**Instituto Superior de Engenharia de Lisboa**

Modulação e Padrões de Desenho

Ano letivo 2015/2016

Trabalho final

Docente: Luís Falcão Aluno: Iurie Marcinschi Nº 30156

Index

[Scope 3](#_Toc457320179)

[Football App 3](#_Toc457320180)

[Arquitetura - Diagrama de dependências 3](#_Toc457320181)

[Http Server – Controller 4](#_Toc457320182)

[Service 4](#_Toc457320183)

[Domain Mapper 4](#_Toc457320184)

[Soccer Api 4](#_Toc457320185)

[Http Helper 4](#_Toc457320186)

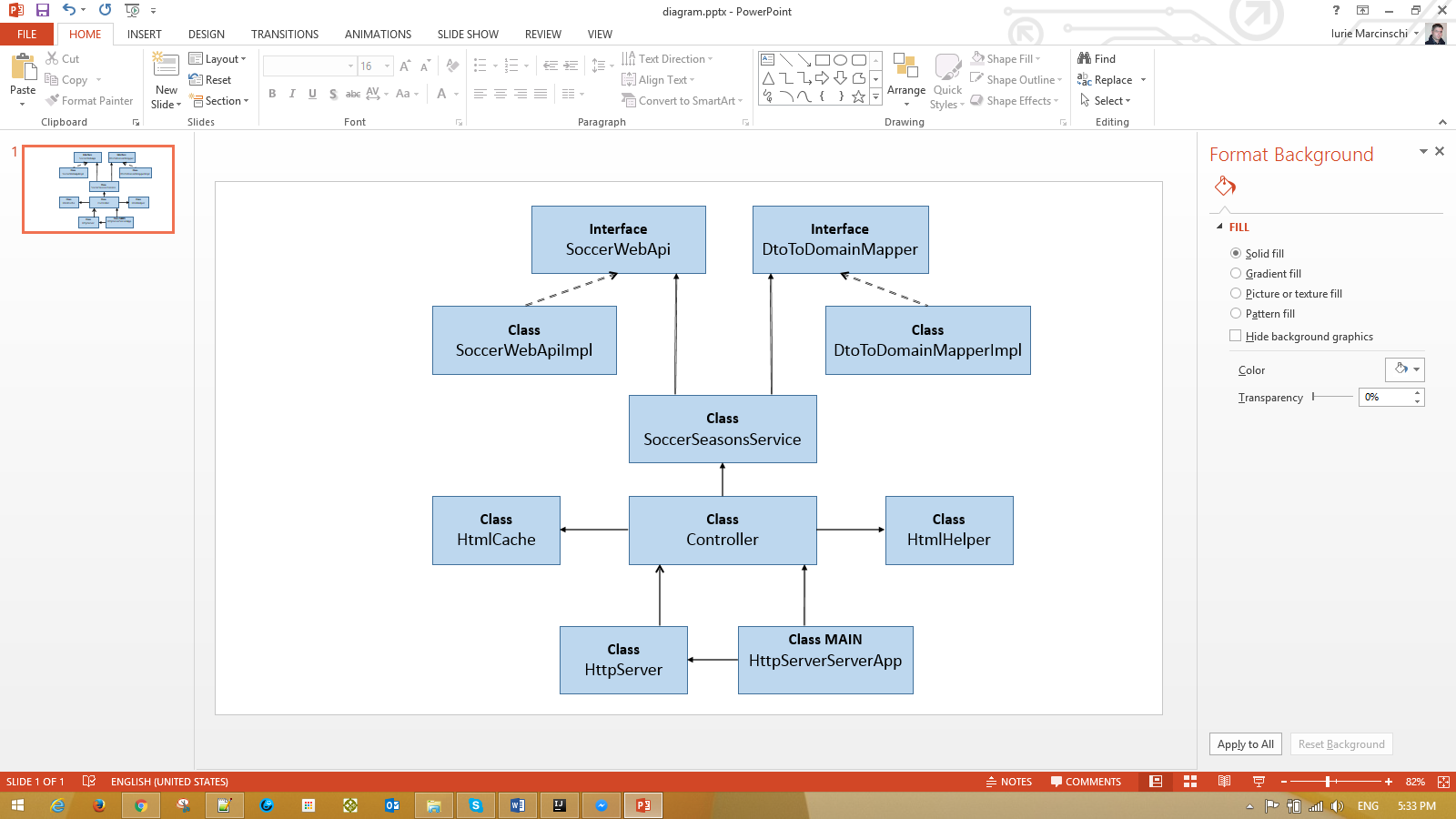
[Cache 4](#_Toc457320187)

# Scope

O objetivo deste trabalho é desenvolver uma aplicação *Web* que fornece informação acerca de ligas de futebol pondo em prática a programação assíncrona e utilização de *Streams* e *API* assíncrona de Java 8.

# Football App

## Arquitetura - Diagrama de dependências



Para diminuir dependências entre entidades fez-se uso de *Function* da *API Java 8*. Desta forma é feito o uso de funções extratores sem ter o vínculo a métodos das entidades.

## Http Server – Controller

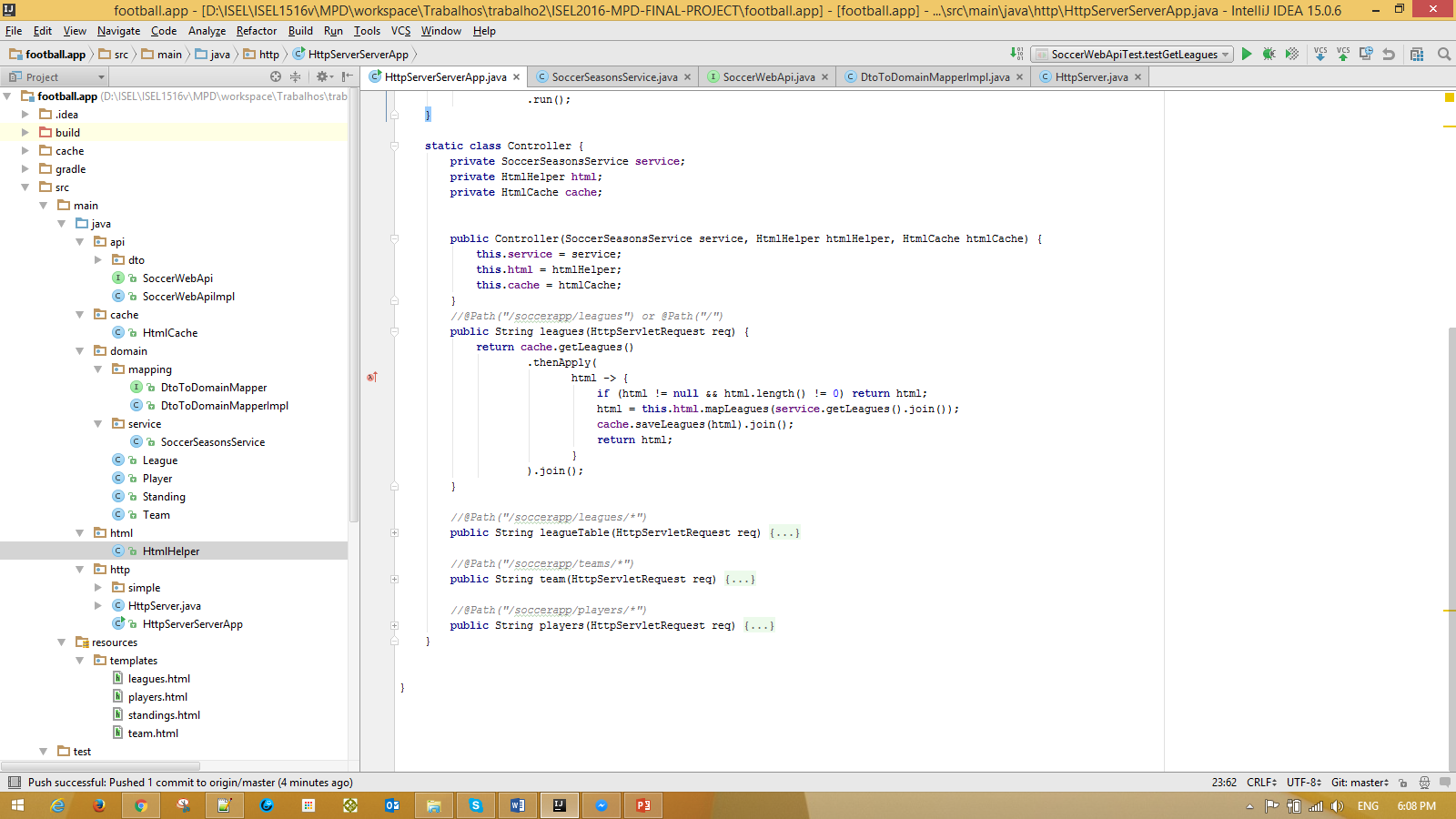
O controlador do servidor para além de resolver os pedidos *Http* trata também de verificar se os recursos solicitados existem em cache (*file system*) em primeiro lugar e se não for o caso solícita o recurso ao serviço responsável no fornecimento do mesmo salvando-o de seguida em cache.

Para tal, o controlador recebe no construtor três entidades:

*HtmlCache.class* – responsável pelo tratamento e fornecimento do conteúdo em cache.

*HtmlHelper.class* – responsável pelo transformação de objetos domínio em representação *Html*.

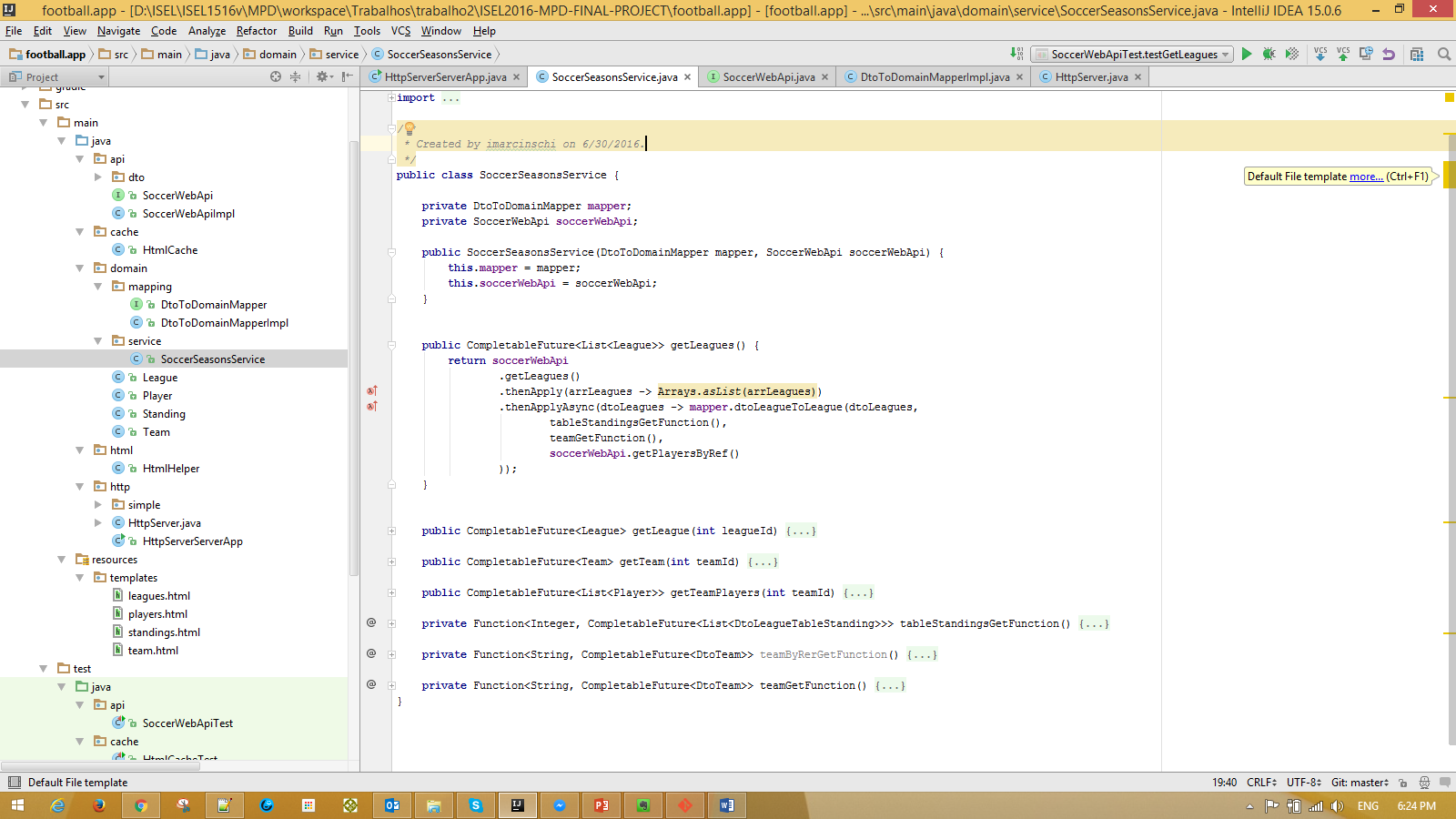
*SoccerSeasonsService.class* – responsável em fornecer os recursos de domínio.



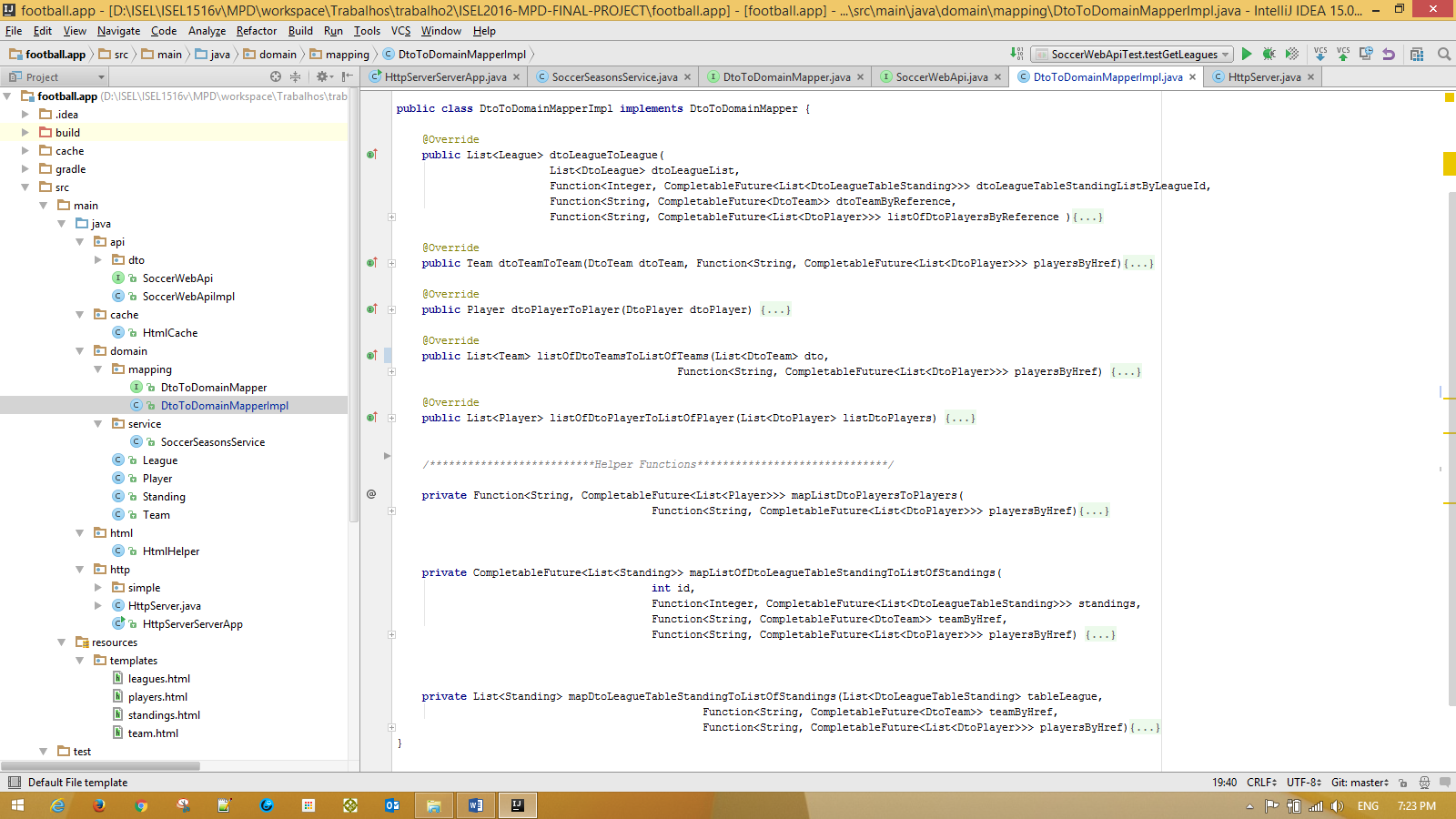
## Service

A entidade *SoccerSeasonsService.class* tem como responsabilidade fornecer objetos domínio solicitados. Para isso na sua instância o serviço recebe no construtor a entidade que representa a interface *API* *Restfull* para obtenção de *DTO’s* e mapia-los para objetos domínio usando a entidade *DtoToDomainMapper.class* também recebida no construtor.

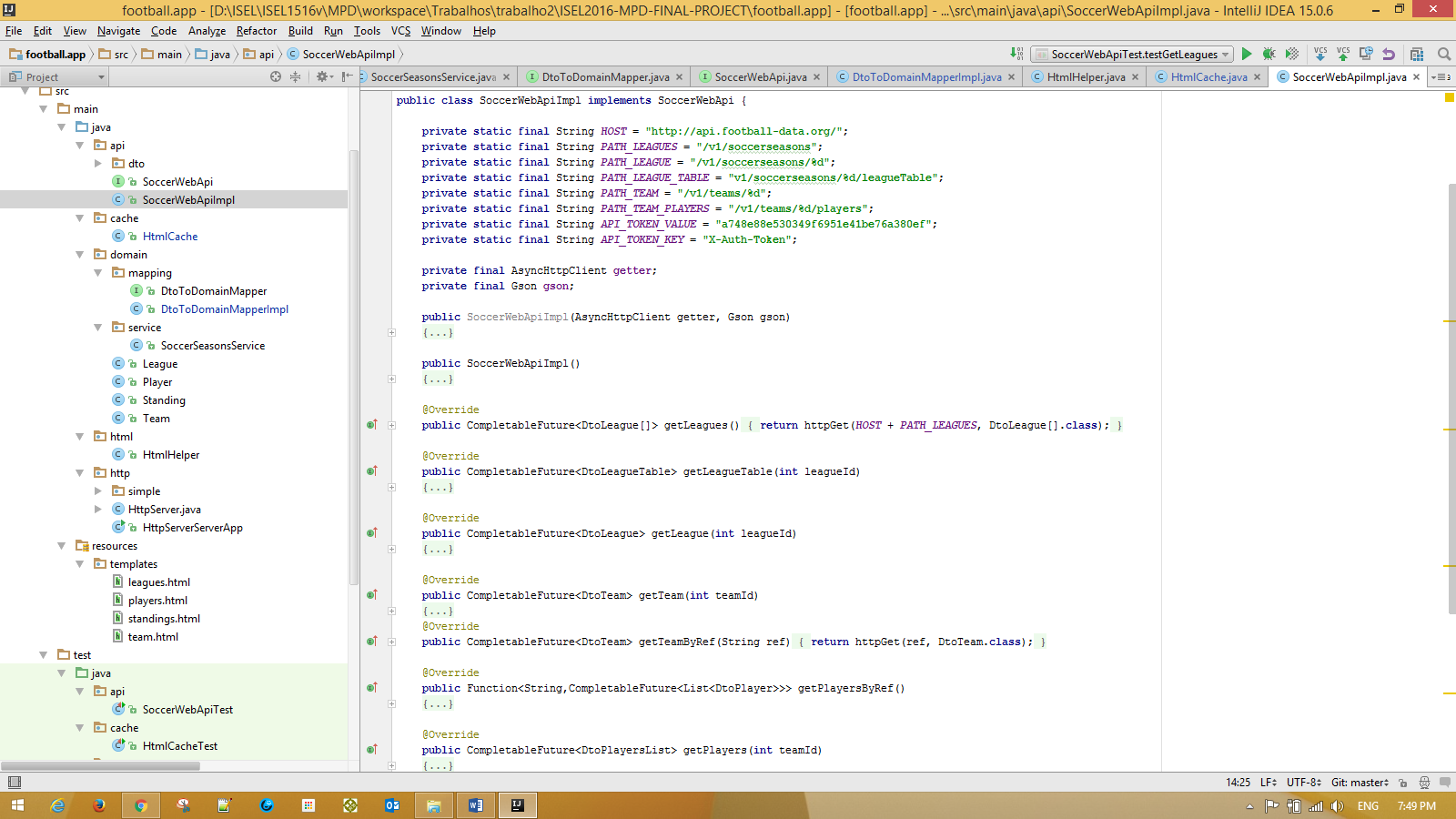
Para garantir que os recursos domínio são instanciados de forma *On Demand,* as funções retornam uma “promessa“ do recurso de forma assíncrona e não bloqueante, utilizando para tal o *CompletableFuture* da *API Java 8.*

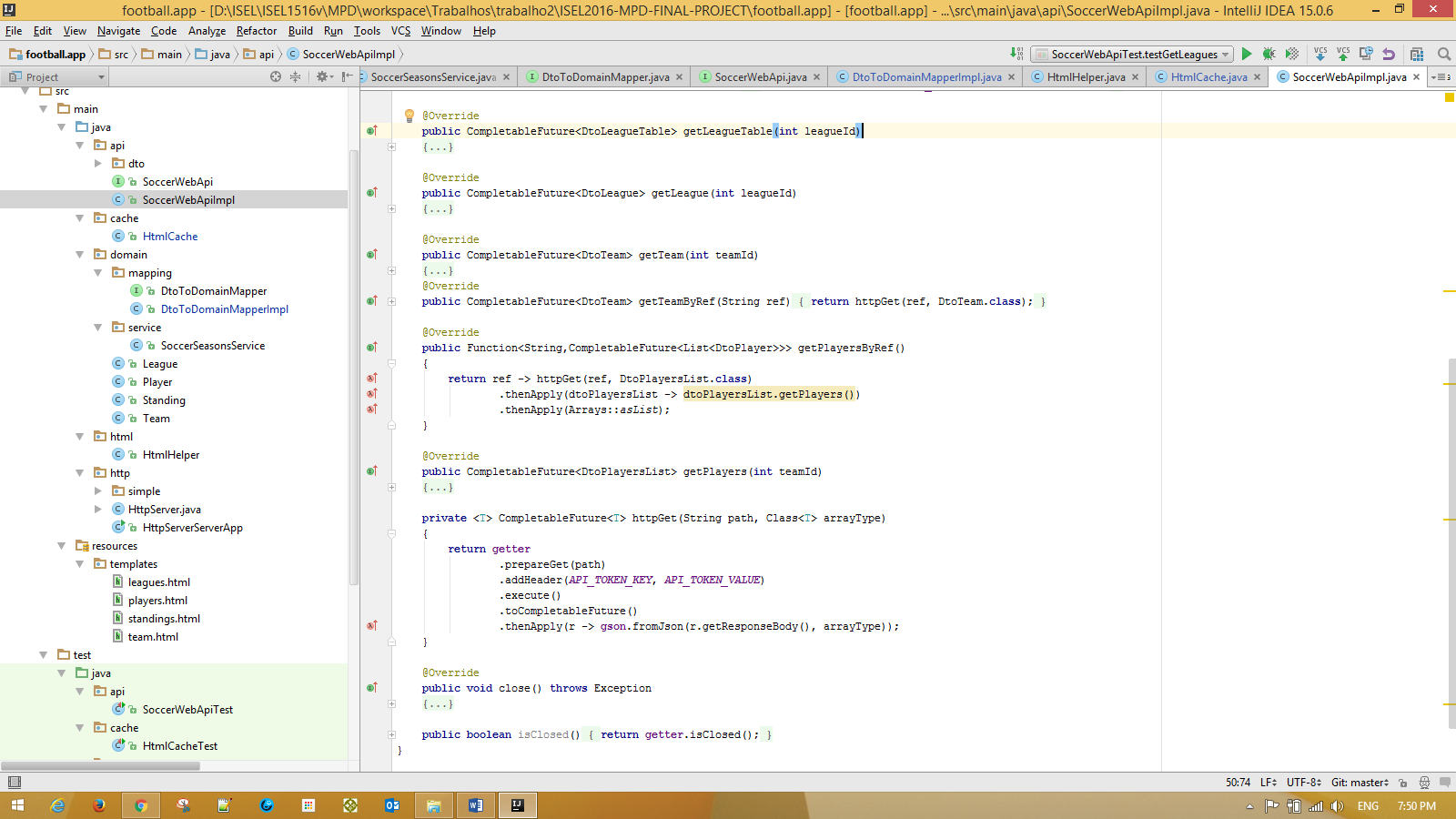


## Domain Mapper

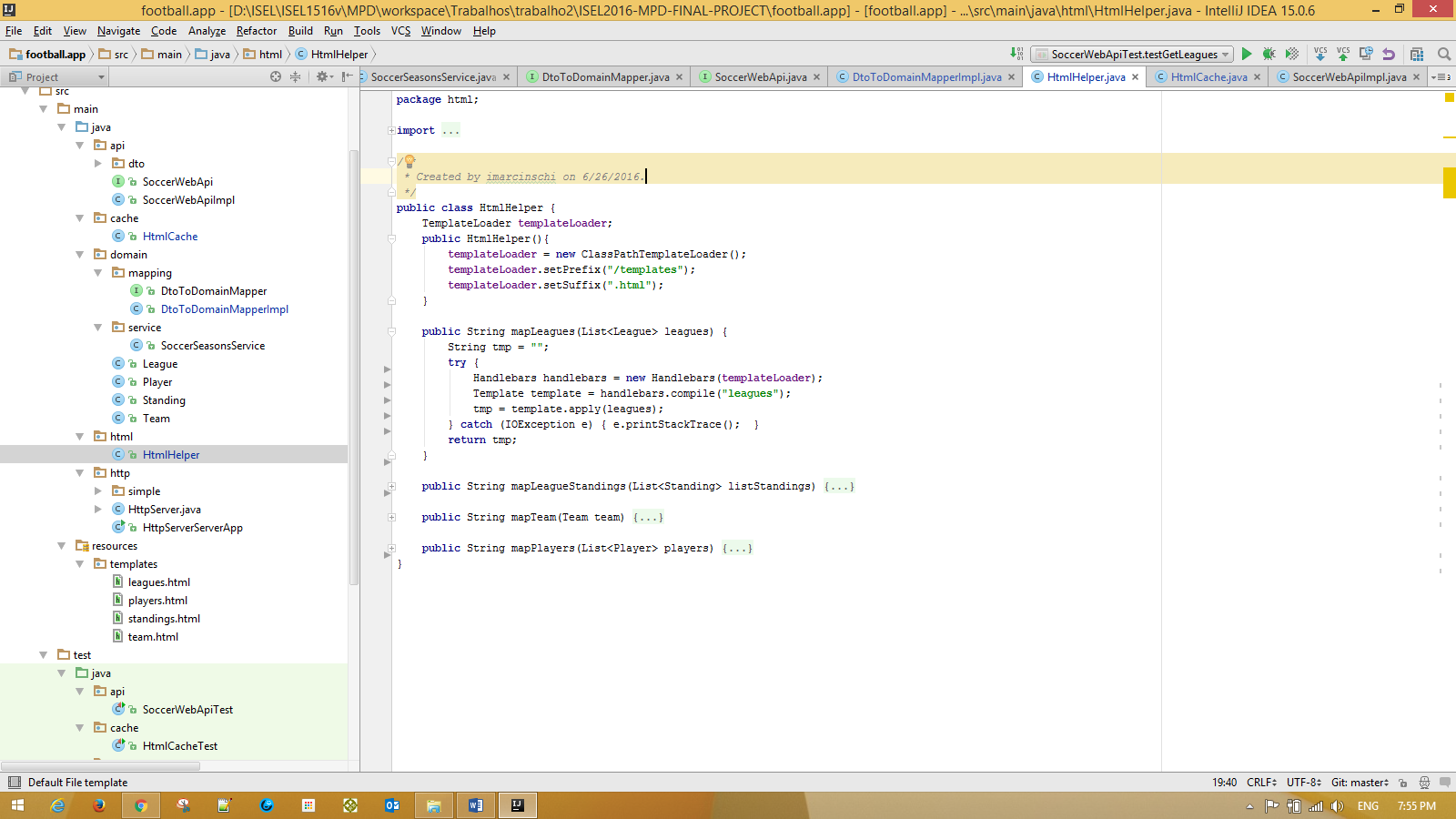


## Soccer Api



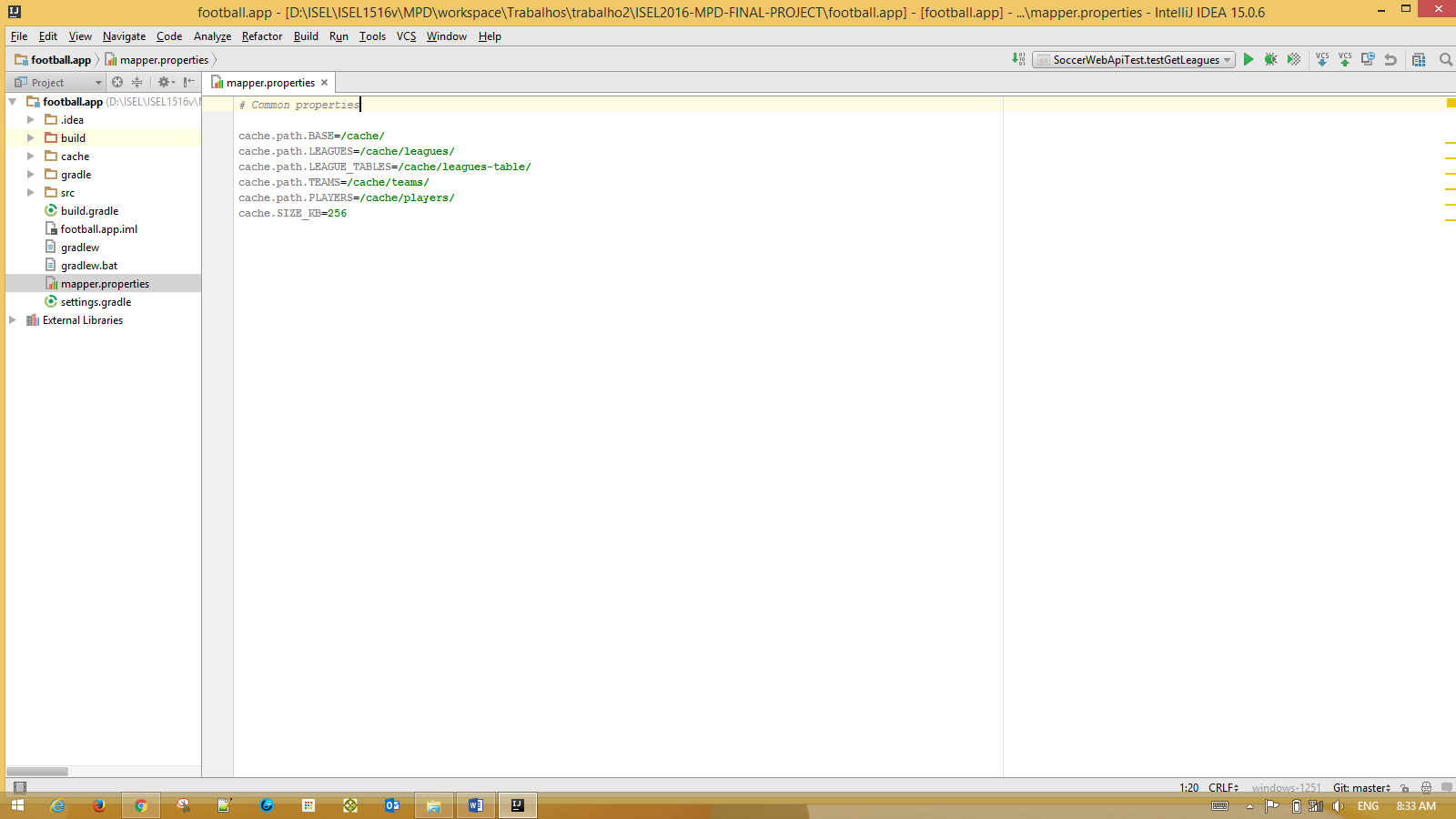


## Http Helper

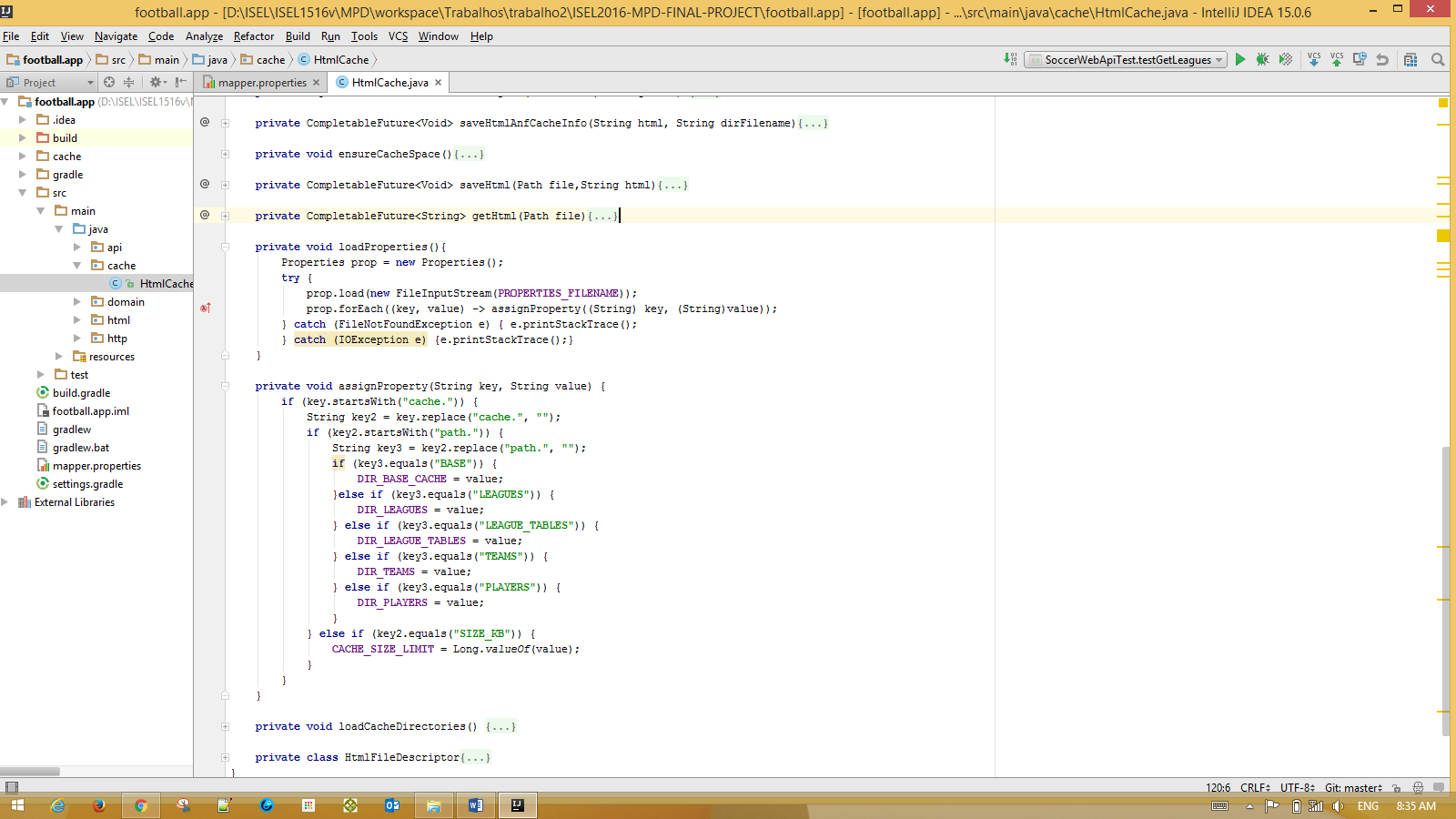


## Cache

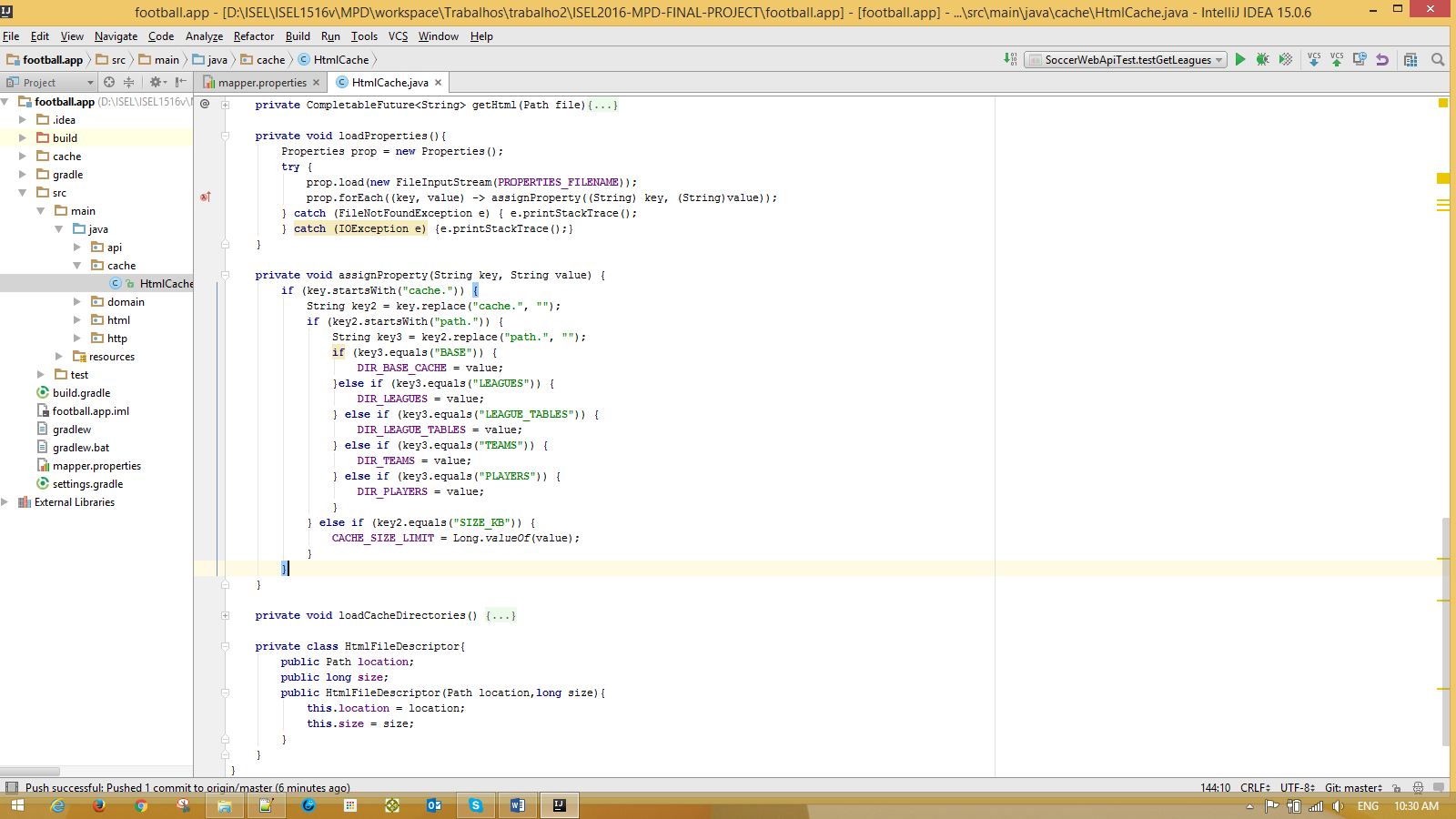
A cache implementada funciona a nivel de filesystem. Os ficheiros html estão a ser guardados em diretorios definidos no ficheiro “*mapper.properties*” onde também define-se o limite em Kb da dimensão da cache.

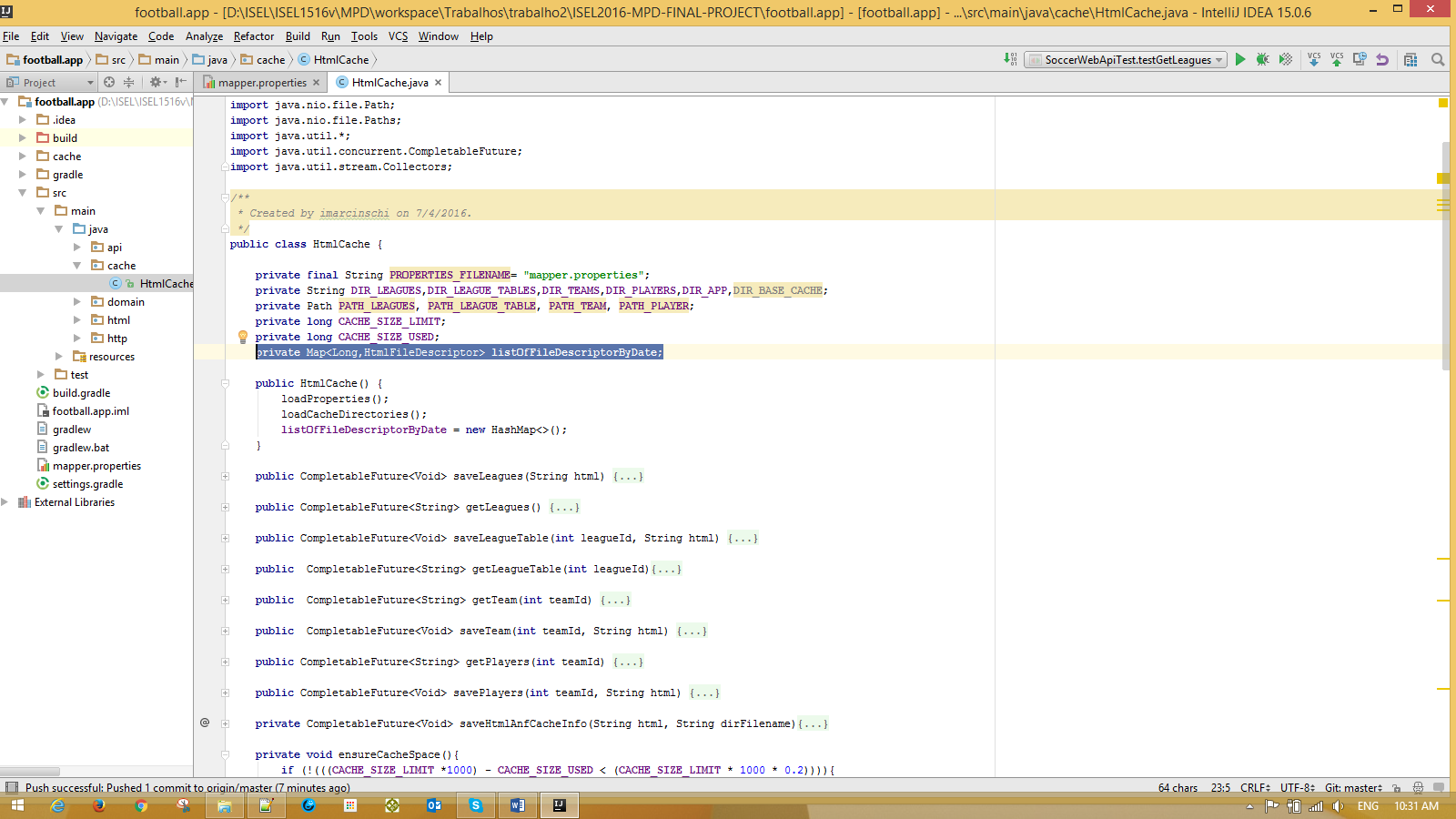


As propriedades são carregadas via methodos *loadProperties* e *assignProperties* utilizando a classe *Properties* de *Java.Utils*. A classe *Properties* representa um conjunto persistente de propriedades em forma de *HashTable*.

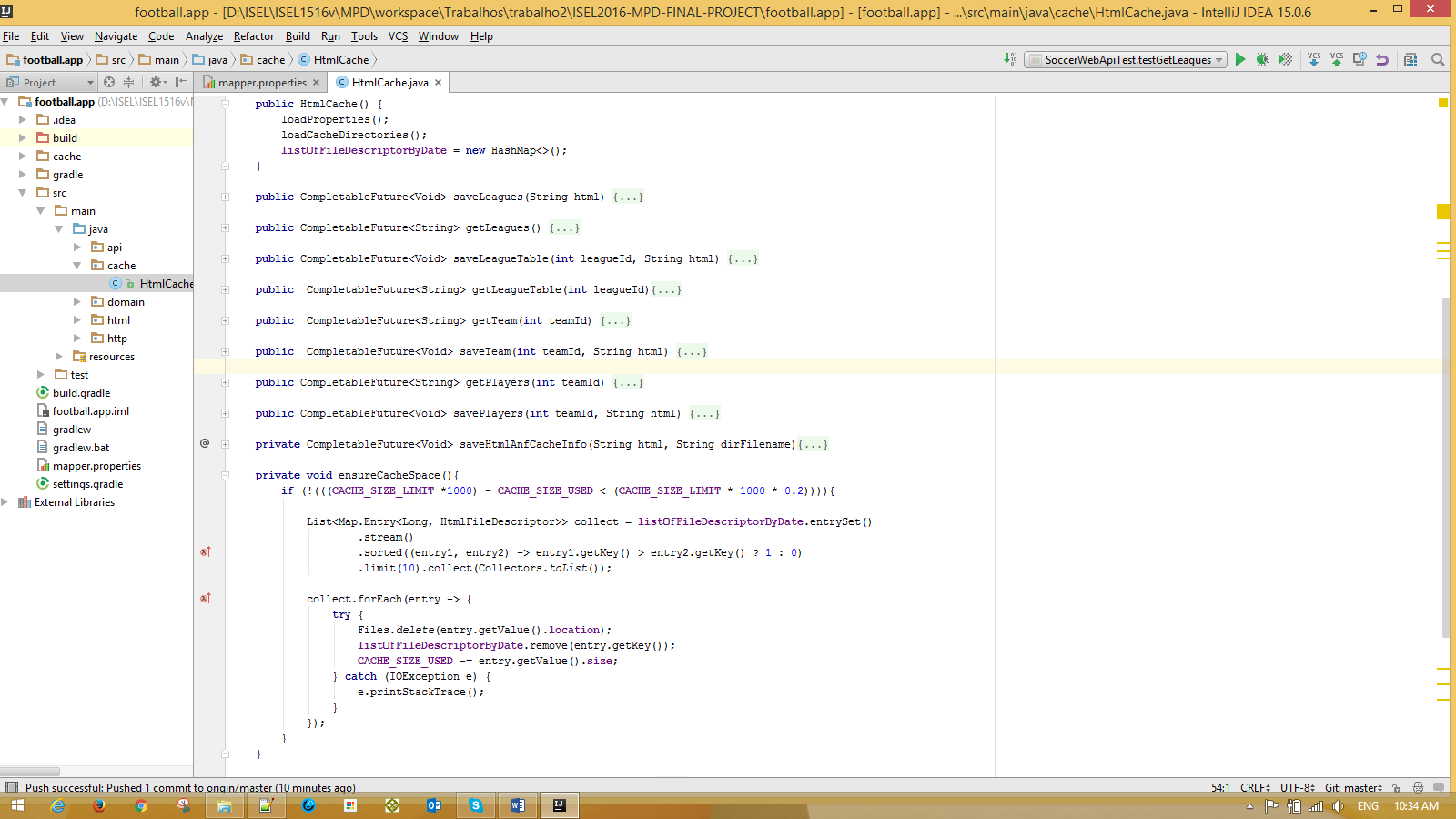


O controlo da dimenção ocupada segue o conceito *FIFO*, para existe um mapa de pares “chave”/”valor” em que a “chave” é o *timestamp* da criação do ficheiro html em filesystem e o “valor” é um objeto do tipo classe *HtmlFileDescriptor* implementada e que contem informação (*Path location*, *size*) acerca o ficheiro *Html.*





O controlo do espa;o ocupado em cache é feito via a função *ensureCacheSpace* que verifica se o espaço ocupado ultrapassa 20% do limite permetido então elimina do *filesystem* os 10 ficheiros html mais antigos.



Esta verificação *ensureCacheSpace* é feita cada vez que é efetuada uma inserção de ficheiros em cache.

Todos os métodos *Getters* e *Setters* da cache invocam as funções *getHtml* e *seveHtml* consecutivamente, passando informação *Path* do ficheiro.

