Università degli studi di Milano Bicocca

Gruppo-BrewDay-2

Progetto-BrewDay-2

Brew Day!

Author:
David Chieregato 806961
Davide Ditolve 806953
Chunhua He 804350
Luca De Filippi 757401

Contents

Abstract

Home brewing, the process of producing beer on a small scale for personal purposes, is an activity that receives growing attention among beer enthusiasts. Every home brewer owns a brewing equipment (kettles, fermenters, pipes, etc.) with a certain maximum brewing capacity, the number of litres the equipment is able to handle in a single "batch". Brewing also requires ingredients, whose actual amounts vary from recipe to recipe; these are various kinds of malts, hops, yeasts and sugars (and of course, water). Brewers like to log their recipes, for future reference, and maintain an updated list of available ingredients, for shopping before the next brew.

The goal of this project is to develop an application for home brewers thats allows them to maintain a list of recipes, and adapt existing ones. The application must also maintain a list of available ingredients, update this list after a batch and when new ingredients are bought, and produce shopping lists for the next batch. A special characteristic of the application is the "what should I brew today?" feature: it goes through the recipes, and taking into account the available ingredients and brewing capacity of the equipment selects a recipe that can be brewed with the available ingredients, maximizing the use of the ingredients, and the batch size. Brew Day! is an application that allows home brewers to maintain an organized database of their beer recipes. The application allows users to create, store and modify recipes, and later on delete them, if the user wishes to do so. The application is intended for "all-grain" brewers only, and thus all recipes are for this kind of brews (the "extract" brews are not supported).

Every home brewer has a specific equipment, whose characteristics leads to a particular "batch size", the maximum number of litres that can be brewed on a single run. Recipes involve, besides water: malts hops yeasts sugars additives While brewers prefer to create recipes referring to concrete values, like kilograms of a specific malt or grams of a specific hops, the application must store these recipes in some "absolute" measure, that allows for a direct conversion of the recipe when the equipment, and consequently the batch size, is updated. For instance, expressing malt quantities as a percentage of the total, and hops as grams per litre of mash, is a possibility.

Besides the actual recipes, the application must maintain recipe instances, i.e., particular brews based on a recipe; these instances can be accompanied by notes to refer to issues that may affect the resulting beer and the brewers would like to keep logged. A particular kind of

note is the tasting notes, that allows brewers to keep track of opinions on a beer from a particular brew.

Besides these more traditional features of Brew Day!, the application maintains a list of available ingredients. This allows brewers to be notified about missing ingredients for the next brew. A recipe instance, i.e., a particular brew, should allow users to update the available ingredients list, subtracting used ingredients from the available ones. Related to this information, Brew Day! must support a useful feature for brewers: "what should I brew today?" goes through the recipes database, and chooses the recipe that maximizes the use of the available ingredients, taking into account the equipment capacity, of course.

The project must implement the features described above, i.e., creation, modification and deletion of recipes, creation of recipe instances (brews), support for notes on brews, and keeping track of available ingredients. The "what should I brew today?" is a mandatory feature. Optionally, developers may choose to allow ingredients availability manually, as opposed to do it automatically from brews information.

The choice of the development platform, including tools and programming languages to use, is left to the teams. The application may be desktop-based, web-based or even tailored for portable devices. Besides the development of the software, teams must provide accompanying documentation, including a requirements document, a design document, and a brief user manual (installation and usage of the application).

1 Assumptions and glossary

We assume that the equipment capacity is referred to a particular recipe, it will not change the order proportion of the ingredients.

2 Requirements

The goal of this project is to develop an application for home brewers thats allows them to maintain a list of recipes, and adapt existing ones. This application is intended for the "all-grain" users only, thus "extract" brews are not supported.

Functional Requirements:

User's account:

- The application allows users to create an account
- The application allows users to login in their account
- Every user have specific equipment which can produce a maximum "batch size"

User's recipe:

- The application allows users to create a recipe
- The application allows users to store a recipe
- The application allows users to update a recipe
- The application allows users to delete a recipe
- The application stores recipe's ingredients in an "absolute" measure; eg. total percentage
- Recipes are made by ingredients: Water, Malts, Hops, Yeasts, Sugars, Additives

Recipes ingredients:

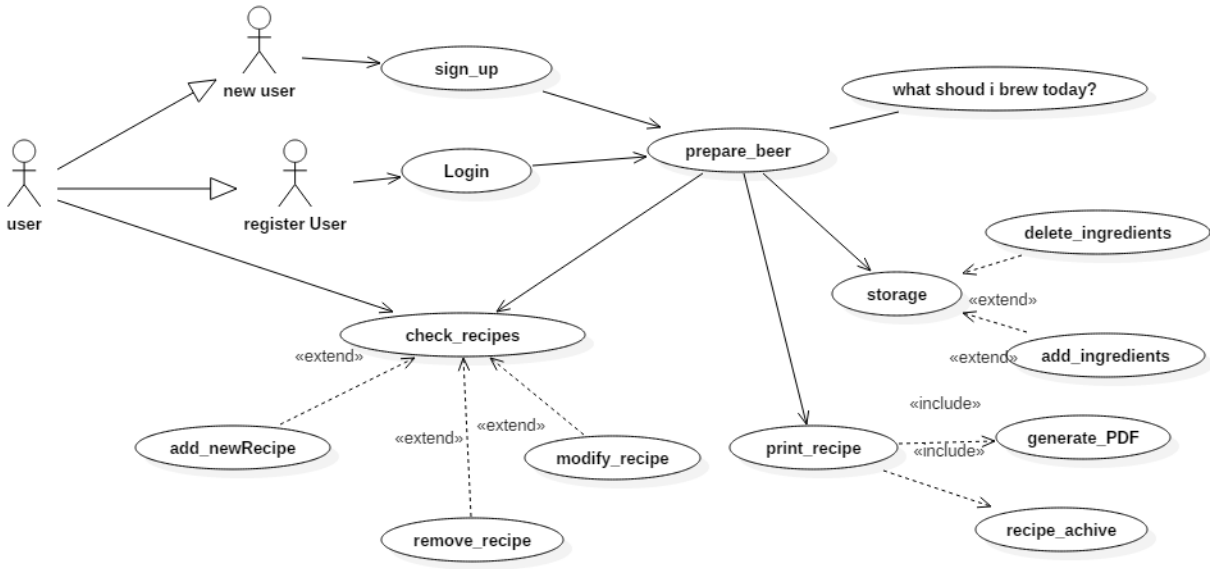
- The application mantain a list of available ingredients
- The application notifies users about missing ingredients
- The application update ingredients's list after a batch
- The application update ingredients's list when new ingredients are bought
- The application produce a shopping list for the next batch

Additional features:

- The application supports notes for brews
- The application has "what should i brew today" feature:
- With the "what should i brew today" feature, the application suggest a recipe made with the available ingredients.
- With the "what should i brew today" feature, the suggested recipe maximizes the usage of available ingredients.
- With the "what should i brew today" feature, the suggested recipe maximizes the batch size.

3 Use Case

3.1 Use Case Diagram



3.2 Use Case Description

What Should I Brew Today?
Portata: Web Page
Livello: Obiettivo Utente
Attore Primario: Utente che desidera creare un tipo di birra
Parti Interessate e Interessi: Sistema
Pre-condizioni: Log-In
Garanzia di successo: Presenza ingredienti nel magazzino
Scenario Principale di Successo:

1. Utente seleziona il "catalogo"
2. Utente seleziona ingrediente
3. Utente accede al catalogo-ricetta

4. Sistema mostra l'elemento selezionato

Scenari alternativi:
Fallimento: il sistema comunica l'errore o "elemento non trovato"
Requisiti speciali: Nessuno
Elenco delle variabili tecnologiche e dei dati: Nessuna
Frequenza di ripetizione: Nessuna

Sign-Up
Attore Primario: Utente che desidera registrarsi
Parti Interessate: Utente, Sistema
Scenario Principale di Successo:

1. Utente fornisce un indirizzo email
2. Utente fornisce un nickname
3. Utente fornisce una password
4. Utente si registra con successo

Scenari alternativi:
Fallimento: Sistema invia un messaggio per invitare utente a riprovare

1. Email già registrata
2. Nickname invalido oppure già presente
3. Password non risponde ai criteri stabiliti dal sistema

Sign-In
Attore Primario: Utente che desidera accedere all'applicazione
Parti Interessate: Utente, Sistema
Scenario Principale di Successo:

1. Utente immette email e password
2. Uente accede all'applicazione

Scenari Alternativi:
Fallimento: Sistema invita a riprovare ad effettuare il log-in

Check-Recipe
Attore Primario: Utente che desidera consultare le ricette
Parti Interessate: Utente, Sistema
Scenario Principale: Utente accede al catalogo delle ricette

Add-New-Recipe
Attore Primario: Utente desidera aggiungere la propria ricetta al catalogo
Parti Interessate: Utente, Sistema
Scenario Principale: Utente crea una nuova ricetta
Scenari Alternativi:
Fallimento: Utente non fornisce una composizione valida

Modify-Recipe
Attore Primario: Utente desidera apportare modifiche a ricette nel catalogo
Parti Interessate: Utente, Sistema
Scenario Principale:

1. Utente modifica uno o più ingrediente presenti in una ricetta
2. Utente modifica la quantità di uno o più ingrediente della ricetta
3. Utente apporta le modifiche

Scenari Alternativi:
Fallimento: Utente apporta modifiche non valide

Delete-Recipe
Attore Primario: Utente che desidera cancellare ricette presenti nel ricettario
Parti Interessate: Utente, Sistema
Scenario Principale: Utente cancella la ricetta

Add-Ingredient
Attore Primario: Utente che aggiunge nuovi ingredienti nel magazzino
Parti Interessate: Utente, Sistema
Scenario Principale:

1. Utente inserisce ingrediente

2. Sistema controlla se ingrediente è presente in magazzino

3. Sistema inserisce ingrediente nel database

Scenari Alternativi:
Fallimento: Sistema informa utente che ingrediente è già presente nel magazzino

1. Ingrediente già presente nel database

Delete-Ingredient
Attore Primario: Utente che desidera cancellare ingredienti dal magazzino
Parti Interessate: Utente, Sistema
Scenario Principale: Sistema cancella ingrediente selezionato dall'utente

Prepare-Beer
Attore Primario: Utente che desidera preparare una birra
Parti Interessate: Utente, Sistema
Scenario Principale:

1. Utente accede alla sezione ricettario

2. Utente inserisce gli ingredienti disponibili

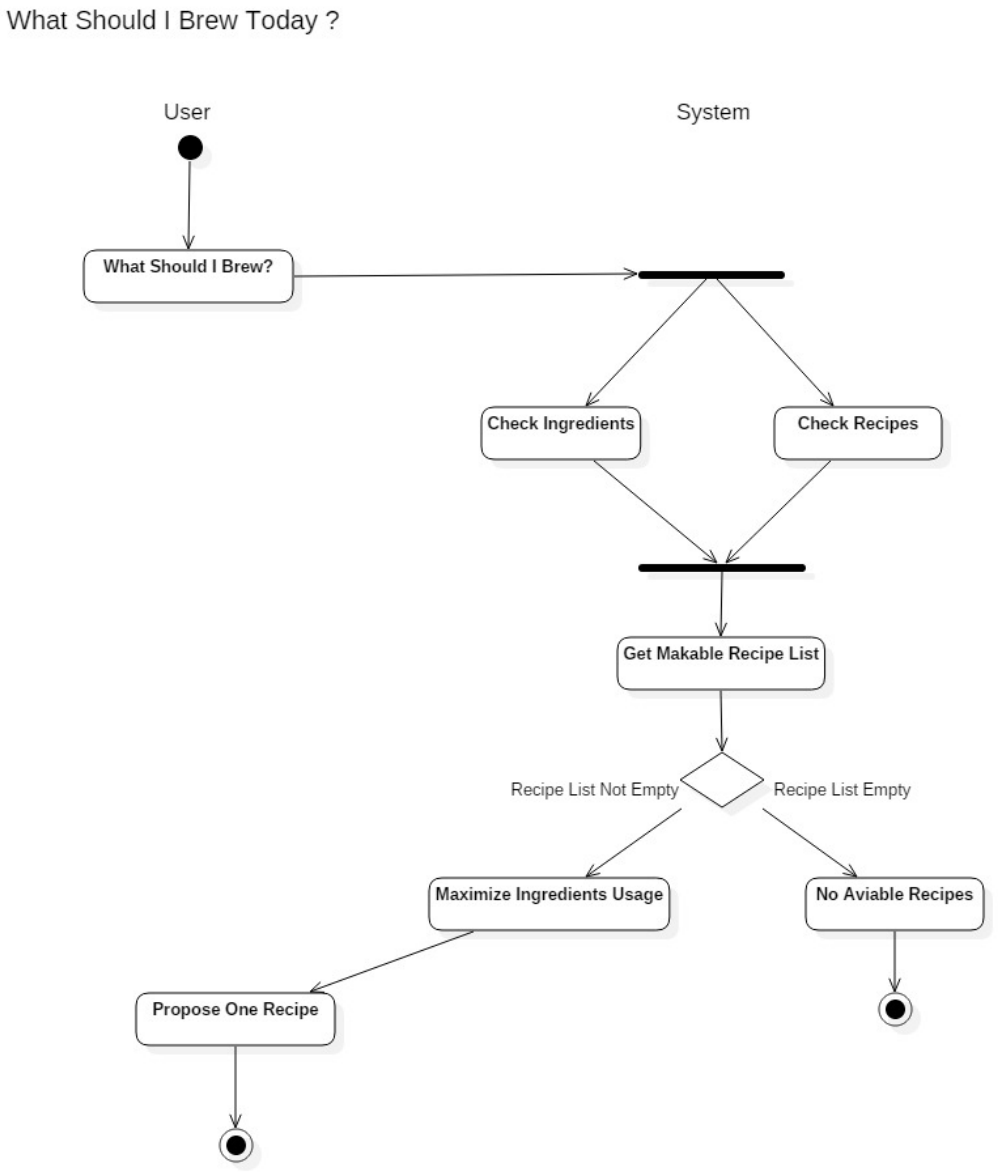
3. Sistema visualizza le possibili ricette che si possono creare

4. Sistema visualizza gli ingredienti mancanti

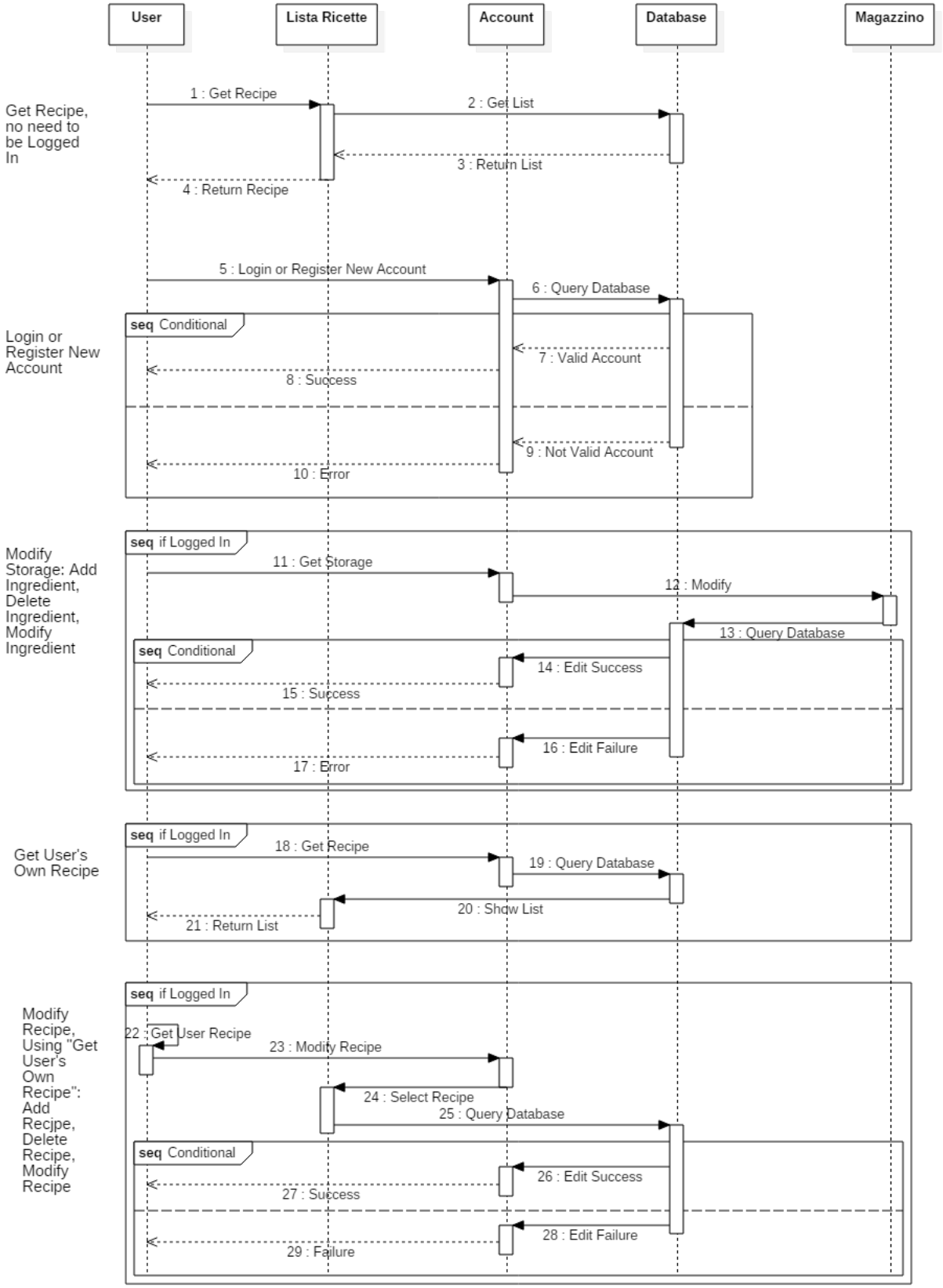
5. Sistema visualizza gli strumenti necessari per preparare la ricetta

4 Activity

We decided to focus on the most complex function to be showed in this Activity Diagram: What Should I Brew Today?



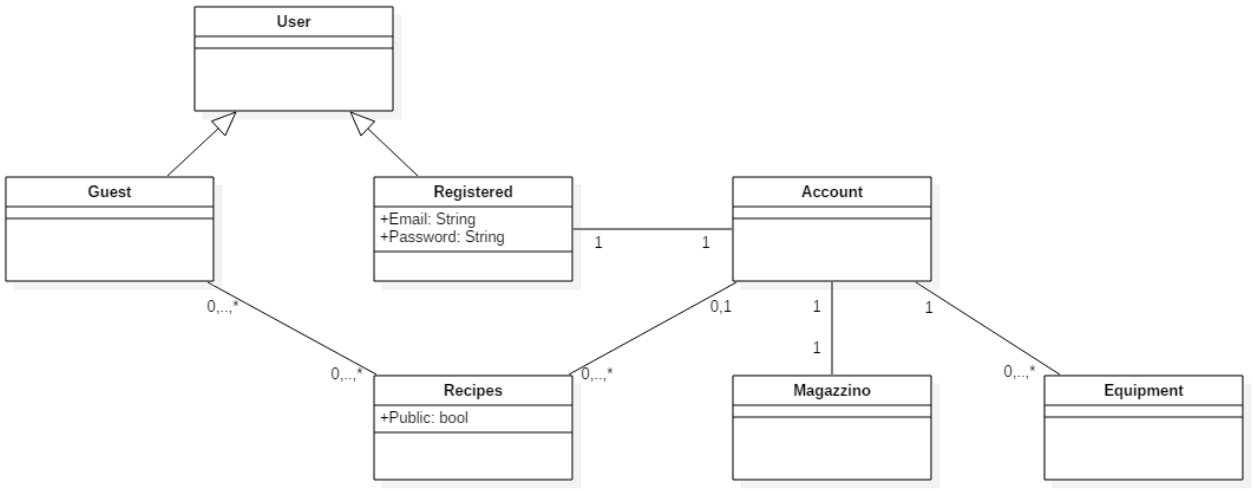
5 Sequence


User
Lista Ricette
Account
Database
Magazzino
Get Recipe, no need to be Logged In
1 : Get Recipe
2 : Get List
3 : Return List
4 : Return Recipe
Login or Register New Account
5 : Login or Register New Account
6 : Query Database
seq Conditional
7 : Valid Account
8 : Success
9 : Not Valid Account
10 : Error
Modify Storage: Add Ingredient, Delete Ingredient, Modify Ingredient
seq if Logged In
11 : Get Storage
12 : Modify
13 : Query Database
seq Conditional
14 : Edit Success
15 : Success
16 : Edit Failure
17 : Error
Get User's Own Recipe
seq if Logged In
18 : Get Recipe
19 : Query Database
20 : Show List
21 : Return List
Modify Recipe, Using "Get User's Own Recipe": Add Recjpe, Delete Recipe, Modify Recipe
seq if Logged In
22 : Get User Recipe
23 : Modify Recipe
24 : Select Recipe
25 : Query Database
seq Conditional
26 : Edit Success
27 : Success
28 : Edit Failure
29 : Failure

In these Sequence Diagrams is evidenced the flow of information from one actor to another.
To be noticed that every action made by the User is always filtered by a Controller, Lista Ricette or Account, which makes the actual query to the needed database.

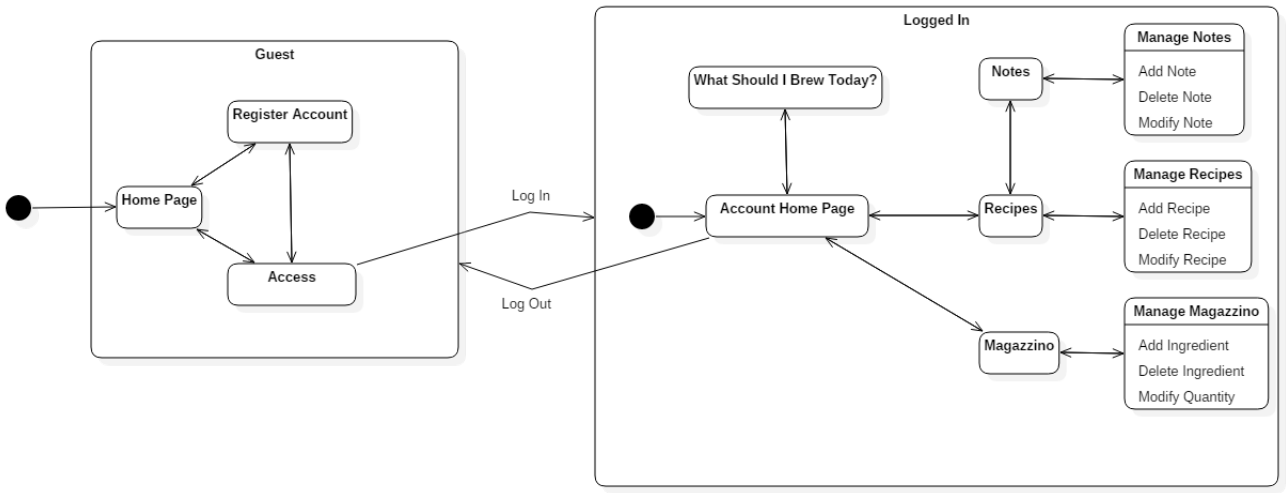
6 Domain

The following diagram express a view of the BrewDay! internal structure and hierarchy.
On this diagram is evidenced the needed information to log in and the dual nature of the Recipes class: Public and Private, which is defined by a boolean attribute.



7 Statechart

The following diagram shows the operations that can be made in every state. Each one of this States represent a view.



8 Implementation Choices

8.1 DAO Pattern

In web programming, the Data Access Object Pattern is an Architectural Pattern used to manage Persistence:
it's a class with its methods that represents a table entity of a Relational Database Management System.
Mainly used to stratify and isolate the access to a table by queries, placed in class methods, hence creating a separation from data layer to business logic and making the code more abstract and maintainable.

Advantages:

- Makes a rigid separation from application components, like *Model* and *Control* of a MVC based application

8.2 MVC Pattern

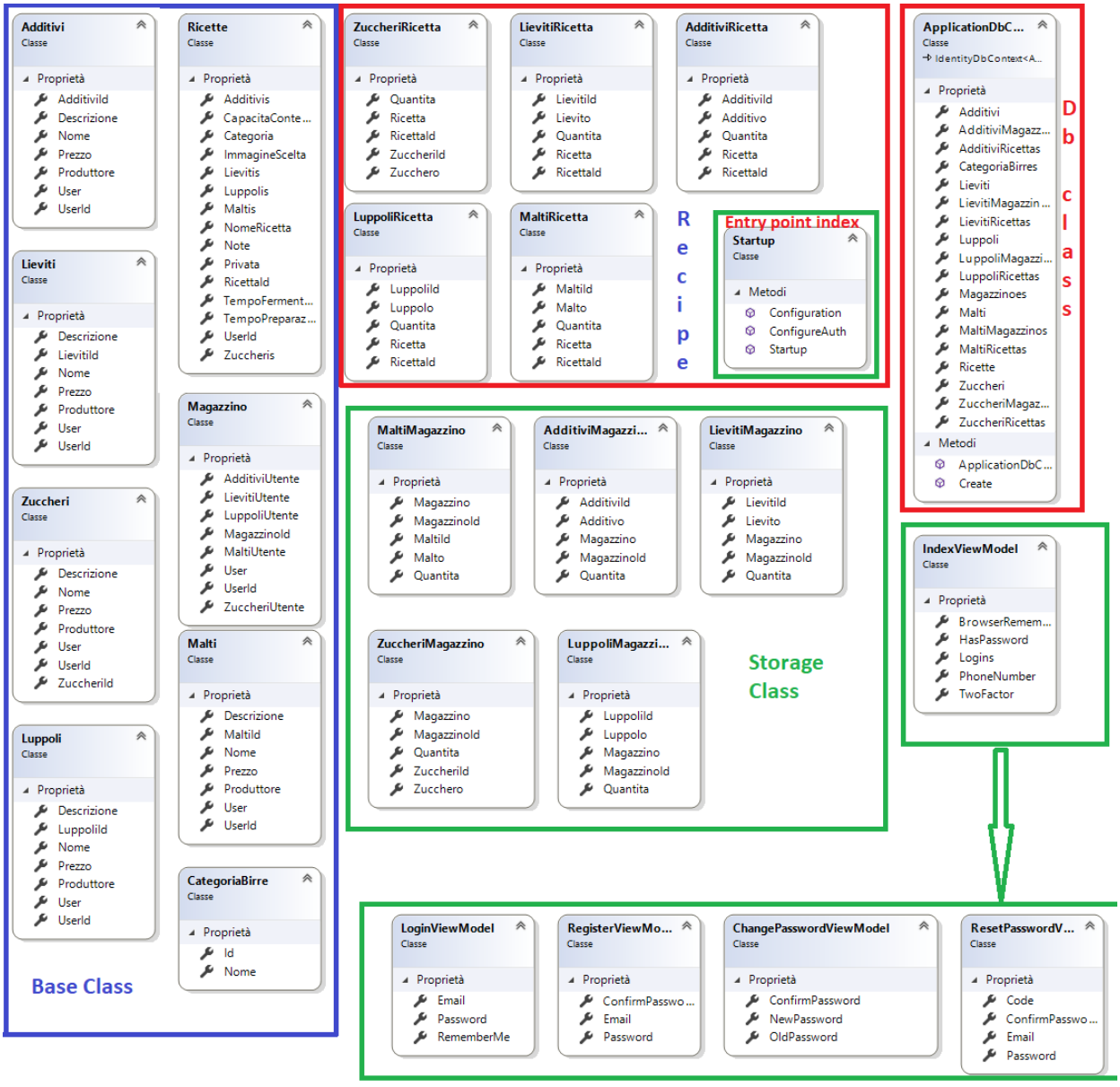
MVC Pattern stands for Model-View-Controller Pattern. It's an Architectural Pattern made by three sections.
The Model catch the application behavior, directly handling data, logic and application's rules.
A View is every output representation of information.
The Controller accept a input and translate in commands for the model or the view.

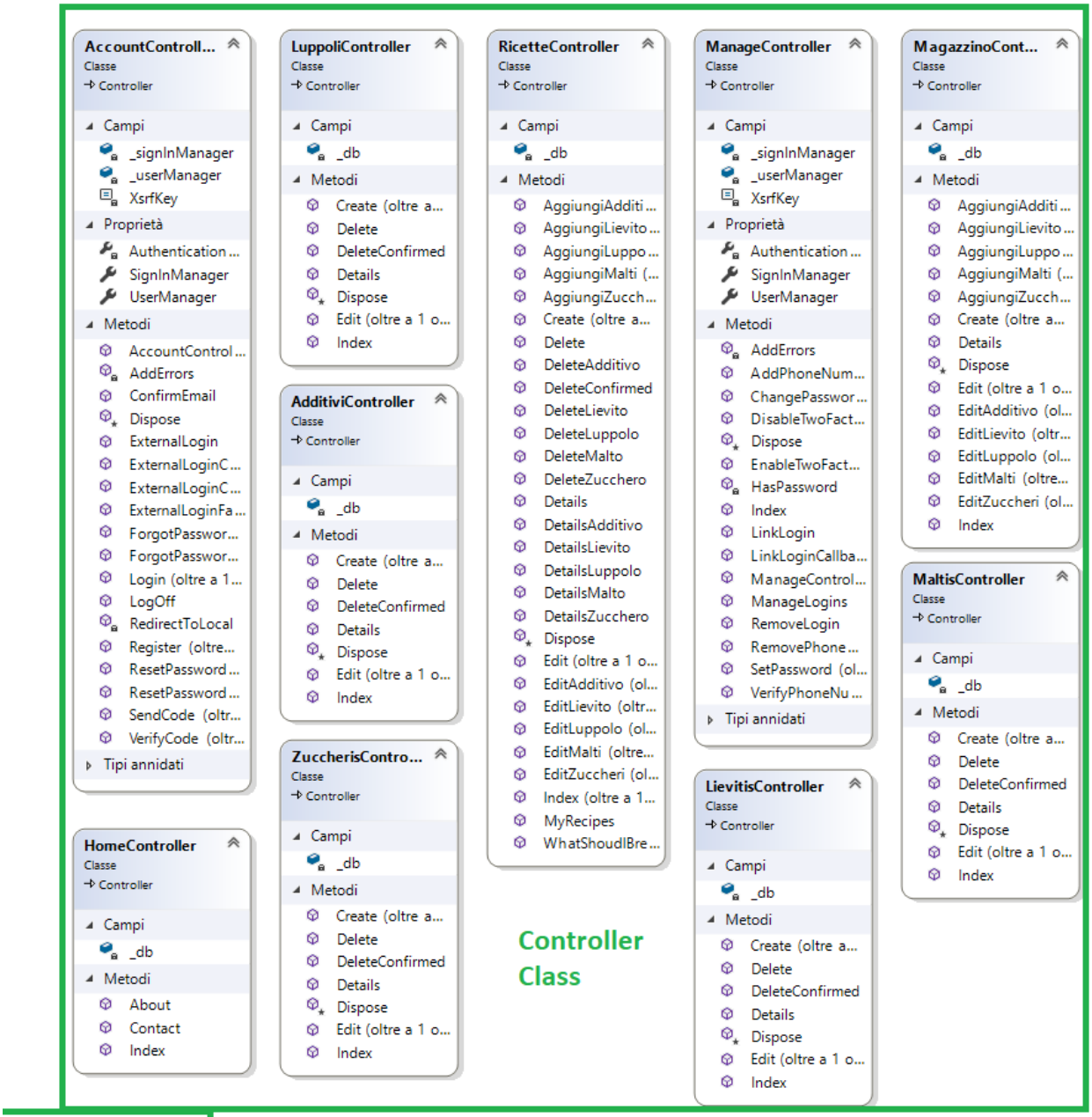
8.3 Row Data Gateway Pattern: LINQ

LINQ means language-integrated query: We use the term language-integrated query to indicate that query is an integrated feature of the developer's primary programming languages (for example, Visual C#, Visual Basic). Language-integrated query allows query expressions to benefit from the rich metadata, compile-time syntax checking, static typing and IntelliSense that was previously available only to imperative code. Language-integrated query also allows a single general purpose declarative query facility to be applied to all in-memory information, not just information from external sources. Exactly the Row Data Gateway definition: An object that acts as a Gateway to a single record in a data source. There is one instance per row.

9 Classes

In a web MVC application there aren't a lot of links between the different classes because this structural pattern is based on the separation of unitary programming logic pages that interact after user requests. There are some auto generated classes in the project that won't be taken into account in the following diagram in the next page. The diagram is split into 2 pages in order to get a better readability.

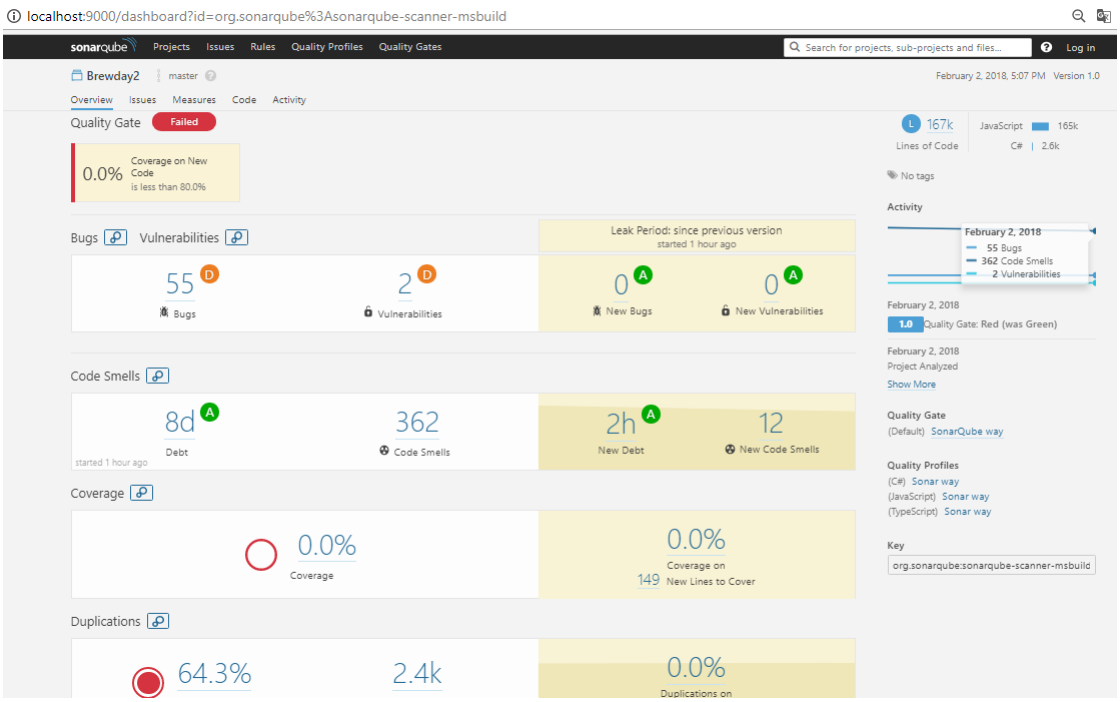

Additivi
Classe
Proprietà
AdditiviId
Descrizione
Nome
Prezzo
Produttore
User
UserId
Lieviti
Classe
Proprietà
Descrizione
LievitiId
Nome
Prezzo
Produttore
User
UserId
Zuccheri
Classe
Proprietà
Descrizione
Nome
Prezzo
Produttore
User
UserId
ZuccheriId
Luppoli
Classe
Proprietà
Descrizione
LuppoliId
Nome
Prezzo
Produttore
User
UserId
Base Class
Ricette
Classe
Proprietà
Additivis
CapacitaConte...
Categoria
ImmagineScelta
Lievitis
Luppolis
Maltis
NomeRicetta
Note
Privata
RicettaId
TempoFerment...
TempoPreparaz...
UserId
Zuccheris
Magazzino
Classe
Proprietà
AdditiviUtente
LievitiUtente
LuppoliUtente
MagazzinoId
MaltiUtente
User
UserId
ZuccheriUtente
Malti
Classe
Proprietà
Descrizione
MaltiId
Nome
Prezzo
Produttore
User
UserId
CategoriaBirre
Classe
Proprietà
Id
Nome
ZuccheriRicetta
Classe
Proprietà
Quantita
Ricetta
RicettaId
ZuccheriId
Zucchero
LievitiRicetta
Classe
Proprietà
LievitiId
Lievito
Quantita
Ricetta
RicettaId
AdditiviRicetta
Classe
Proprietà
AdditiviId
Additivo
Quantita
Ricetta
RicettaId
LuppoliRicetta
Classe
Proprietà
LuppoliId
Luppolo
Quantita
Ricetta
RicettaId
MaltiRicetta
Classe
Proprietà
MaltiId
Malto
Quantita
Ricetta
RicettaId
Recipe
Entry point index
Startup
Classe
Metodi
Configuration
ConfigureAuth
Startup
ApplicationDbC...
Classe
IdentityDbContext<A...
Proprietà
Additivi
AdditiviMagazz...
AdditiviRicettas
CategoriaBirres
Lieviti
LievitiMagazzin...
LievitiRicettas
Luppoli
LuppoliMagazzi...
LuppoliRicettas
Magazzinoes
Malti
MaltiMagazzinos
MaltiRicettas
Ricette
Zuccheri
ZuccheriMagaz...
ZuccheriRicettas
Metodi
ApplicationDbC...
Create
Db class
MaltiMagazzino
Classe
Proprietà
Magazzino
MagazzinoId
MaltiId
Malto
Quantita
AdditiviMagazzi...
Classe
Proprietà
AdditiviId
Additivo
Magazzino
MagazzinoId
Quantita
LievitiMagazzino
Classe
Proprietà
LievitiId
Lievito
Magazzino
MagazzinoId
Quantita
ZuccheriMagazzino
Classe
Proprietà
Magazzino
MagazzinoId
Quantita
ZuccheriId
Zucchero
LuppoliMagazzi...
Classe
Proprietà
LuppoliId
Luppolo
Magazzino
MagazzinoId
Quantita
Storage Class
IndexViewModel
Classe
Proprietà
BrowserRemem...
HasPassword
Logins
PhoneNumber
TwoFactor
LoginViewModel
Classe
Proprietà
Email
Password
RememberMe
RegisterViewMo...
Classe
Proprietà
ConfirmPasswo...
Email
Password
ChangePasswordViewModel
Classe
Proprietà
ConfirmPassword
NewPassword
OldPassword
ResetPasswordV...
Classe
Proprietà
Code
ConfirmPasswo...
Email
Password


AccountControll...
Classe
Controller
Campi
_signInManager
_userManager
XsrfKey
Proprietà
Authentication ...
SignInManager
UserManager
Metodi
AccountControl ...
AddErrors
ConfirmEmail
Dispose
ExternalLogin
ExternalLoginC ...
ExternalLoginC ...
ExternalLoginFa ...
ForgotPasswor...
ForgotPasswor...
Login (oltre a 1...
LogOff
RedirectToLocal
Register (oltre...
ResetPassword...
ResetPassword...
SendCode (oltr...
VerifyCode (oltr...
Tipi annidati
LuppoliController
Classe
Controller
Campi
_db
Metodi
Create (oltre a...
Delete
DeleteConfirmed
Details
Dispose
Edit (oltre a 1 o...
Index
AdditiviController
Classe
Controller
Campi
_db
Metodi
Create (oltre a...
Delete
DeleteConfirmed
Details
Dispose
Edit (oltre a 1 o...
Index
ZuccherisContro...
Classe
Controller
Campi
_db
Metodi
Create (oltre a...
Delete
DeleteConfirmed
Details
Dispose
Edit (oltre a 1 o...
Index
RicetteController
Classe
Controller
Campi
_db
Metodi
AggiungiAdditi...
AggiungiLievito...
AggiungiLuppo...
AggiungiMalti (...
AggiungiZucch...
Create (oltre a...
Delete
DeleteAdditivo
DeleteConfirmed
DeleteLievito
DeleteLuppolo
DeleteMalto
DeleteZucchero
Details
DetailsAdditivo
DetailsLievito
DetailsLuppolo
DetailsMalto
DetailsZucchero
Dispose
Edit (oltre a 1 o...
EditAdditivo (ol...
EditLievito (oltr...
EditLuppolo (ol...
EditMalti (oltre...
EditZuccheri (ol...
Index (oltre a 1...
MyRecipes
WhatShoudIBre...
ManageController
Classe
Controller
Campi
_signInManager
_userManager
XsrfKey
Proprietà
Authentication ...
SignInManager
UserManager
Metodi
AddErrors
AddPhoneNum...
ChangePasswor...
DisableTwoFact...
Dispose
EnableTwoFact...
HasPassword
Index
LinkLogin
LinkLoginCallba...
ManageControl...
ManageLogins
RemoveLogin
RemovePhone...
SetPassword (ol...
VerifyPhoneNu...
Tipi annidati
LievitisController
Classe
Controller
Campi
_db
Metodi
Create (oltre a...
Delete
DeleteConfirmed
Details
Dispose
Edit (oltre a 1 o...
Index
MagazzinoCont...
Classe
Controller
Campi
_db
Metodi
AggiungiAdditi...
AggiungiLievito...
AggiungiLuppo...
AggiungiMalti (...
AggiungiZucch...
Create (oltre a...
Details
Dispose
Edit (oltre a 1 o...
EditAdditivo (ol...
EditLievito (oltr...
EditLuppolo (ol...
EditMalti (oltre...
EditZuccheri (ol...
Index
MaltisController
Classe
Controller
Campi
_db
Metodi
Create (oltre a...
Delete
DeleteConfirmed
Details
Dispose
Edit (oltre a 1 o...
Index
HomeController
Classe
Controller
Campi
_db
Metodi
About
Contact
Index
Controller Class
ForgotPassword...
Classe
Proprietà
Email
Account class

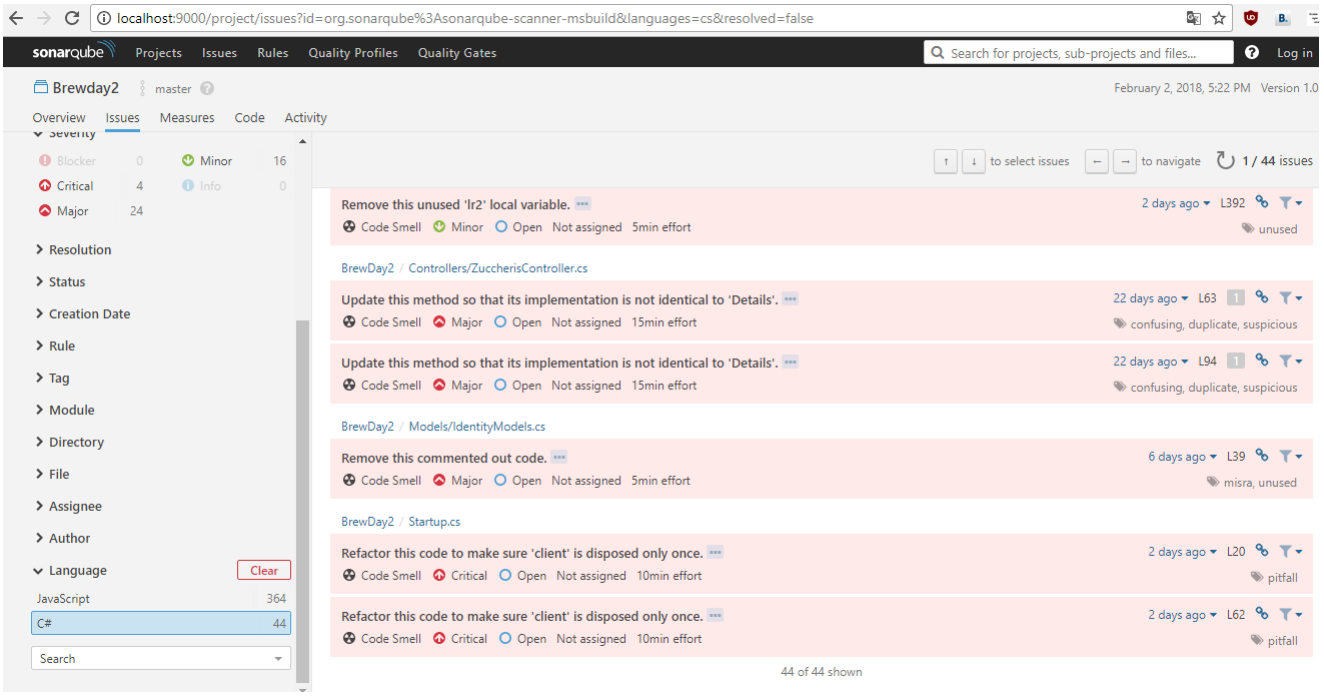
10 Test

10.1 SonarQube

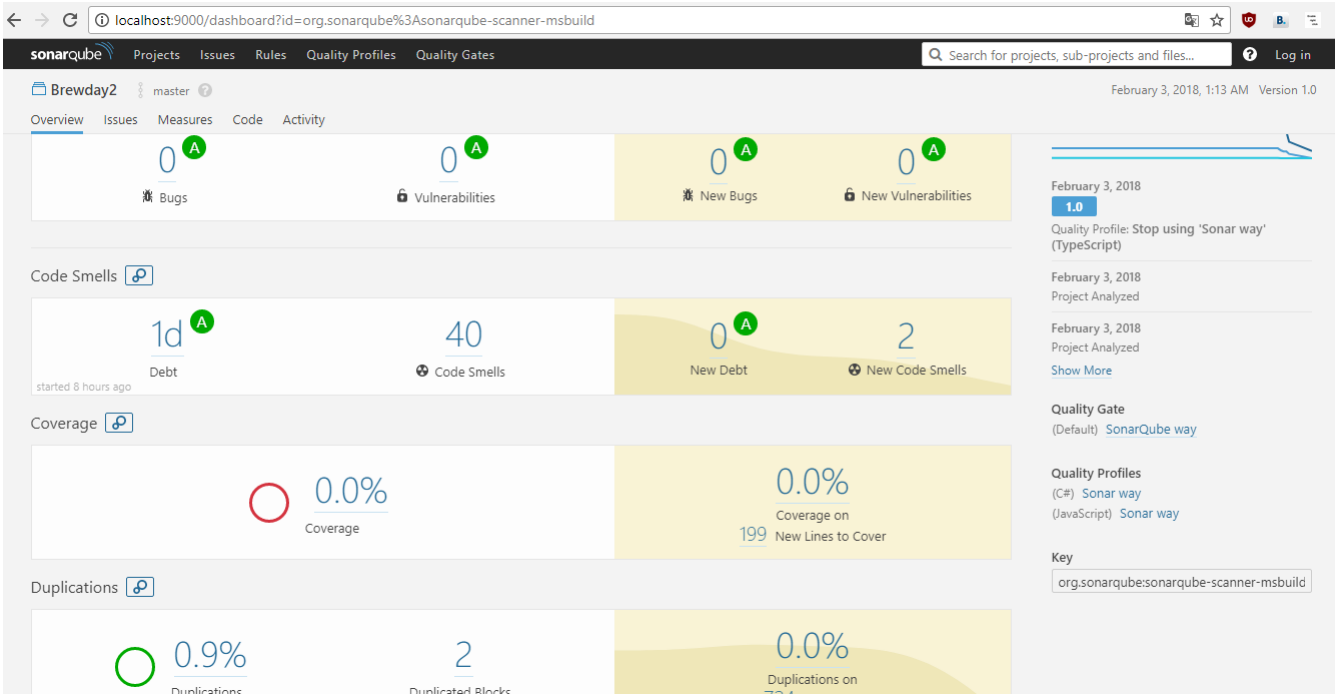
After a first execution of the SonarQube test engine we noticed that the worst code quality issue where related to Jquery javascript plugin.
Jquery is a rewrite engine, so it's normal that the code quality analysis result is not good: it's creating a different way to use the javascript language so it needs a different kind of language test.



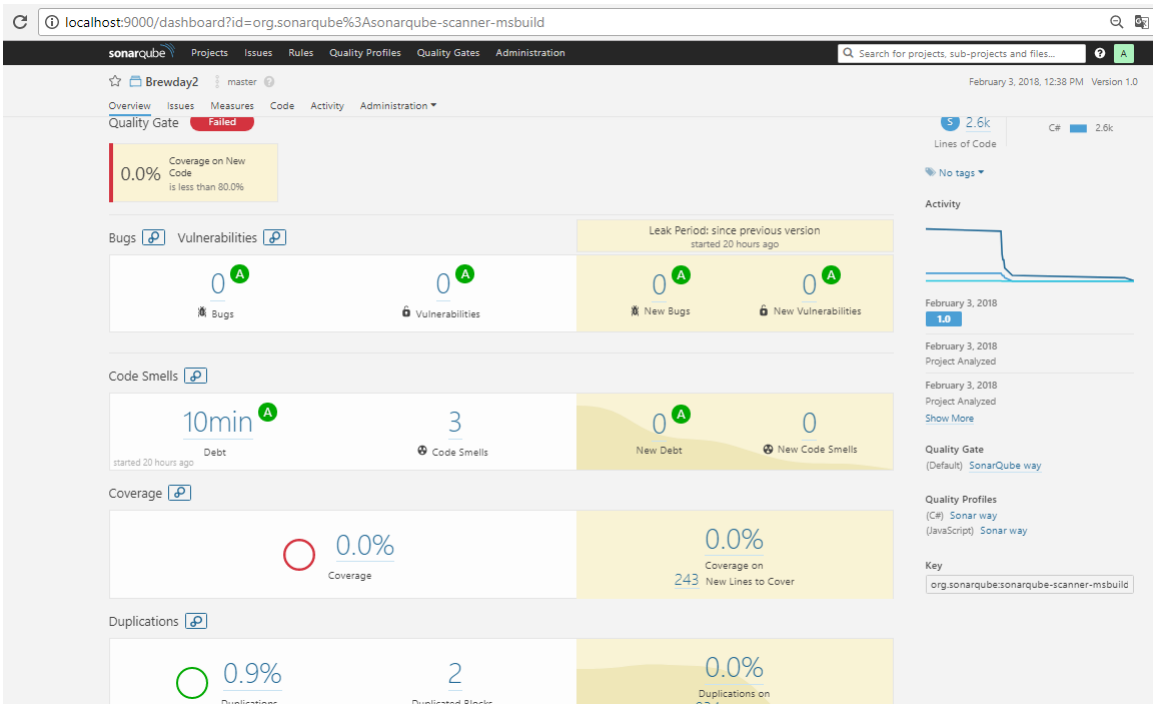
So we concentrated on the C# issues exposed by SonarQube.



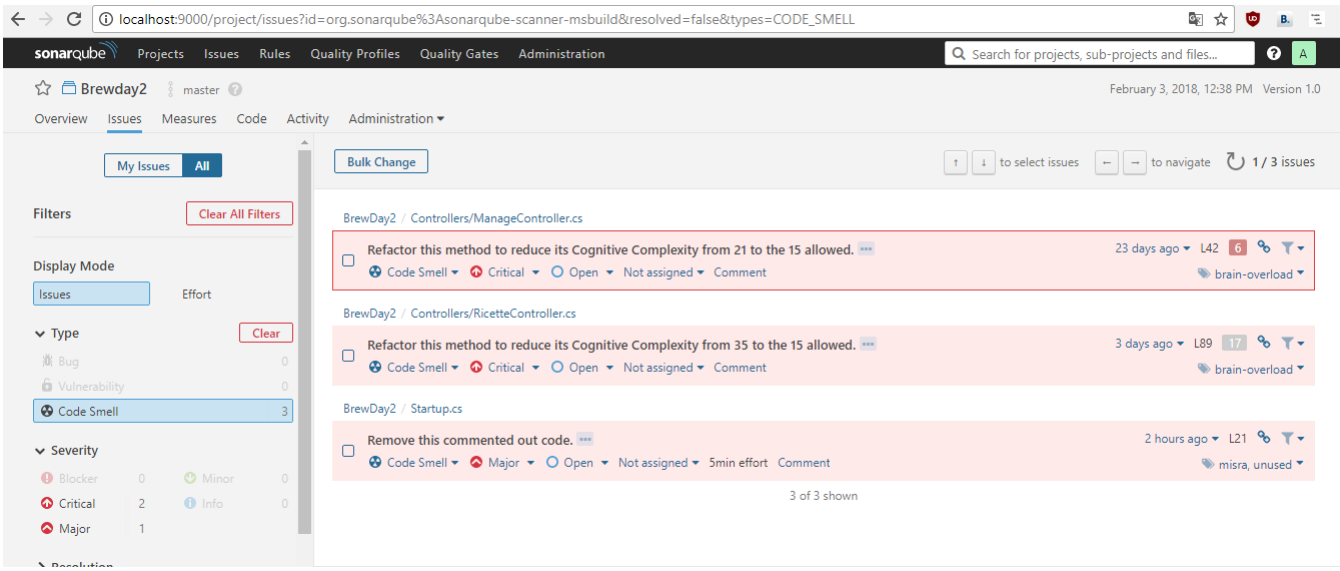
After excluding JS related libraries from the analysis we removed the worst issues and the code quality now gets an 'A'.



We managed to fix almost all the code-smells detected:



Only a few complexity related issue are still not addressed, one is related to the "What should I brew Today" feature that needs a lot of data computation to be executed and it wouldn't be a good choice to divide it in smaller functions because there is no logical cohesion between sub-operations.
The other one is related to a C# default account management function not written by us.



10.2 Unit Test

On the unit test side we did a lot of experimentations to make it available in the SonarQube report: MSTest, MSTest V2, Moq, AutoMoq, SimpleStubs, Nsubstitute, Rhino Mocks, nUnit and xUnit.

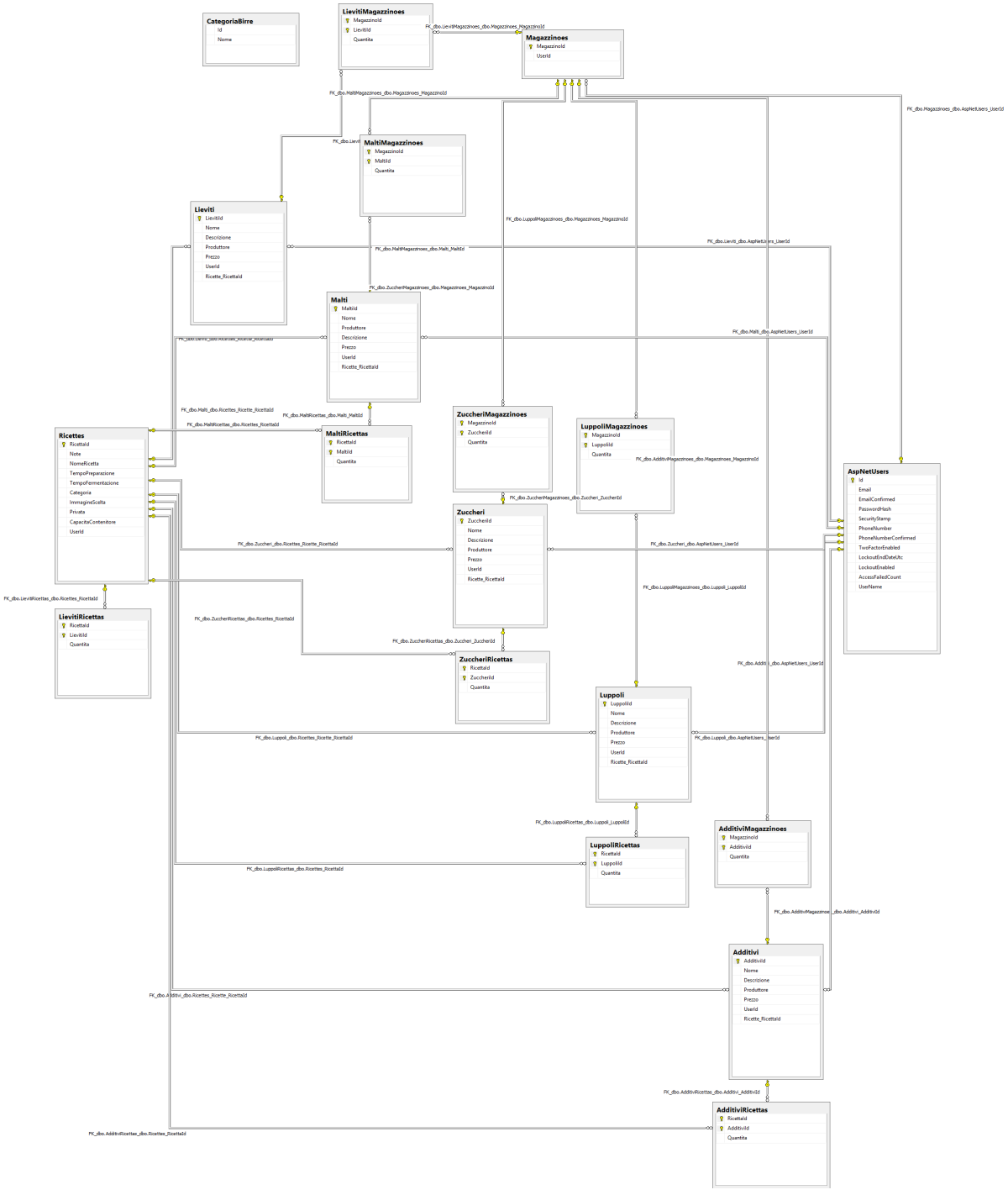
We tried every possible easily available option in order to get the code coverage working in SonarQube, but it seems that it doesn't support most of the standard C# testing frameworks available in Visual Studio.

The only one that seems to be supported is OpenCover, however it's very complicated to set-up (tricky configuration) and it's not issue-free as we can see from their github https://github.com/OpenCover/opencover/issues/121 (they are passing the buck to SonarQube).

So we decided to implement our tests in the standard Visual Studio MSTest V2 platform.
The tests cover most of the code, but it won't be visible in the SonarQube report.
Every controller has from one to five tests per Method in order to make sure everything is working.

11 ER-Diagram

Image included in this document below this page, but it's better to look at the separated file provided in the Docsimageser.png path because it's big and can't fit an a4 page in any way, cutting would cause confusion.


CategoriaBirre
Id
Nome
Lieviti Magazzinoes
MagazzinoId
LieviId
Quantita
FK_dbo.LieviMagazzinoes_dbo.Magazzinoes_MagazzinoId
Magazzinoes
MagazzinoId
UserId
FK_dbo.MaltiMagazzinoes_dbo.Magazzinoes_MagazzinoId
FK_dbo.Magazzinoes_dbo.AspNetUsers_UserId
MaltiMagazzinoes
MagazzinoId
MaltId
Quantita
Lieviti
LieviId
Nome
Descrizione
Produttore
Prezzo
UserId
Ricette_RicettaId
FK_dbo.LuppoliMagazzinoes_dbo.Magazzinoes_MagazzinoId
FK_dbo.MaltiMagazzinoes_dbo.Malti_MaltId
FK_dbo.Lieviti_dbo.AspNetUsers_UserId
FK_dbo.ZuccheriMagazzinoes_dbo.Magazzinoes_MagazzinoId
Malti
MaltId
Nome
Produttore
Descrizione
Prezzo
UserId
Ricette_RicettaId
FK_dbo.Malti_dbo.AspNetUsers_UserId
FK_dbo.Malti_dbo.Ricettes_Ricette_RicettaId
FK_dbo.MaltiRicettas_dbo.Malti_MaltId
FK_dbo.MaltiRicettas_dbo.Ricettes_RicettaId
ZuccheriMagazzinoes
MagazzinoId
ZuccheriId
Quantita
LuppoliMagazzinoes
MagazzinoId
LuppoliId
Quantita
FK_dbo.AdditiviMagazzinoes_dbo.Magazzinoes_MagazzinoId
Ricettes
RicettaId
Note
NomeRicetta
TempoPreparazione
TempoFermentazione
Categoria
ImmagineScelta
Privata
CapacitaContenitore
UserId
MaltiRicettas
RicettaId
MaltId
Quantita
FK_dbo.ZuccheriMagazzinoes_dbo.Zuccheri_ZuccheriId
AspNetUsers
Id
Email
EmailConfirmed
PasswordHash
SecurityStamp
PhoneNumber
PhoneNumberConfirmed
TwoFactorEnabled
LockoutEndDateUtc
LockoutEnabled
AccessFailedCount
UserName
Zuccheri
ZuccheriId
Nome
Descrizione
Produttore
Prezzo
UserId
Ricette_RicettaId
FK_dbo.Zuccheri_dbo.Ricettes_Ricette_RicettaId
FK_dbo.Zuccheri_dbo.AspNetUsers_UserId
FK_dbo.LuppoliMagazzinoes_dbo.Luppoli_LuppoliId
FK_dbo.LieviRicettas_dbo.Ricettes_RicettaId
LieviRicettas
RicettaId
LieviId
Quantita
FK_dbo.ZuccheriRicettas_dbo.Ricettes_RicettaId
FK_dbo.ZuccheriRicettas_dbo.Zuccheri_ZuccheriId
ZuccheriRicettas
RicettaId
ZuccheriId
Quantita
Luppoli
LuppoliId
Nome
Descrizione
Produttore
Prezzo
UserId
Ricette_RicettaId
FK_dbo.Luppoli_dbo.Ricettes_Ricette_RicettaId
FK_dbo.Luppoli_dbo.AspNetUsers_UserId
FK_dbo.LuppoliRicettas_dbo.Luppoli_LuppoliId
AdditiviMagazzinoes
MagazzinoId
AdditiviId
Quantita
LuppoliRicettas
RicettaId
LuppoliId
Quantita
FK_dbo.LuppoliRicettas_dbo.Ricettes_RicettaId
FK_dbo.AdditiviMagazzinoes_dbo.Additivi_AdditiviId
Additivi
AdditiviId
Nome
Descrizione
Produttore
Prezzo
UserId
Ricette_RicettaId
FK_dbo.AdditiviRicettas_dbo.Additivi_AdditiviId
AdditiviRicettas
RicettaId
AdditiviId
Quantita
FK_dbo.AdditiviRicettas_dbo.Ricettes_RicettaId

12 Guides

There are 2 guides available for different kinds of users:

1. User Guide (a functional guide to the user interface interaction)
2. Developer Guide (a fully comprehensive guide that covers the compilation, debuging and deployment of the project)

13 Code manual

The code documentation manual with descriptions of every function is available at the link http://brewday.cu.cc/documentazione (if the domain will come back online soon) or http://185.219.133.28/documentazione/ (direct access with ip in case of problems with the domain).