os dados do ou dos estabelecimentos sejam carregados por qualquer método da aplicação, deve-se fazer uma requisição HTTP GET para a API desenvolvida que como resposta irá devolver os dados. No corpo da requisição pode conter o identificador do estabelecimento, onde a API irá devolver apenas os dados referentes a este identificador ou não conter nada, fazendo com que retorno todos os estabelecimentos cadastrados no banco de dados.

Como o sistema não recupera os dados de todos os estabelecimentos cadastrados no banco de dados, não é necessário a apresentação do passo a passo deste tipo de requisição. A Tabela 13 apresenta o passo a passo da execução do caso de uso Carregar dados Estabelecimento que traz apenas os dados de um único estabelecimento.

Tabela 13 - Documentação do Caso de Uso Carregar dados Estabelecimento.

Nome do Caso de Uso	Carregar dados Estabelecimento

Atores Envolvidos	Microsoft Azure, API Desenvolvida.
Objetivo	Este caso de uso descreve os passos da requisição dos dados do estabelecimento cadastrado no banco de dados localizado na nuvem.
Prioridade de Desenvolvimento	Alta
Ações do Ator	**Ações do Sistema**
1. Usuário seleciona um item da lista de agendamentos na tela inicial.	
	2. Ao perceber que o item foi clicado, o sistema recebe o identificador do agendamento selecionado e manda uma requisição HTTP GET (contendo no corpo do pedido o identificador do estabelecimento) para a API desenvolvida, para que selecione os dados do estabelecimento daquele identificador. Logo então, o sistema recebe como resposta as informações do estabelecimento e exibe nos campos da tela.
Validações	1. Usuário deve clicar em algum item da lista de agendamentos.
	2. A requisição deve conter o identificador do estabelecimento.
	3. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

Ao mesmo tempo que é carregado os dados do determinado estabelecimento é também dos horários disponíveis. O sistema ativa um serviço que irá mandar uma requisição HTTP GET para API contendo o identificador do estabelecimento. Retornando então, os horários daquele estabelecimento. O sistema então filtra os horários disponíveis para agendamento e os mostra em conjunto com o horário agendado pelo usuário.

A Tabela 14 apresenta detalhadamente o caso de uso Carregar Horários disponíveis.

Tabela 14 - Documentação do Caso de Uso Carregar Horários disponíveis.

Nome do Caso de Uso	Carregar Horários disponíveis

Atores Envolvidos	Microsoft Azure, API Desenvolvida.
Objetivo	Este caso de uso descreve os passos da requisição dos dados dos horários cadastrados no banco de dados.
Prioridade de Desenvolvimento	Alta
Ações do Ator	**Ações do Sistema**
1. Usuário seleciona um item da lista de agendamentos na tela inicial.	
	2. Ao perceber que o item foi clicado, o sistema recebe o identificador do agendamento selecionado e manda uma requisição HTTP GET (contendo no corpo do pedido o identificador do estabelecimento) para a API desenvolvida, para que selecione todos os horários cadastrados para aquele estabelecimento.
	3. Com os dados em “mãos”, o sistema filtra apenas os horários disponíveis para agendamento e adiciona também a esta lista, o horário agendado pelo usuário, que é filtrado pelo seu identificador armazenado nos dados do agendamento. Após esse processo de filtragem, o sistema exibe os horários para o usuário.
Validações	1. Usuário deve clicar em algum item da lista de agendamentos.
	2. A requisição deve conter o identificador do estabelecimento.
	3. O agendamento deve conter o identificador do horário agendado pelo usuário.
	4. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

O caso de uso de agendar ou alterar o local, ativa a ação de inserção ou alteração do agendamento no banco de dados. Se caso a propriedade do valor esteja nula, o sistema ativa a ação de inserção no banco de dados, caso a propriedade já contém um valor, quer dizer que a ação será de atualização dos dados. A função de inserção envia uma requisição do tipo HTTP POST contendo em seu corpo, os dados

do novo objeto sem o identificador, já a atualização, manda o tipo HTTP PUT contendo o valor do identificador, para o Microsoft Azure executar.

A Tabela 15 apresenta o passo a passo do caso de uso Agendar / Alterar campo detalhadamente.

Tabela 15 - Documentação do Caso de Uso Agendar / Alterar campo.

Nome do Caso de Uso	Agendar / Alterar campo
Atores Envolvidos	Microsoft Azure, API Desenvolvida.
Objetivo	Este caso de uso descreve os passos das funções de inserção ou atualização do agendamento de um campo *society*.
Prioridade de Desenvolvimento	Essencial
Ações do Ator	**Ações do Sistema**
1. Usuário clica no botão de agendar / alterar.	
	2. Sistema checa se nenhum campo está vazio, se estiver, um alerta é exibido ao usuário. Caso todos os campos estão preenchidos, o sistema passa o objeto agendamento para a classe de serviço, verificando se a propriedade de identificação do objeto está vazia. Se estiver vazia o sistema manda para o Azure uma requisição HTTP POST contendo o valor do objeto e por fim, o objeto é inserido no banco de dados. Se o valor da propriedade não for nulo, o sistema entra na ação de atualização do agendamento, passando o valor do objeto através de uma requisição HTTP PUT, que atualizará os dados do agendamento no banco de dados localizado na nuvem da Microsoft.
	3. Finalizado o processo de atualização ou inserção, o sistema volta para a tela inicial, mostrando os agendamentos novos ou atualizados.
Validações	1. Todos os campos requeridos devem estar preenchidos.
	2. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

O usuário tem a opção de escolher o estabelecimento a onde ele deseja alugar o campo *society*, o aplicativo possui a opção de exibir os estabelecimentos disponíveis através de um mapa que utiliza a API nativa do Android em conjunto com o Google Maps API. Basicamente, consiste em mostrar *pins* (marcadores em formato de uma imagem de futebol) que tem o objetivo de exibir onde estão localizados os estabelecimentos cadastrados e exibir quais estão próximos ao usuário. A localização do usuário é dada através de um componente instalado no projeto, chamado *Xam.Plugin.Geolocator*[16]. Caso o usuário se interesse pelo estabelecimento, ele pode clicar no *pin* do estabelecimento para agendar o campo *society* deste.

Na Tabela 16 é apresentado o passo a passo do caso de uso Consultar Mapa de Estabelecimentos.

Tabela 16 - Documentação do Caso de Uso Consultar Mapa.

Nome do Caso de Uso	Consultar Mapa de Estabelecimentos
Atores Envolvidos	Usuário
Objetivo	Este caso de uso descreve os passos das funções de inserção ou atualização do agendamento de um campo *society*.
Prioridade de Desenvolvimento	Essencial
Ações do Ator	**Ações do Sistema**
1. Usuário clica no botão de mostrar estabelecimentos no mapa.	
	2. O sistema demarca os estabelecimentos através de suas coordenadas estabelecidas via código no mapa. Logo depois, esses itens são decorados por uma imagem e são exibidos no mapa. Se o GPS do aparelho estiver ativado o componente consegue localizar o usuário. Caso esteja ligado, o mapa direciona a visão do cliente para a sua localização atual.
Validações	1. Usuário deve estar com o GPS do aparelho ativado.

[16] *Xam.Plugin.Geolocator*, é um componente gratuito de multiplataforma, desenvolvido por James Montemagno. Este componente, também chamado de *plugin*, que capta a localização do GPS do aparelho celular. Saiba mais em: <https://jamesmontemagno.github.io/GeolocatorPlugin/>.

	2. Usuário deve estar conectado à internet.
	3. Usuário deve dar permissão para o aplicativo.

Fonte: Elaborado pelo autor.

O usuário do aplicativo possui a oportunidade de visualizar seu perfil e que, através da Graph API do Facebook o sistema consegue captar suas fotos de capa e de perfil de usuário do Facebook, além de ver suas informações cadastradas previamente no sistema.

A Tabela 17 apresenta o passo a passo do caso de uso Visualizar Perfil.

Tabela 17 - Documentação do Caso de Uso Visualizar Perfil.

Nome do Caso de Uso	Visualizar Perfil
Atores Envolvidos	Usuário, Graph API, Microsoft Azure.
Objetivo	Este caso de uso descreve os passos da ação de visualizar as informações cadastrados do usuário.
Prioridade de Desenvolvimento	Média
Ações do Ator	**Ações do Sistema**
1. Usuário clica no botão de mostrar perfil.	
	2. O sistema recupera o *Token* de Acesso do usuário e passa para a Graph API através de um pedido HTTP GET que é composto pelo *access token* do usuário e as informações que o sistema quer de resposta, após o processamento da Graph API é retornado as informações. O sistema ativa o serviço do Microsoft Azure, para que ele busque no banco de dados as informações do usuário. Este processo acontece através de uma requisição feita API via HTTP GET, composta pelo identificador do usuário. Por fim, é retornado as informações do cliente e são exibidos para ele os dados recebidos como respostas das duas APIs.
Validações	1. Usuário deve estar conectado à internet.

Fonte: Elaborado pelo autor.

Após visualizar suas informações, o usuário pode editar suas informações de perfil, como, alterar seu nome, seu e-mail, data de nascimento etc. Concluída a alteração, o sistema atualiza no banco de dados, localizado na nuvem, as informações do usuário.

A Tabela 18 apresenta o passo a passo do caso de uso de edição de perfil.

Tabela 18 - Documentação do Caso de Uso Editar Perfil.

Nome do Caso de Uso	Editar Perfil
Atores Envolvidos	Usuário, Microsoft Azure.
Objetivo	Este caso de uso descreve os passos da ação de alterar as informações de perfil do usuário.
Prioridade de Desenvolvimento	Média
Ações do Ator	**Ações do Sistema**
1. Usuário clica no botão de editar perfil.	
	2. O sistema ativa o serviço do Microsoft Azure, para que ele busque no banco de dados as informações do usuário. Este processo acontece através de uma requisição feita API via HTTP GET, composta pelo identificador do usuário. Por fim, é retornado as informações do cliente e são exibidos nas caixas de texto.
3. Usuário altera as informações e clica no botão atualizar.	
	4. O sistema armazena as propriedades que contém as informações das caixas de texto e transforma em uma entidade chamada *User*. Então é ativado o serviço do Microsoft Azure, que toma conta de enviar para a nuvem em uma requisição HTTP PUT contendo o identificador e objeto com as informações do usuário. Logo em seguida é atualizado no banco de dados o usuário decorrente do identificador repassado para a API. Por fim, é exibida uma mensagem ao usuário, descrevendo se o sistema obteve êxito ou falha na operação de atualização de seus dados.

Validações	1. Usuário deve estar conectado à internet.
	2. Usuário não deve deixar nenhum campo vazio.

Fonte: Elaborado pelo autor.

2.5.4. Diagramas de Classe

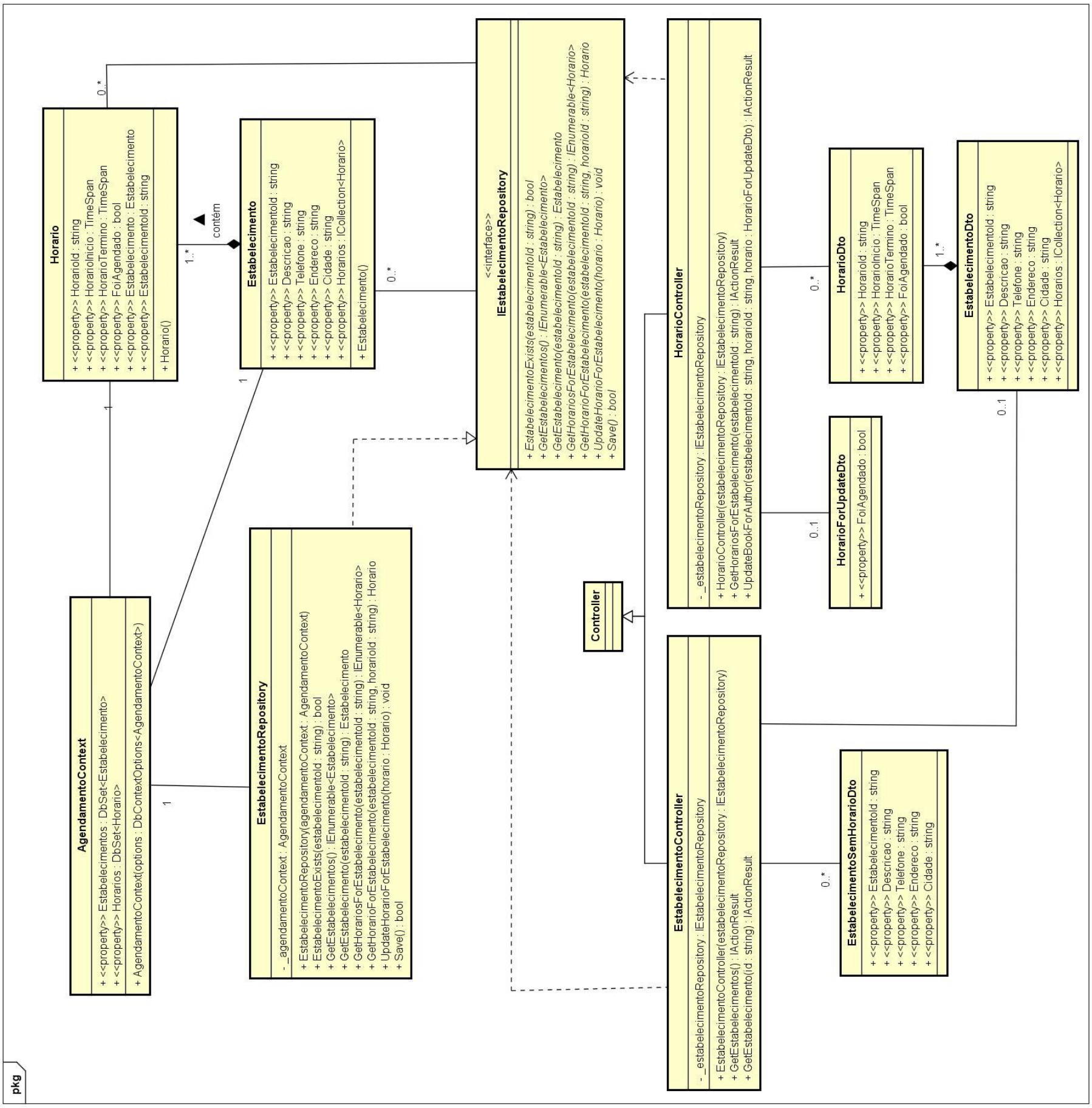
O diagrama de classes sendo o mais utilizado e um dos mais importantes da UML, fornece ao desenvolvedor todas as classes (estrutura de classes) que serão ou já estão empregadas no sistema, os atributos e métodos utilizados que irão atender as funções propostas ao sistema além de estabelecer as relações e como as classes trocam as informações (GUEDES, 2009).

Nos subcapítulos 2.5.4.1 e 2.5.4.2 serão apresentados os diagramas de classe da API desenvolvida e do sistema e na Seção Desenvolvimento, será abordado alguns dos principais métodos utilizados nestas classes.

2.5.4.1. Diagrama de Classe da API

Na Figura 11 é apresentado o diagrama de classes da API desenvolvida para o apoio do projeto.

Figura 11 - Diagrama de Classe da API do projeto.



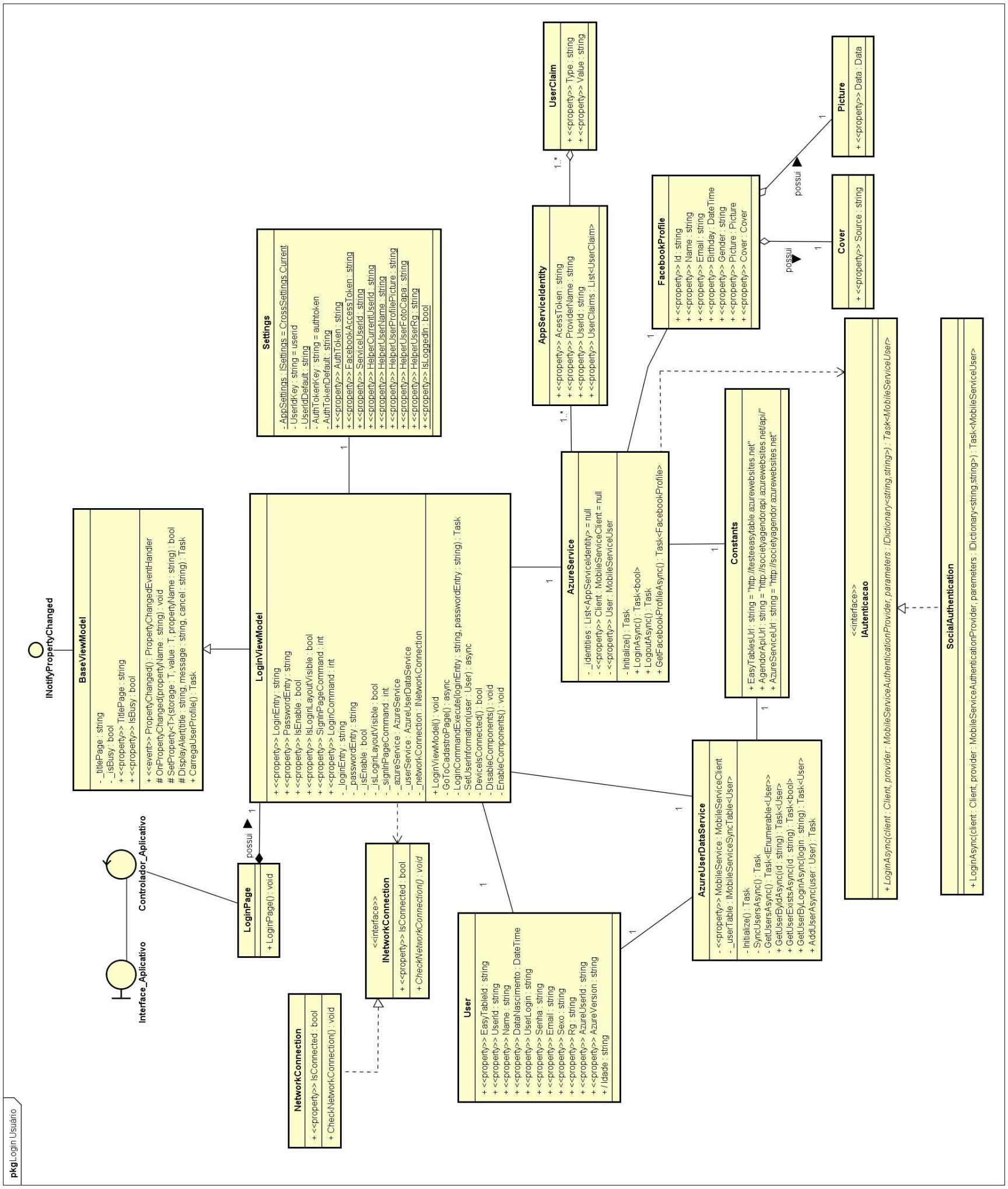
Fonte: Elaborado pelo autor.

2.5.4.2. Diagrama de Classe do Sistema

As figuras neste subcapítulo detalham as relações, os atributos, métodos das classes utilizadas. Os diagramas foram separados de acordo com as telas do aplicativo, pois, se ficassem todos juntos em apenas um diagrama, dificultariam o seu entendimento.

A Figura 12 ilustra os relacionamentos entre as classes da tela de login do sistema.

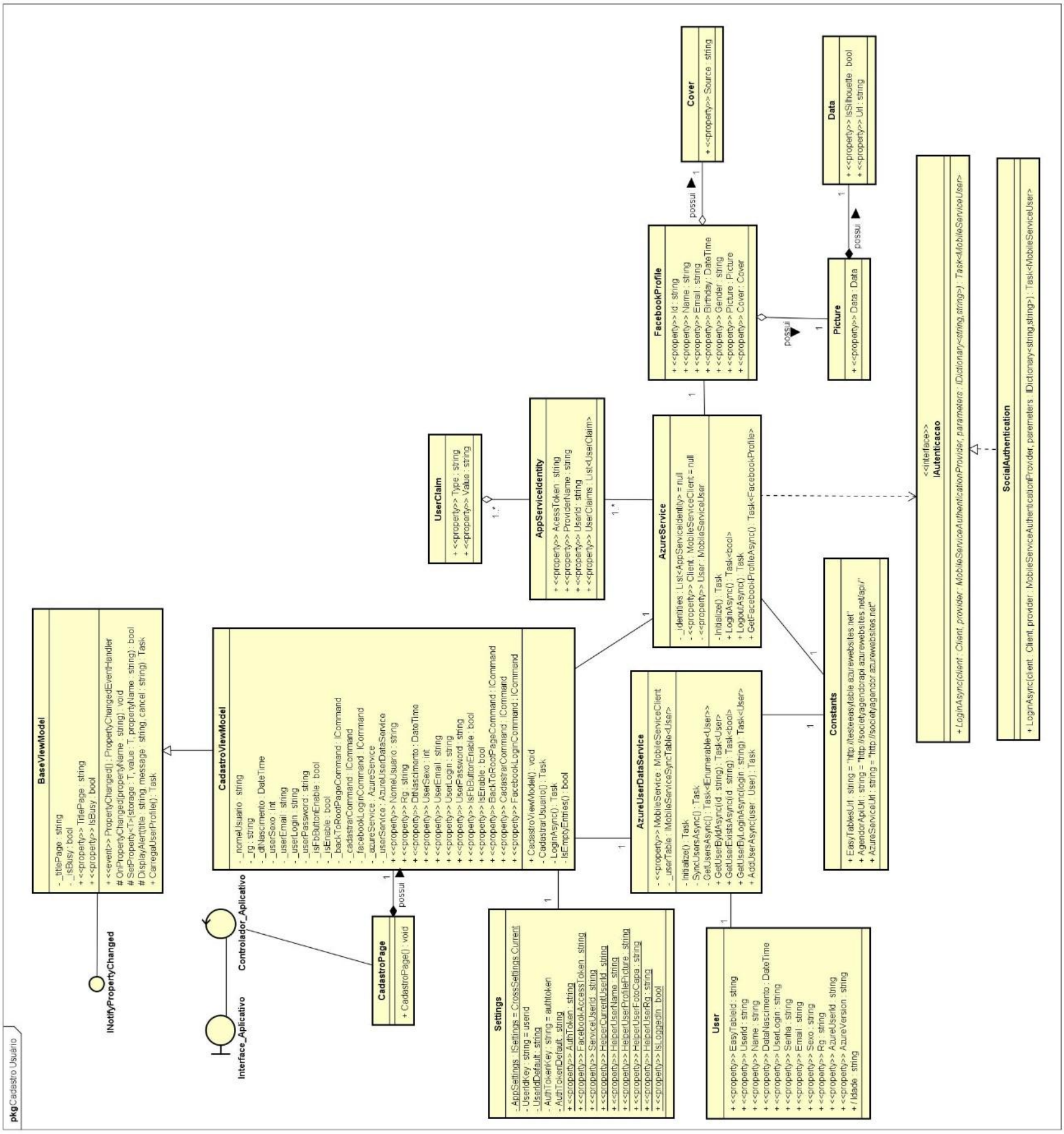
Figura 12 - Diagrama de Classe da Tela de Login.



Fonte: Elaborado pelo autor.

A Figura 13 apresenta os relacionamentos, atributos e métodos das classes da tela de cadastro de usuário.

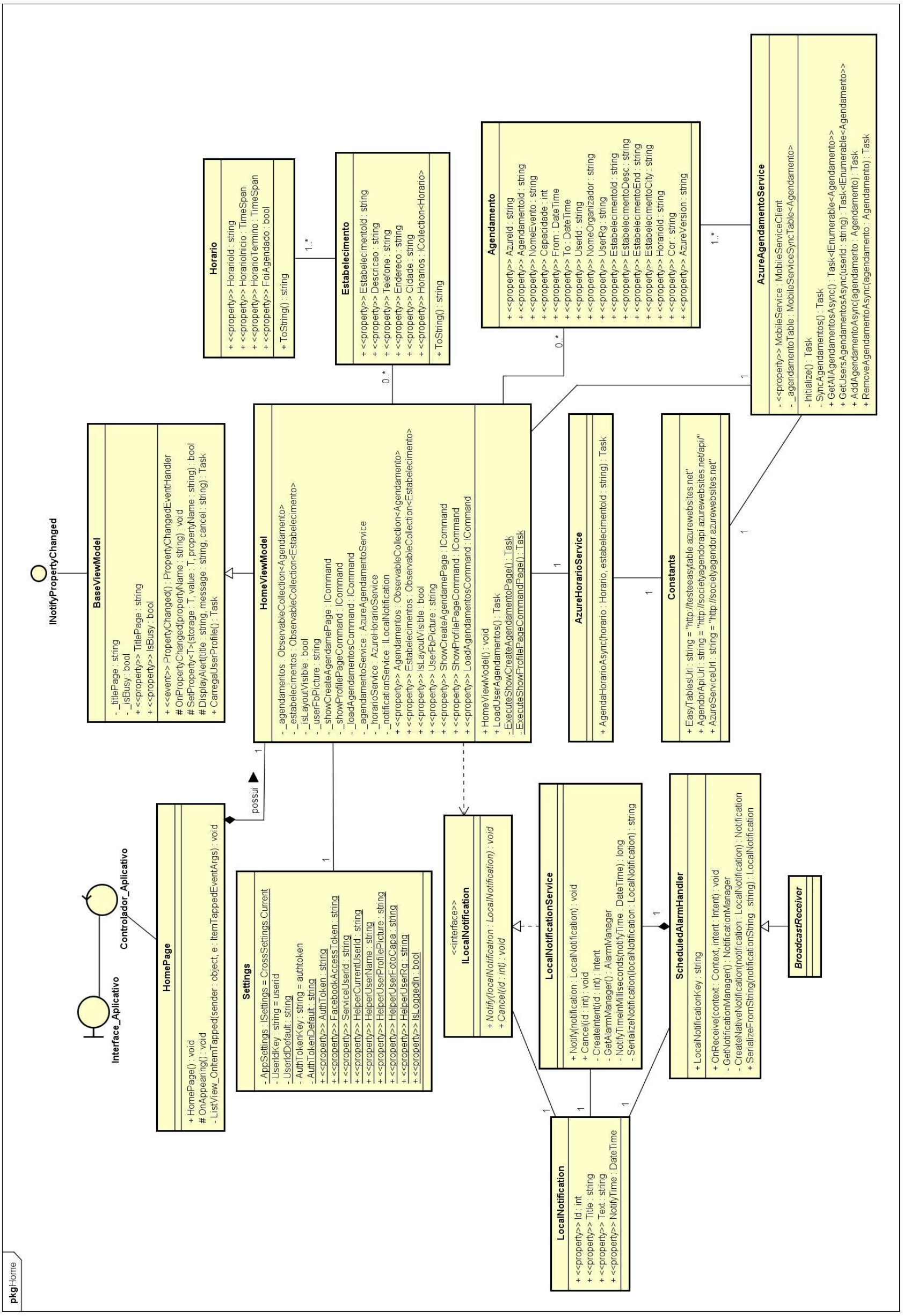
Figura 13 - Diagrama de Classe da Tela de Cadastro de usuário.



Fonte: Elaborado pelo autor.

A Figura 14 apresenta o diagrama da tela inicial do sistema.

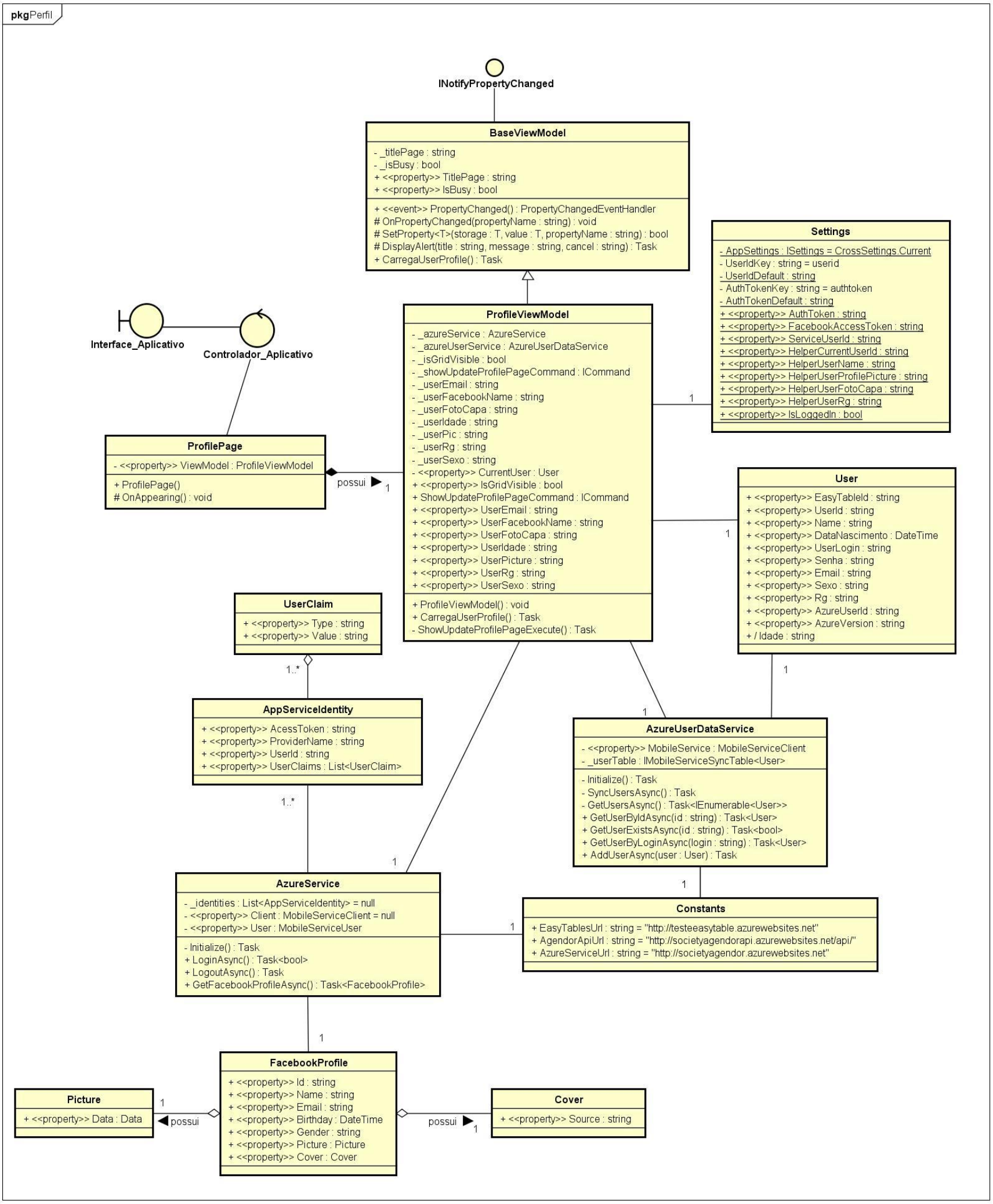
Figura 14 - Diagrama de Classe da Tela Inicial.



Fonte: Elaborado pelo autor.

A Figura 15 ilustra as relações, atributos e métodos das classes da tela de visualização de perfil do aplicativo.

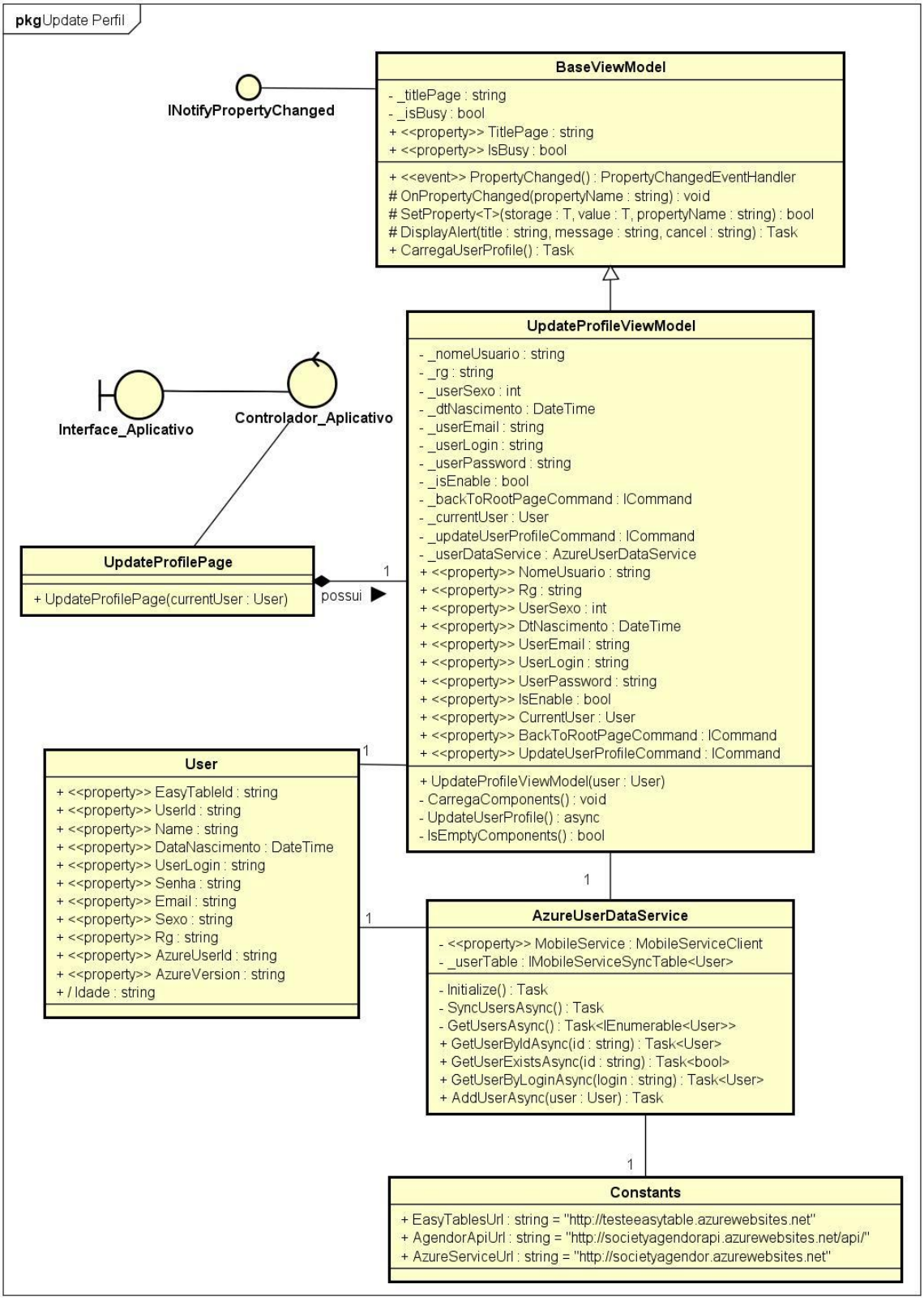
Figura 15 - Diagrama de Classe da Tela de Perfil do usuário.



Fonte: Elaborado pelo autor.

A Figura 16 apresenta os atributos, métodos e as relações entre as classes da tela de edição de perfil.

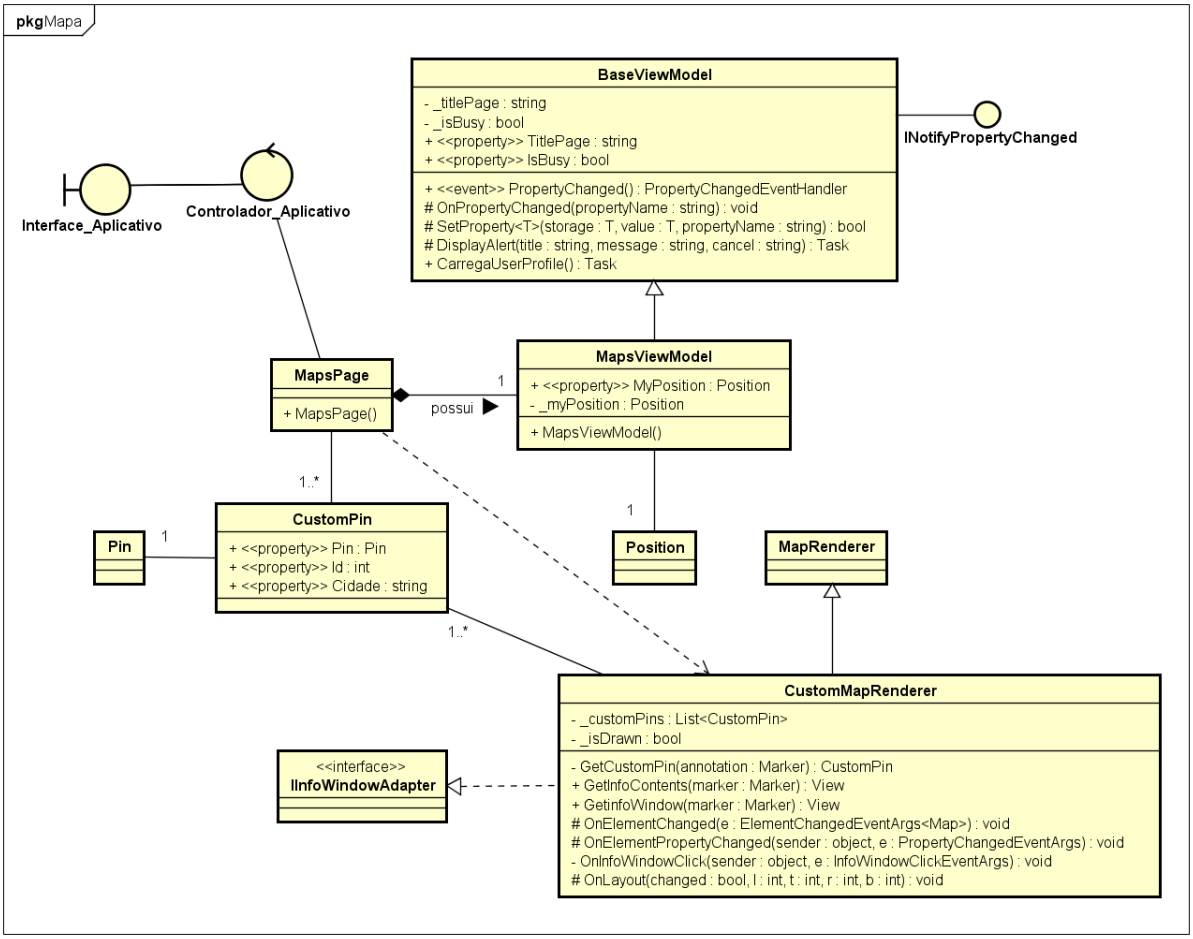
Figura 16 - Diagrama de Classe da Tela de Edição de Perfil.



Fonte: Elaborado pelo autor.

Na Figura 17 é ilustrado os atributos, métodos e relações das classes utilizadas na tela de exibição dos estabelecimentos cadastrados no sistema.

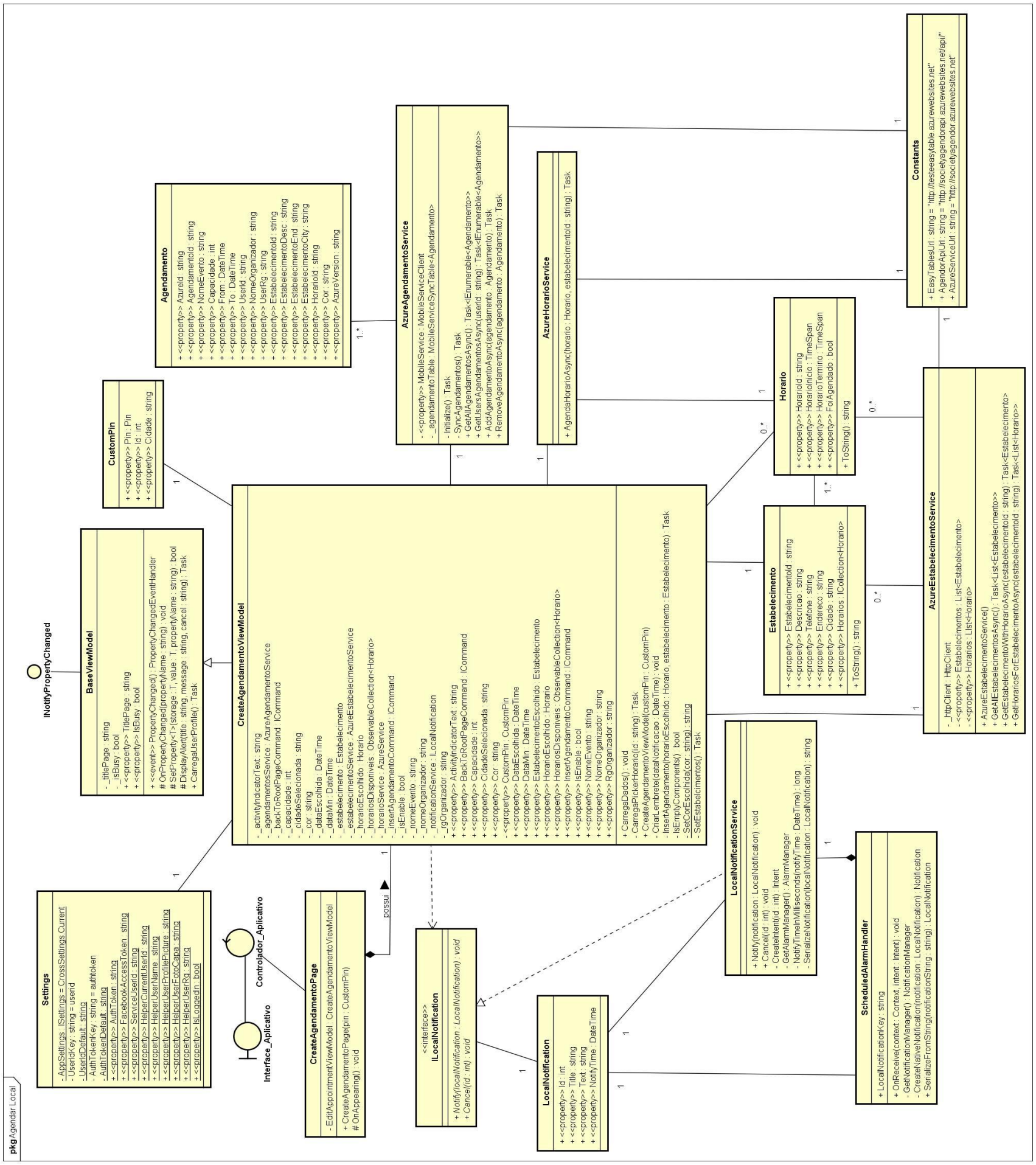
Figura 17 - Diagrama de Classe da Tela de exibição dos Estabelecimentos.



Fonte: Elaborado pelo autor.

A Figura 18 apresenta o diagrama de classe da tela de agendamento do estabelecimento.

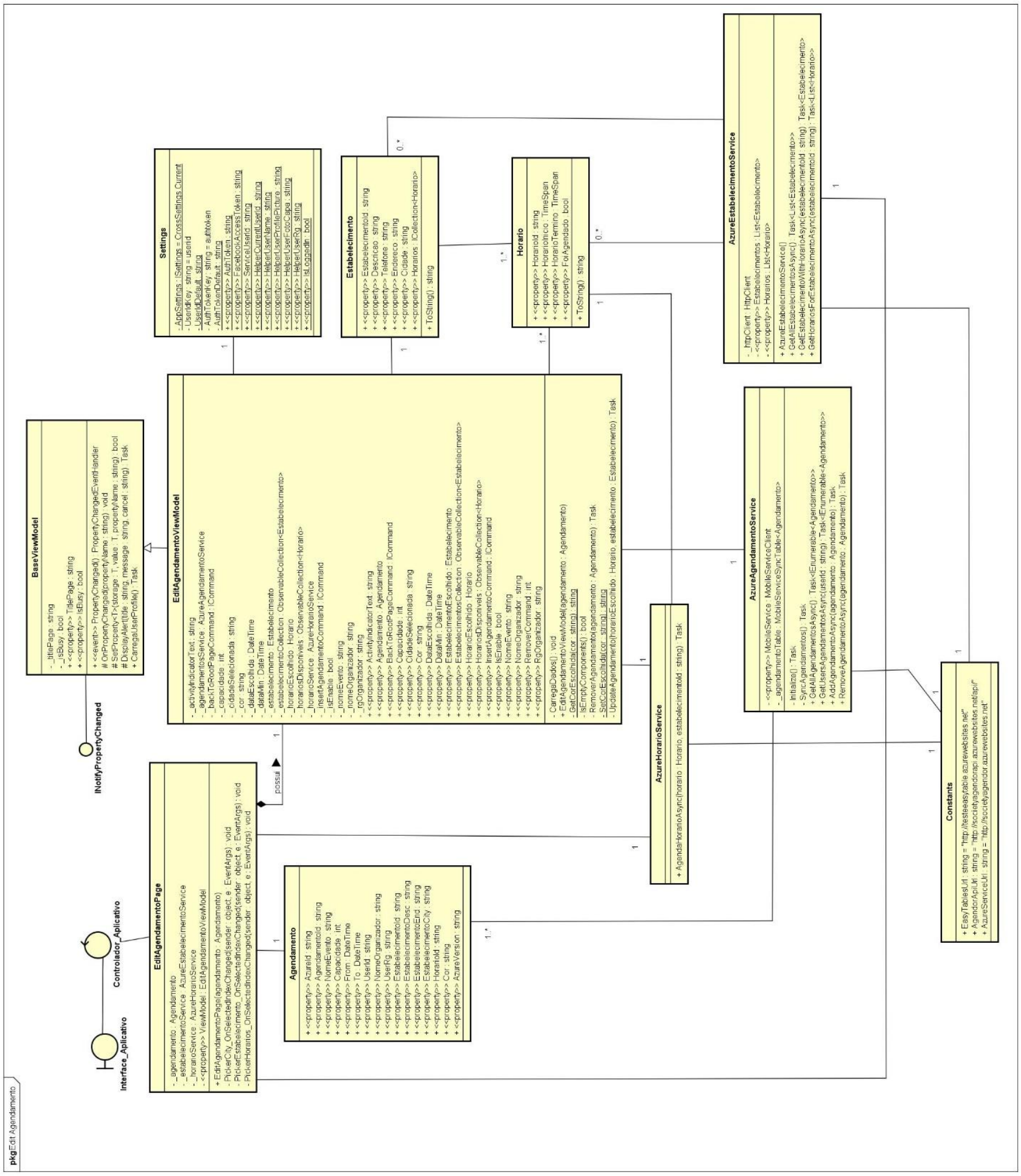
Figura 18 - Diagrama de Classe da Tela de Criação do Agendamento.



Fonte: Elaborado pelo autor.

Por fim, a Figura 19 ilustra os atributos, métodos e relações entre as classes da tela de edição de agendamento.

Figura 19 - Diagrama de Classe da Tela de Edição de Agendamento.



Fonte: Elaborado pelo autor.

2.6. Desenvolvimento

Neste capítulo será abordado o então desenvolvimento do sistema de locação de campo *society*. Será detalhado algumas das principais funções que o aplicativo possui e da API desenvolvida. Além de apresentar as telas que compõem o aplicativo, possibilitando a interação usuário-sistema.

2.6.1. API

Nas classes “EstabelecimentoController” e “HorarioController”, como podem ser vistas na Figura 11, são de grande importância, pois são elas que realizam a comunicação e recebem os dados do aplicativo, logo, através de seus métodos, decidem qual função executar de acordo com o cabeçalho localizado na URI, que vem do aplicativo.

Nesta Seção serão apresentadas com mais detalhe, as classes contidas no diagrama da Figura 11, mas sem os parâmetros de seus métodos.

A Tabela 19 detalha a classe “EstabelecimentoController”.

Tabela 19 - Descrição da classe EstabelecimentoController.

Nome da Classe	EstabelecimentoController
Resumo	Tem a função de retornar todos os estabelecimentos e suas informações contidas no banco de dados ou apenas os dados de certo estabelecimento.
Método	**Descrição**
GetEstabelecimentos	Retorna para o aplicativo todos os estabelecimentos contidos no banco de dados.
GetEstabelecimento	Retorna ao o aplicativo os dados do estabelecimento que contém o mesmo identificador do parâmetro.

Fonte: Elaborado pelo autor.

A Tabela 20 detalhada a classe “HorarioController”.

Tabela 20 - Descrição da classe HorarioController.

Nome da Classe	HorarioController
Resumo	Tem a função de retornar ou atualizar os horários de um estabelecimento. Através da atualização do horário, é possível saber se o estabelecimento está agendado em certo horário.
Método	**Descrição**
GetHorariosForEstabelecimento	Retorna para o aplicativo todos os horários de certo estabelecimento contido no banco de dados.
UpdateBookForAuthor	Este método tem a função de mudar o valor da propriedade que agenda o horário do estabelecimento para verdadeiro.

Fonte: Elaborado pelo autor.

A Tabela 21 descreve classe “EstabelecimentoRepository” que implementa a interface “IEstabelecimentoRepository”.

Tabela 21 - Descrição da classe EstabelecimentoRepository.

Nome da Classe	EstabelecimentoRepository
Resumo	Tem a função de tratar as funções de criação, exclusão, alteração das duas entidades (Estabelecimento e Horário) no banco de dados.
Método	**Descrição**
EstabelecimentoExists	Checa se já existe determinado estabelecimento com o mesmo valor de identificador no parâmetro. Se existir retorna verdadeiro senão falso.
GetEstabelecimentos	Retorna todos os estabelecimentos cadastrados no banco de dados.
GetEstabelecimento	Busca no banco de dados o estabelecimento correspondente ao valor do identificador no parâmetro e o retorna.
GetHorariosForEstabelecimento	Retorna do banco de dados os horários para o estabelecimento com o mesmo identificador do parâmetro.

GetHorarioForEstabelecimento	Retorna apenas o horário correspondente ao identificador e ao estabelecimento do parâmetro.
UpdateHorarioForEstabelecimento	Atualiza as informações do horário.
Save	Retorna verdadeiro ou falso se o banco de dados efetuou com sucesso uma transação.

Fonte: Elaborado pelo autor.

As classes “Horario” e “Estabelecimento” representam as entidades, elas apenas possuem atributos, já a classe “AgendamentoContext” tem apenas dois atributos que representam as tabelas das entidades e as ações relacionadas ao banco de dados, como inserção, exclusão, alteração etc.

2.6.2. Aplicativo

No subcapítulo 2.5.4.2 é apresentado o diagrama de classe do sistema e é perceptível que o sistema possui diversos atributos, classes, métodos e relações. Serão apresentados os resumos das classes com maior grau de importância e as descrições de seus respectivos métodos (não contendo sua assinatura).

Além das descrições e resumos, também serão abordadas as telas relacionadas as respectivas *ViewModels*. Lembrando que cada classe do tipo *ViewModel*, possui uma *View* que é construída por XAML (ver Seção 2.4.1 e 2.4.2) e que forma a interface de usuário.

A Tabela 22 apresenta com mais detalhe a descrição de cada método contido na classe “BaseViewModel”. Ela é a classe pai das *ViewModels* e, por ter esta característica de apenas possuir métodos que serão herdados à suas filhas, não possui uma *View*.

Tabela 22 - Descrição da classe BaseViewModel.

Nome da Classe	BaseViewModel
Resumo	Classe pai, possuindo atributos e métodos que serão herdados nas classes filhas sem precisar reescrever o mesmo código. Implementa o mecanismo de notificação, cuja função é alertar, todas as utilizações de uma propriedade, que seu valor foi modificado,

	fazendo com que logo em seguida a propriedade assuma este novo valor.
Método	**Descrição**
OnPropertyChanged	Método que notifica que alguma mudança aconteceu em uma propriedade.
SetProperty	Método utilizado para atualizar o valor de uma propriedade. Ele compara o valor atual com o valor novo. Se o valor atual for diferente do novo, o método faz com que a propriedade recebe o valor novo. Se não houve alteração, o atual permanece na propriedade.
DisplayAlert	Método que exibe um alerta para o usuário.
CarregaUserProfile	Este método não foi implementado de fato na classe, pois algumas classes o utilizam de maneira diferente, por isso as classes que utilizam deste, irão sobrescrevê-lo.

Fonte: Elaborado pelo autor.

A Tabela 23 apresenta com mais detalhes, as funções da classe “CadastroViewModel”.

Tabela 23 - Descrição da classe CadastroViewModel.

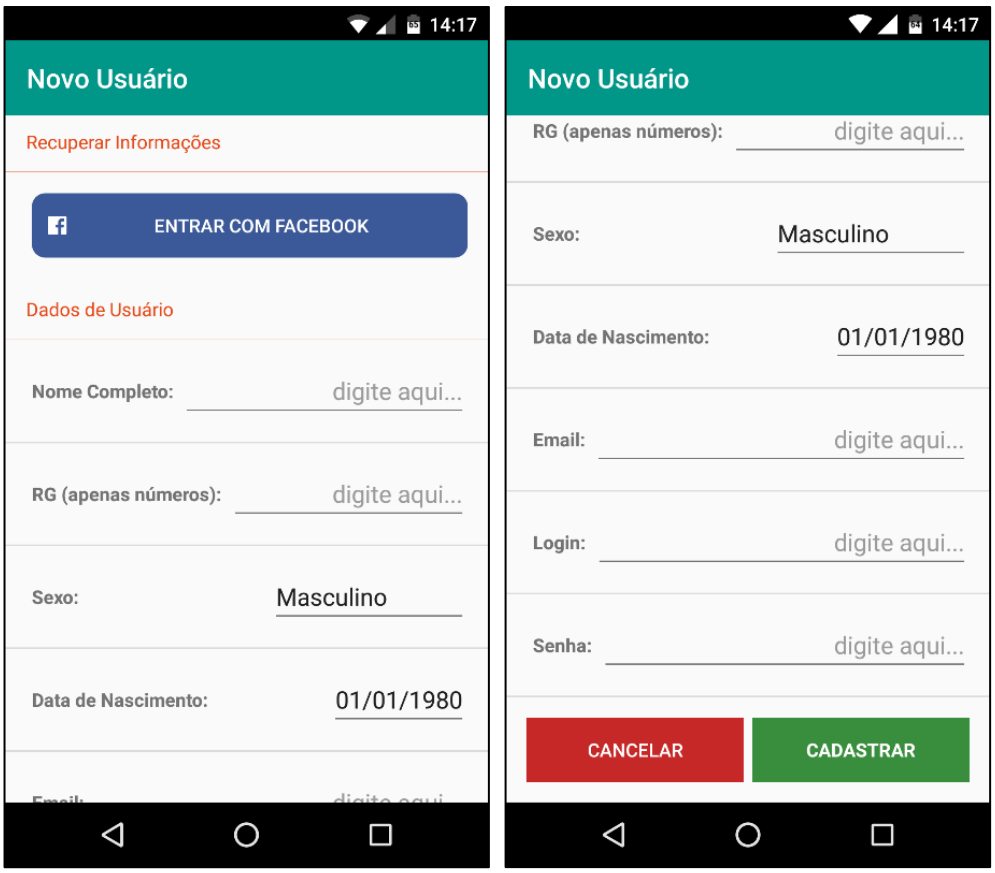
Nome da Classe	CadastroViewModel
Resumo	Possui a função de cadastrar um usuário no sistema. Possuindo métodos que conferem se os dados das propriedades estão válidos para que então possa chamar a classe de serviço e mandar os dados para a API e inserir no banco de dados.
Método	**Descrição**
CadastroViewModel	O construtor da classe tem as seguintes funções: inicializar as variáveis das classes de serviço, atribuir um título a tela de cadastro e habilitar os botões da tela.

CadastrarUsuario	Método que recolhe os valores das propriedades e os adiciona em uma instância de uma entidade usuário, logo depois, manda este valor transformado para o serviço responsável por enviar os dados para API.
LoginAsync	Método que faz o usuário entrar em sua conta do Facebook. Após este processo de login efetuado, o sistema ativa o serviço que recupera os dados da conta do usuário, logo então, atribui os valores às propriedades que exibirão as informações na tela.
IsEmptyEntries	Método que retorna verdadeiro ou falso caso alguma propriedade esteja vazia ou com algum espaço em branco.

Fonte: Elaborado pelo autor.

A tela que é ligada ao cadastro de usuário é apresentada na Figura 20. A *View* “CadastroPage.xaml” possui a função de criar esta interface de usuário.

Figura 20 - Tela de Cadastro de Usuário.



Fonte: Elaborado pelo autor.

A seguir, a Tabela 24 apresenta os detalhes da classe “LoginViewModel” e de seus métodos.

Tabela 24 - Descrição da classe LoginViewModel.

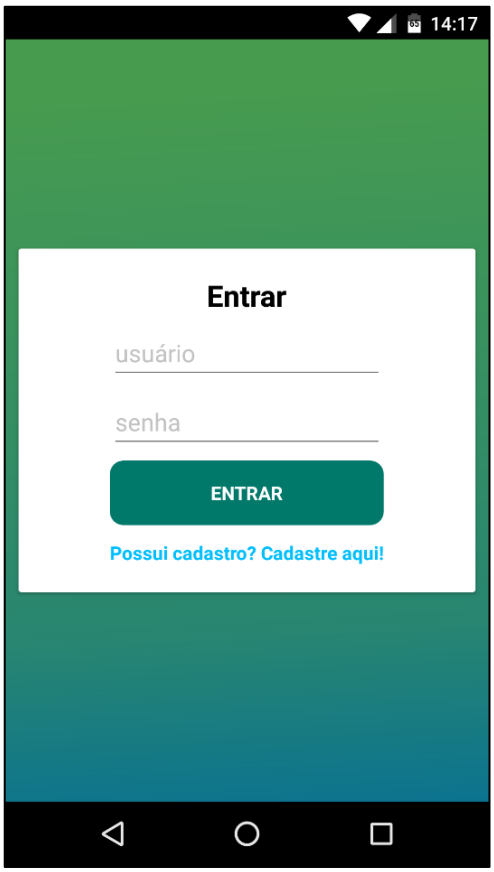
Nome da Classe	LoginViewModel
Resumo	Realiza a entrada do usuário no sistema. Faz a checagem dos dados de usuário inseridos no campo de nome de usuário e senha, checa se o aparelho possui conexão com a Internet.

Método	Descrição
LoginViewModel	O construtor da classe tem as seguintes funções: inicializar as variáveis das classes de serviço, atribuir um título a tela de login e habilitar os botões da tela.
GoToCadastroPage	Método que abre a tela de cadastro de usuário.
LoginCommandExecute	Método que realiza a entrada do usuário no sistema. Chama os métodos que realizam a checagem de conexão com a Internet e dos campos de login e senha.
SetUserInformation	Se realizado o login, o sistema ativa este método para que o sistema recupera as informações da conta do Facebook do usuário e faz com que certas propriedades recebam esses dados.
DeviceIsConnected	Realiza a checagem de conexão com a Internet do aparelho celular.
DisableComponents	Desabilita as caixas de texto do formulário de cadastro.
EnableComponents	Habilita as caixas de texto do formulário de cadastro.

Fonte: Elaborado pelo autor.

A Figura 21 apresenta a interface de login do aplicativo, que está associada à “LoginPage.xaml”.

Figura 21 - Tela de Login do aplicativo.



Fonte: Elaborado pelo autor.

Na Tabela 25 é possível conferir, com mais detalhes, os métodos da classe “HomeViewModel”.

Tabela 25 - Descrição da classe HomeViewModel.

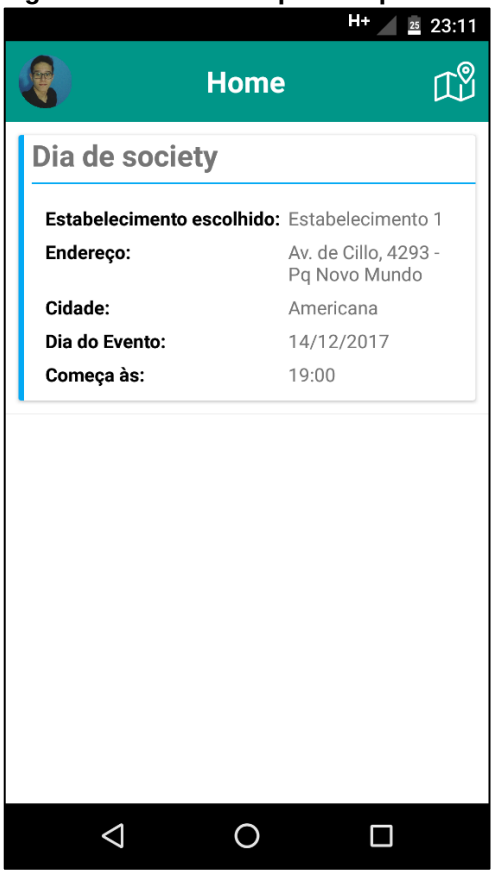
Nome da Classe	HomeViewModel
Resumo	Consiste em recuperar e mostrar os agendamentos do usuário. Se o agendamento passar de sua data, a própria classe ao executar o comando de atualizar, exclui o item automaticamente.
Método	**Descrição**
HomeViewModel	O construtor da classe tem as seguintes funções: inicializar as variáveis das classes de serviço, atribuir um título a tela inicial, habilitar a visibilidade da tela e carregar uma foto de perfil padrão.

LoadUserAgendamentos	Método que recupera apenas os agendamentos do atual usuário e checa se o item já passou da data de agendada e então o remove da tela e manda uma requisição de exclusão para o banco de dados através da chamada do serviço.
ExecuteShowCreateAgendamentoPage	Abre a tela que aponta os estabelecimentos cadastrados em um mapa.
ExecuteShowProfilePageCommandPage	Abre a tela de perfil de usuário.

Fonte: Elaborado pelo autor.

A tela principal do aplicativo está vinculada com a *View* “HomePage.xaml”, que tem a função de criar a interface de usuário apresentada na Figura 22.

Figura 22 - Tela Principal do aplicativo.



Fonte: Elaborado pelo autor.

Na Tabela 26 é possível conferir a descrição da classe “ProfileViewModel” e de seus métodos.

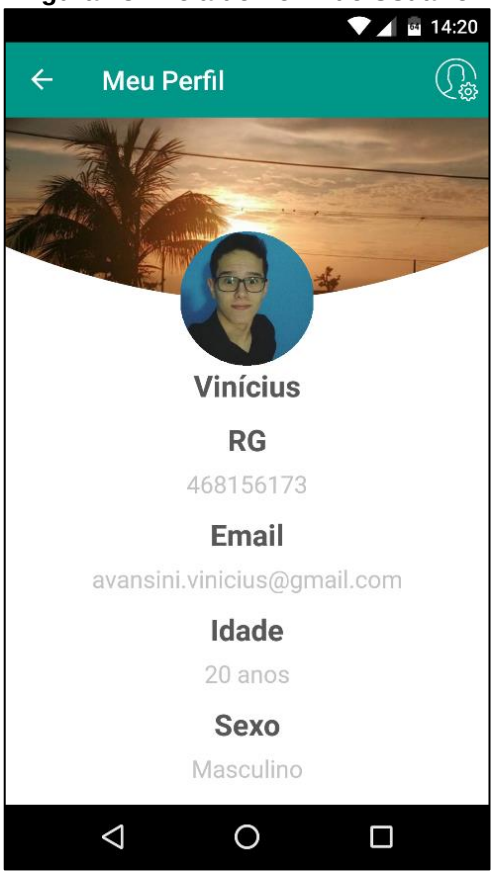
Tabela 26 - Descrição da classe ProfileViewModel.

Nome da Classe	ProfileViewModel
Resumo	Consiste em recuperar e mostrar as informações de perfil do usuário, além de fazer uma requisição à Graph API do Facebook para recuperar as fotos de capa e de perfil do usuário.
Método	**Descrição**
ProfileViewModel	O construtor da classe tem as seguintes funções: inicializar as variáveis das classes de serviço, atribuir um título a tela de perfil, habilitar a visibilidade da tela.
CarregaUserProfile	Sobrescreve o método que foi criado na classe pai (“BaseViewModel”), cuja função é realizar uma requisição à Graph API para que esta retorne informações do usuário (fotos de capa e de perfil do Facebook do usuário).
ShowUpdateProfilePageExecute	Abre a tela de edição de perfil.

Fonte: Elaborado pelo autor.

A classe que tem a função de interpretar o código XAML e criar a interface de usuário correspondente a Figura 23, ocasionando a visualização do perfil do usuário, é a “ProfilePage.xaml”.

Figura 23 - Tela de Perfil de Usuário.



Fonte: Elaborado pelo autor.

A Tabela 27 apresenta a descrição da classe “UpdateProfileViewModel” e de seus métodos.

Tabela 27 - Descrição da classe UpdateProfileViewModel.

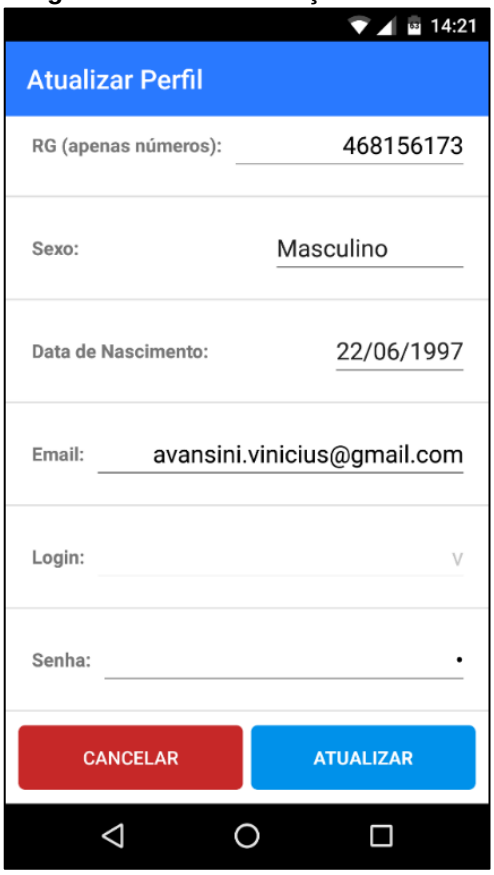
Nome da Classe	UpdateProfileViewModel
Resumo	Classe que atualiza as informações de perfil do usuário. Seus métodos fazem chamadas às classes de serviço que realizam requisições de atualização dos dados do usuário no banco de dados do Azure. Além de realizar a validação dos campos de texto preenchidos pelo usuário.

Método	Descrição
UpdateProfileViewModel	O construtor da classe tem as seguintes funções: inicializar as variáveis das classes de serviço, atribuir um título a tela de edição de perfil e carregar as caixas de texto com os dados do usuário.
CarregaComponents	Atribui às propriedades os valores das informações do usuário atual, para que ele possa visualizar e alterá-los, se quiser.
UpdateUserProfile	Valida os campos preenchidos pelo usuário e, se estiverem corretos, atribui os valores das propriedades a uma instância da entidade usuário, transformando em apenas um item, para que assim o serviço entenda e possa atualizar as informações do usuário no banco de dados.
IsEmptyComponents	Método que retorna verdadeiro ou falso caso alguma propriedade esteja vazia ou com algum espaço em branco.

Fonte: Elaborado pelo autor.

A Figura 24 apresenta a interface de edição das informações do usuário e está associada a *View* “UpdateProfilePage.xaml”.

Figura 24 - Tela de Edição de Perfil.



Fonte: Elaborado pelo autor.

Na Tabela 28 é possível conferir com mais detalhe o resumo da classe “MapsViewModel”.

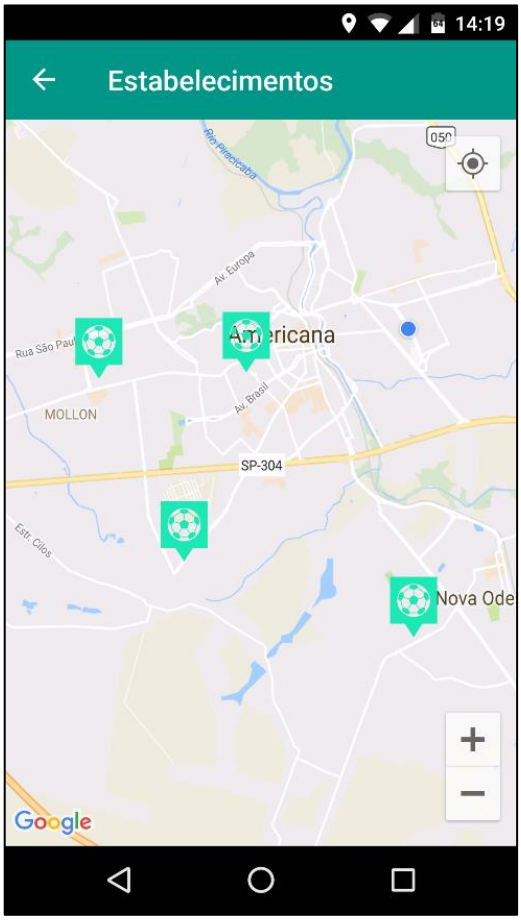
Tabela 28 - Descrição da classe MapsViewModel.

Nome da Classe	MapsViewModel
Resumo	Tem a função de apenas apontar os estabelecimentos cadastrados, no mapa. Possibilitando que o usuário, ao clicar sobre um marcador, o leve direto para a tela de agendamento daquele estabelecimento.
Método	**Descrição**
MapsViewModel	O construtor da classe possui a tarefa de atribuir um título a tela e recuperar a posição do GPS do aparelho celular.

Fonte: Elaborado pelo autor.

Na *View*, “MapsPage.xaml”, é criada a interface que exibirá os estabelecimentos cadastrados no mapa. Pode-se conferir, com mais detalhes, na Figura 25, a tela que é exibida ao usuário.

Figura 25 - Tela de Estabelecimentos Cadastrados.



Fonte: Elaborado pelo autor.

A Tabela 29 apresenta a descrição dos métodos da classe “CreateAgendamentoViewModel”.

Tabela 29 - Descrição da classe CreateAgendamentoViewModel.

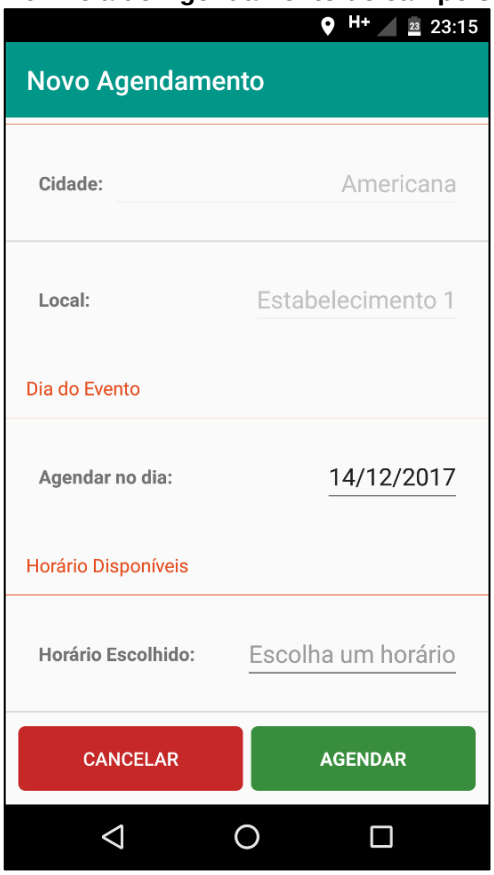
Nome da Classe	CreateAgendamentoViewModel
Resumo	Responsável por recuperar as informações do agendamento realizado pelo usuário através da classe de serviço que faz uma requisição ao banco de dados do Azure.

Método	Descrição
EditAgendamentoViewModel	O construtor da classe tem as seguintes funções: inicializar as variáveis das classes de serviço, atribuir um título a tela, carregar as caixas de texto com os dados do agendamento.
CarregaDados	Atribui a uma propriedade o valor da cidade do estabelecimento selecionado. Assim que a propriedade for modificada, ela ativará o método “SetEstabelcimentos”.
InsertAgendamento	Método responsável por agendar um local. Realiza a validação das informações inseridas pelo usuário, logo em seguida caso estas sejam válidas, o método atribui a uma instância da classe “Agendamento”, os valores correspondentes as suas propriedades e a envia para o serviço que realizará a ação de inserção no banco de dados do Azure. E por fim, cria uma notificação uma hora antes do evento acontecer.
SetEstabelcimentos	Carrega a caixa de texto com a descrição do estabelecimento escolhido.
CarregaPickerHorario	Carrega um componente com os horários disponíveis para aquele estabelecimento.
SetCorEscolhida	Converte um nome da cor escolhida pelo usuário em hexadecimal, facilitando depois para o sistema aplicar no componente visual da tela.
IsEmptyComponents	Método que retorna verdadeiro ou falso caso alguma propriedade esteja vazia ou com algum espaço em branco.
CriarLembrete	Cria uma notificação a partir de uma hora antes do evento acontecer.

Fonte: Elaborado pelo autor.

Na Figura 26 é apresentada a tela que o usuário loca um campo *society* em um determinado estabelecimento. Esta é criada a partir da *View* “CreateAgendamento.xaml”.

Figura 26 - Tela de Agendamento de campo *society*.



Fonte: Elaborado pelo autor.

E por fim, a Tabela 30 apresenta a descrição das funções da classe “EditAgendamentoViewModel” e de seus métodos.

Tabela 30 - Descrição da classe EditAgendamentoViewModel.

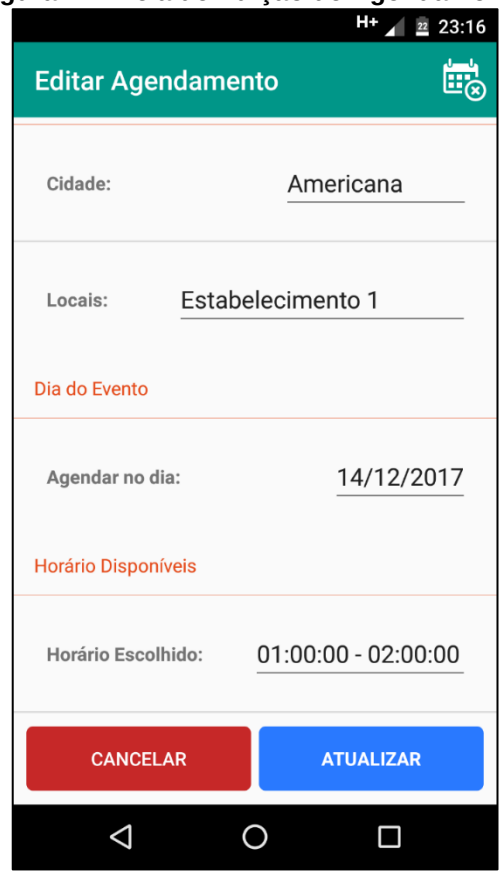
Nome da Classe	EditAgendamentoViewModel
Resumo	Responsável por recuperar as informações do estabelecimento e do horário, escolhidas pelo usuário, afim de agendar o local.

Método	Descrição
CreateAgendamentoViewModel	O construtor da classe tem as seguintes funções: inicializar as variáveis das classes de serviço, atribuir um título a tela, carregar as caixas de texto com os dados do estabelecimento escolhido e alguns dados do usuário atual.
CarregaDados	Atribui a uma propriedade o valor da cidade do estabelecimento selecionado. Assim que a propriedade for modificada, ela ativará o método “SetEstabelcimentos”.
UpdateAgendamento	Método responsável por alterar o agendamento já realizado pelo usuário. Realiza a validação das informações inseridas pelo usuário, logo em seguida caso estas sejam válidas, o método atribui a uma instância da classe “Agendamento”, os valores correspondentes as suas propriedades e a envia para o serviço que realizará a ação de atualização no banco de dados do Azure.
RemoverAgendamento	O método exclui o atual agendamento do usuário, tornando o horário disponível para futuras locações.
SetCorEscolhida	Converte um nome da cor escolhida pelo usuário em hexadecimal, facilitando depois para o sistema aplicar no componente visual da tela.
IsEmptyComponents	Método que retorna verdadeiro ou falso caso alguma propriedade esteja vazia ou com algum espaço em branco.
GetCorEscolhida	Converte o valor da cor em hexadecimal para a descrição da cor correspondente a este valor.

Fonte: Elaborado pelo autor.

Na Figura 27 é apresentada a tela de edição de um agendamento, que é criada a partir da classe “EditAgendamentoPage.xaml”.

Figura 27 - Tela de Edição de Agendamento.



Fonte: Elaborado pelo autor.

2.7. Testes de Aceitação

Os testes direcionados ao *software* desenvolvido, mostram se o sistema realmente atende as funções propostas e se existem defeitos ou anomalias que ocasionam a mal funcionalidade deste (SOMMERVILLE, 2011). A partir dos resultados obtidos dos testes, será decidido se haverá retrabalho no sistema (por parte dos desenvolvedores) ou não.

FREEMAN e PRYCE (2012, p. 10) contam que existem diversas terminologias para testes de aceitação. Podendo ser: “testes funcionais”, “testes de cliente” ou “testes de sistema”. Mas além destas opções, segundos os autores, “o importante é ter clareza” sobre como usar estes testes, é ajudar o desenvolvedor a entender e concordar o que se deve fazer em seguida. Usar os testes como ferramenta de guia, para que possa garantir que nada foi deixado de lado na hora do desenvolvimento (FREEMAN; PRYCE, 2012, p. 10).

Para apresentar que o sistema atende aos requisitos funcionais apresentados na Seção 2.2.1, foram efetuados testes que auxiliam a demonstração dos resultados. Nesta Seção são apresentados os relatórios dos testes efetuados em cima dos requisitos funcionais do aplicativo, chegando à conclusão que todos foram atendidos.

Na Tabela 31 é apresentado o relatório do teste do requisito RF001.

Tabela 31 – Relatório do Teste de Aceitação do Requisito Funcional 001.

Identificador	TA001
Objeto de Teste	Requisito Funcional RF001.
Localização	Tela de Login > Tela de Cadastro de Usuário
Procedimento	1. O usuário pressiona o botão “Entrar com Facebook”. 2. O sistema abre uma tela de autenticação do Facebook. 3. O usuário preenche os campos de login e senha do Facebook e pressiona o botão “Entrar”. 4. O usuário aceita as permissões do aplicativo pressionando o botão “Continuar”.

	5. O sistema recebe o *Token* de Acesso do usuário e requisita as informações para preencher os campos de texto do aplicativo com os dados do usuário.
Resultado Esperado	1. Sistema recebe o *Token* de Acesso do usuário. 2. Sistema preenche os campos de texto com as informações recuperadas da Graph API do Facebook a partir do *Token* de Acesso.
Resultado Obtido	Realizado com sucesso.

Fonte: Elaborado pelo autor.

Na Tabela 32 é possível ver o relatório do requisito RF002.

Tabela 32 – Relatório do Teste de Aceitação do Requisito Funcional 002.

Identificador	TA002
Objeto de Teste	Requisito Funcional RF002.
Localização	Tela de Login > Tela Principal
Procedimento	1. Usuário entra no aplicativo. 2. O sistema irá requisitar à API que retorne, do banco de dados, os agendamentos do usuário atual. 3. Ao receber o / os agendamentos, o sistema os exibe na tela.
Resultado Esperado	1. O sistema pode receber nenhum ou muitos itens da API. 2. O sistema deverá organizar a exibição dos agendamentos em ordem crescente, de acordo com a data agendada. 3. Se algum erro acontecer, o sistema irá exibir um alerta ao usuário, acusando qual foi o erro.
Resultado Obtido	Resultados 1 e 2 realizados com sucesso. Resultado 3 só exibirá a exceção gerada pelo sistema e não apontando o erro ocorrido.

Fonte: Elaborado pelo autor.

Na Tabela 33 é apresentado o relatório do requisito RF003.

Tabela 33 – Relatório do Teste de Aceitação do Requisito Funcional 003.

Identificador	TA003
Objeto de Teste	Requisito Funcional RF003.
Localização	Tela de Login
Procedimento	1. Usuário preenche os campos de login e senha [válidos]. 2. O usuário pressiona o botão “Entrar”. 3. O sistema abre uma tela de login com o Facebook. 4. Sistema direciona o usuário para a tela principal do aplicativo.
Resultado Esperado	1. O sistema abrirá uma tela de login com o Facebook, onde o usuário irá entrar com seu login e senha do provedor. 2. Facebook valida as informações do usuário. 3. O usuário irá aceitar as permissões do aplicativo. 4. *Token* de Acesso do usuário é armazenado.
Resultado Obtido	Resultados 1, 3 e 4 realizados com sucesso. Resultado 2 foi realizado com sucesso, mas o sistema não realiza o tratamento de algum erro caso o usuário rejeite as permissões do aplicativo.

Fonte: Elaborado pelo autor.

Na Tabela 34 é possível ver o relatório do teste do requisito RF004.

Tabela 34 – Relatório do Teste de Aceitação do Requisito Funcional 004.

Identificador	TA004
Objeto de Teste	Requisito Funcional RF004.
Localização	Tela de Login > Tela Principal
Procedimento	1. Usuário entra com sucesso no sistema. 2. O sistema manda uma requisição a API para que retorne apenas os agendamentos correspondentes ao identificador do usuário atual. 3. Sistema organiza os agendamentos em ordem crescente. Em paralelo, faz uma checagem se algum agendamento já foi

	realizado, de acordo com a sua data. Se a data é ultrapassada, o sistema excluí o item.
Resultado Esperado	1. A API deve retornar no mínimo um agendamento para que possa checar se a data do agendamento é ultrapassada. 2. Se for ultrapassada, o sistema excluí.
Resultado Obtido	Realizado com sucesso.

Fonte: Elaborado pelo autor.

Na Tabela 35 é apresentado o relatório do teste do requisito RF005.

Tabela 35 - Relatório do Teste de Aceitação do Requisito Funcional 005.

Identificador	TA005
Objeto de Teste	Requisito Funcional RF005.
Localização	Tela Principal > Tela de Estabelecimentos
Procedimento	1. O usuário deve pressionar o botão que exibe a tela do mapa de estabelecimentos. 2. O sistema cria marcações na API do Google Maps de acordo com os estabelecimentos já estabelecidos em seus métodos. 3. O sistema capta a localização atual do usuário através do GPS do aparelho celular e libera a visualização da tela para o usuário.
Resultado Esperado	1. A API do Google Maps irá demarcar os estabelecimentos (já estabelecidos pelo método) através de *pins*. 2. O sistema irá identificar a posição atual do usuário, através do GPS. 3. Se o GPS estiver desligado, o sistema irá alertar o usuário.
Resultado Obtido	Resultado 1 e 2 realizados com sucesso. O resultado 3 não foi desenvolvido.

Fonte: Elaborado pelo autor.

Na Tabela 36 é apresentado o relatório do teste do requisito RF006.

Tabela 36 - Relatório do Teste de Aceitação do Requisito Funcional 006.

Identificador	TA006
Objeto de Teste	Requisito Funcional RF006.
Localização	Tela de Login
Procedimento	1. O usuário preenche o campo de login e senha. 2. Pressiona o botão “Entrar”. 3. O sistema valida se o dispositivo celular possui conexão com a Internet.
Resultado Esperado	Mesmo se o usuário preencheu usuário e senha incorretos, o sistema exibirá um alerta para usuário, informando que o aparelho não está conectado com a Internet.
Resultado Obtido	Realizado com sucesso.

Fonte: Elaborado pelo autor.

Na Tabela 37 é possível ver o relatório do teste do requisito RF007.

Tabela 37 - Relatório do Teste de Aceitação do Requisito Funcional 007.

Identificador	TA007
Objeto de Teste	Requisito Funcional RF007.
Localização	Tela Principal > Tela de Edição de Agendamento
Procedimento	1. O usuário seleciona um agendamento na tela principal, onde são exibidos apenas os agendamentos referentes ao seu identificador. 2. O sistema abre a tela de edição com os detalhes do agendamento selecionado e prontos para serem editados.
Resultado Esperado	Serão exibidos apenas os agendamentos relacionados ao identificador do usuário atual.
Resultado Obtido	Realizado com sucesso.

Fonte: Elaborado pelo autor.

Na Tabela 38 é apresentado o relatório do teste relacionado ao requisito RF008.

Tabela 38 - Relatório do Teste de Aceitação do Requisito Funcional 008.

Identificador	TA008
Objeto de Teste	Requisito Funcional RF008.
Localização	Tela Principal > Tela de Estabelecimentos > Tela de Novo Agendamento
Procedimento	1. O usuário seleciona o *pin* do estabelecimento que ele queira agendar. 2. O usuário preenche os campos de texto e escolhe um horário. 3. Ao pressionar o botão “Agendar”, o sistema valida os campos preenchidos e cria uma notificação uma hora antes do evento acontecer.
Resultado Esperado	O sistema cria uma notificação local uma hora antes do evento da locação acontecer.
Resultado Obtido	Realizado com sucesso.

Fonte: Elaborado pelo autor.

3. CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo final, desenvolver um aplicativo móvel que facilitasse a ação de agendar um campo de futebol *society*. O então Society Agendor, foi desenvolvido utilizando o *framework* Xamarin em conjunto com a linguagem C# e o Visual Studio 2017. Lembrando que o aplicativo foi apenas desenvolvido para a plataforma Android, mas ao se utilizar do Xamarin, possibilita o desenvolvedor a criar aplicativos para mais de uma, como para a plataforma iOS e Windows.

Durante os capítulos foram apresentados os processos de desenvolvimento do aplicativo, detalhando e documentando desde a metodologia de desenvolvimento incremental, os requisitos, o planejamento, os recursos e as tecnologias utilizadas, os diagramas e por fim o então desenvolvimento, dando origem ao aplicativo construído.

No desenvolvimento do aplicativo, houveram dificuldades, sendo que, para facilitar as operações no banco de dados localizado no Microsoft Azure e, tratar as entidades, precisou-se desenvolver uma API que se assegurava de fazer estas operações e devido a este fator, o autor teve que aprender como desenvolver uma API. A programação utilizando o *framework* Xamarin foi uma outra dificuldade encontrada, pois, como dito, a plataforma ainda está se amadurecendo, alguns erros são encontrados no meio do caminho do desenvolvimento e foi um ponto onde o estudo teve que ser aprofundado para entender um pouco da arquitetura e as metodologias que são utilizadas para a programação.

Entretanto, com a ajuda de artigos e a própria documentação do Xamarin e de outros serviços fornecidos pela Microsoft, foi de grande importância para o desenvolvimento deste aplicativo além de outros pequenos projetos criados antes. O aplicativo executa todos os requisitos que foram apontados e cumpre o que promete, mas precisa ter uma gama maior de estabelecimentos associados.

Como possíveis trabalhos futuros, melhorias devem ser feitas tanto na documentação do projeto quanto no desenvolvimento de novas possibilidades para o aplicativo. Sendo sugerido:

- Desenvolver métodos que possam convidar amigos para o jogo e ver eles aceitando o convite para o evento;

- Possibilidade de calcular o estabelecimento mais próximo dos convidados, ou seja, o ponto de intersecção;
- Construir um portal para o cadastro de estabelecimentos, onde esses poderão manter os seus horários de funcionamento;
- Desenvolver uma metodologia de autenticação na API, possibilitando a segurança no acesso às informações do banco de dados;
- Migrar do serviço de *Easy Tables* API, do Azure, para a própria API desenvolvida, focalizando as transações de dados em apenas um lugar.

REFERÊNCIAS

ANDERSON, Rick. ASP.NET Core: Adding a controller. **Microsoft Docs**, 2017. Disponível em: <https://docs.microsoft.com/pt-br/aspnet/core/tutorials/first-mvc-app/adding-controller>. Acesso em 07 de jun. 2017, às 21h.

ARAÚJO, Everton Coimbra de. A evolução da linguagem de programação C#. **DevMedia**, 2013. Disponível em: <http://www.devmedia.com.br/a-evolucao-da-linguagem-de-programacao-c/28639>. Acesso em 03 de maio 2017, às 17h53min.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 6023 : informação e documentação : Referências : elaboração. Rio de Janeiro, 2002. 24p.

BERNARDO, Letícia Ellen. **CINE COLLECTION:** um aplicativo para recomendação de filmes. 2017. TG (Trabalho de Graduação do Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas), Faculdade de Tecnologia de Americana, Americana, 2017.

CAMPOS, Rafael Tweedie. **Software como Serviço:** um framework para fornecer ferramentas de simulação analítica. 2012. 95 f. Dissertação (Mestrado em Ciência da Computação, Processamento Paralelo e Distribuído) - Faculdade de Informática - Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre/RS, 2012. Disponível em: <http://tede.pucrs.br/tde_busca/arquivo.php?codArquivo=4413>. Acesso em 10 de jun. 2017, às 13h.

FACEBOOK. **Visão geral da Graph API**, 2017. Disponível em: <https://developers.facebook.com/docs/graph-api/overview/>. Acesso em 07 de set. 2017.

FREEMAN, Steve; PRYCE, Nat. **Desenvolvimento de Software Orientado a Objetos, Guiado por Testes**. Rio de Janeiro: Alta Books, 2012. p. 1-10.

GIT. **Primeiros passos:** sobre controle de versão, 2017. Disponível em: <https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Versão>. Acesso em 23 de set. 2017.

GOOGLE DEVELOPERS. **Introdução à Google Maps Android API**, 2017. Disponível em: <https://developers.google.com/maps/documentation/android-api/intro?hl=pt-br>. Acesso em 09 de out. 2017.

GUEDES, Gilleanes T. A. **UML 2: uma abordagem prática**. São Paulo: Novatec Editora, 2009.

IBM. Aspectos fundamentais da computação em nuvem. IBM developerWorks, 13 jan 2011. Disponível em: <https://www.ibm.com/developerworks/br/cloud/library/cl-cloudintro/>. Acesso em 13 de jun. 2017, às 11h.

LAUDON, Kenneth C.; LAUDON, Jane P. **Sistemas de informações gerenciais**. Tradução de Thelma Guimarães. 7ª. ed. São Paulo: Pearson Prentice Hall, 2017. p. 7-9.

MICROSOFT AZURE. **Serviço de aplicativo**, 2017. Disponível em: <https://azure.microsoft.com/pt-br/services/app-service/>. Acesso em 7 de set. 2017.

MICROSOFT AZURE. **Sobre Aplicativos Móveis no Serviço de Aplicativo do Azure**, 2017. Disponível em: <https://docs.microsoft.com/pt-br/azure/app-service-mobile/app-service-mobile-value-prop>. Acesso em 1 de set. 2017.

MICROSOFT AZURE. **Visão geral de aplicativos Web**, 2017. Disponível em: <https://docs.microsoft.com/pt-br/azure/app-service/app-service-web-overview>. Acesso em 1 de set. 2017.

MICROSOFT. CLR (Common Language Runtime):.NET Framework (current version). *In.:* **Microsoft:** developer network, 2017. Disponível em: <https://msdn.microsoft.com/pt-br/library/8bs2ecf4(v=vs.110).aspx>. Acesso em: 16 de maio 2017, às 21h13min.

MICROSOFT. Info: Visão geral do modelo de Code-Behind do ASP.NET. *In.:* **Microsoft:** developer network, 2009. Disponível em: <https://support.microsoft.com/pt-br/help/303247/info-asp.net-code-behind-model-overview>. Acesso em: 16 de maio 2017, às 20h22min.

MICROSOFT. Instalando o.NET Framework:.NET Framework (current version). *In.:* **Microsoft:** developer network, 2017. Disponível em: <https://msdn.microsoft.com/pt-br/library/5a4x27ek(v=vs.110).aspx>. Acesso em: 03 de maio 2017, às 12h32min.

MICROSOFT. Introdução à linguagem C# e ao.NET Framework. *In.:* **Microsoft Docs**, 2017. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>. Acesso em 03 de maio 2017, às 17h30min.

MICROSOFT. O que é IaaS? *In.:* **Microsoft Azure:** Visão geral, 2017. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-iaas/>. Acesso em 13 de jun. 2017, às 14h.

MICROSOFT. O que é o SaaS? *In.:* **Microsoft Azure:** Visão geral, 2017. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-saas/>. Acesso em 13 de jun. 2017, às 14h38min.

MICROSOFT. O que é PaaS? *In.:* **Microsoft Azure:** Visão geral, 2017. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-paas/>. Acesso em 13 de jun. 2017, às 14h15min.

MICROSOFT. Por que o Azure? *In.:* **Microsoft Azure:** O que é o Azure?, 2017. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-azure/>. Acesso em 10 de jun. 2017, às 18h36min.

MICROSOFT. The MVVM Pattern. *In.:* **Microsoft:** developer network, 2012. Disponível em: <https://msdn.microsoft.com/pt-br/library/hh848246(d=printer).aspx>. Acesso em: 15 jun 2017.

MICROSOFT. Um tour pela linguagem C#. *In.:* **Microsoft Docs**, 2016. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/index>. Acesso em 03 de maio 2017, às 17h35min.

MONO. About Mono. *In.:* **Docs**, 2016. Disponível em: <http://www.mono-project.com/docs/about-mono/>. Acesso em 14 de jun. 2017, às 15h.

NOVÁK, I.; *et al*. **Visual Studio 2010 and .NET 4:** six in one. Indianapolis, Indiana, Estados Unidos da América: Wiley Publishing Inc., 2010.

PETZOLD, Charles. **Creating Mobile Apps with Xamarin.Forms:** Cross-platform C# programming for iOS, Android, and Windows, Redmond, 2016. Disponível em: <https://developer.xamarin.com/guides/xamarin-forms/creating-mobile-apps-xamarin-forms/>. Acesso em 10 de maio 2017, às 10h.

PRESSMAN, R. S. **Engenharia de Software**. 6ª. ed. Rio de Janeiro: McGrawHill, 2006. p. 1-50.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software:** uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016. p. 1-13.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª. ed. São Paulo: Pearson Prentice Hall, 2011. p. 2-159.

STAIR, R. M.; REYNOLDS, G. W. **Princípios de sistemas de informação**. Tradução de Harue Avritscher. 9ª. ed. São Paulo: Cengage Learning, 2011. p. 9-13.

TORGERSEN, M. **.NET Blog**, Estados Unidos da América, 9 mar. 2017. Disponível em: <https://blogs.msdn.microsoft.com/dotnet/2017/03/09/new-features-in-c-7-0/#>. Acesso em 03 de maio 2017, às 17h50min.

VISUAL STUDIO. **Visual Studio Team Services:** panejar melhor, codificar junto e enviar mais rápido, 2017. Disponível em: <https://www.visualstudio.com/pt-br/team-services/>. Acesso em 29 de set. 2017.

W3C. Web Application. **Mobile Web Application Best Practices**, 2010. Disponível em: <http://www.w3.org/TR/mwabp/#webapp-defined>. Acesso em 10 de jun. 2017, às 16h26min.

XAMARIN. Introdução ao ciclo de vida de desenvolvimento de software móvel. **Guias:** cross-plataform, 2017. Disponível em: <https://developer.xamarin.com/pt-br/guides/cross-platform/getting_started/>. Acesso em 14 de jun. 2017, às 16:00.

XAMARIN. Introduction to Portable Class Libraries. **Guias:** cross-plataform, 2017. Disponível em: <https://developer.xamarin.com/pt-br/guides/cross-

platform/application_fundamentals/pcl/introduction_to_portable_class_libraries/>. Acesso em 14 de jun. 2017, às 15h10min.

XAMARIN. **Sharing Code Options**, 2017. Disponível em: <https://developer.xamarin.com/pt-br/guides/cross-platform/application_fundamentals/code-sharing/>. Acesso em 14 de jun. 2017, às 15h45min.

XAMARIN. **Xamarin.Forms XAML Basics:** getting started with cross-platform markup for mobile services, 2017. Disponível em: <https://developer.xamarin.com/guides/xamarin-forms/xaml/xaml-basics/>. Acesso em 14 de jun. 2017, às 13h30min.

APÊNDICE A – CONFIGURAÇÃO E UTILIZAÇÃO DO MICROSOFT AZURE

Este apêndice fornecerá as informações para que qualquer desenvolvedor possa estar usufruindo dos mesmos serviços utilizados neste projeto.

Foi optado pelo autor deste trabalho, disponibilizar este Apêndice A, em um repositório público onde qualquer pessoa possa acessar e visualizar. Sendo que poderá ser atualizado para que acompanhe a tecnologia atual. Para acessar este repositório, faça o *download* de um leitor de QR Code e faça a leitura da Figura 28 ou acesse “https://github.com/viavn/diagramastcc/wiki”.

Figura 28 - Acesso ao repositório do Apêndice A.

Fonte: Elaborado pelo autor.