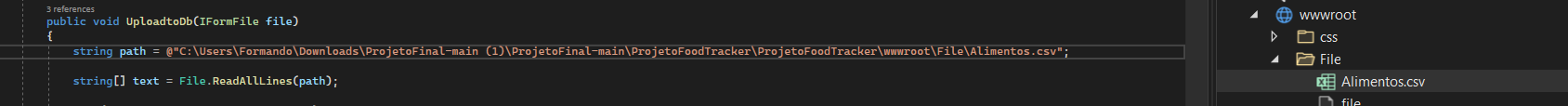
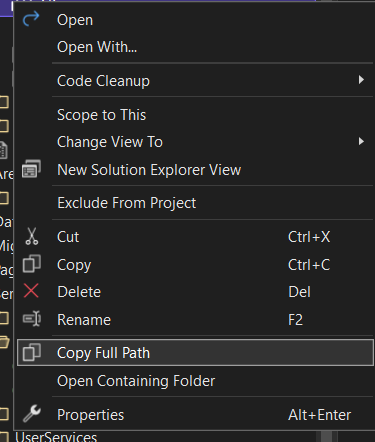
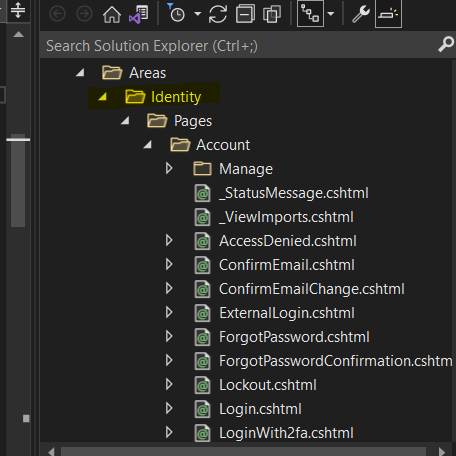
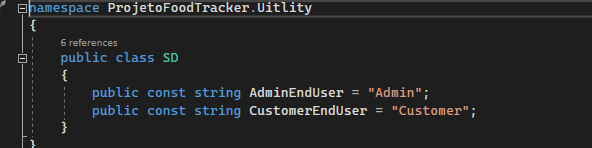
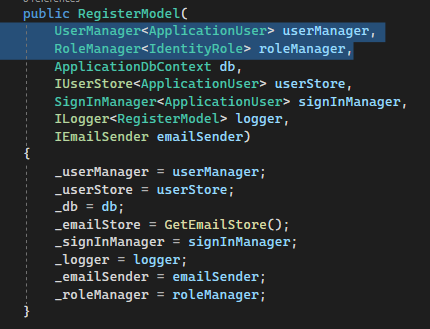
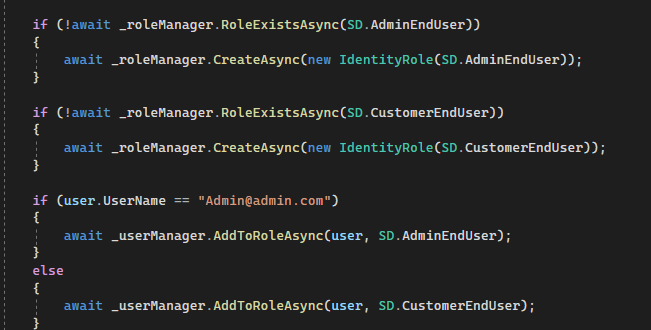
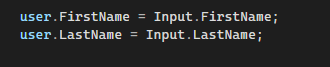
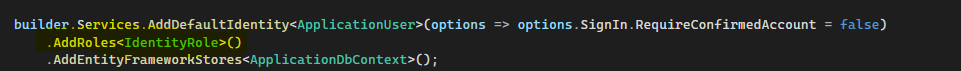
Relatório do Projeto Food Tracker

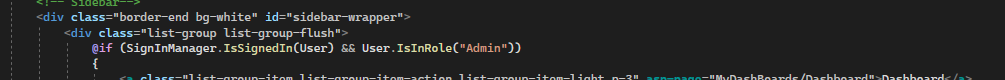
**Detalhes importantes e indispensáveis:**

1. Registar o Admin com [Admin@admin.com](mailto:Admin@admin.com), muito importante para conseguir aceder ao painel do admin;
2. Possivelmente será necessário atualizar o Path para ler o documento
   1. Basta clicar com botão direito do rato no documento e Copy Full Path
   2. E substituir o Path anterior com o novo
3. Clicar em Import para carregar a base de dados;
4. Registar os users que quiser com outros emails;

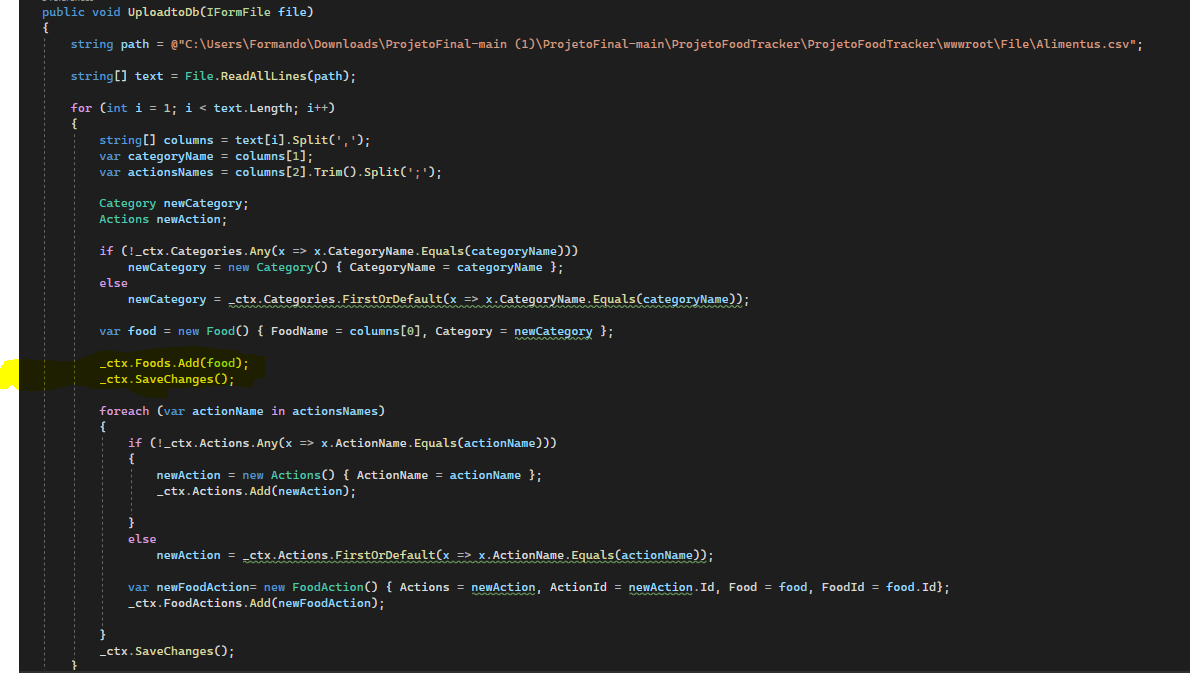
**Explicação das funcionalidades da app/projeto:**

Requisitos App:

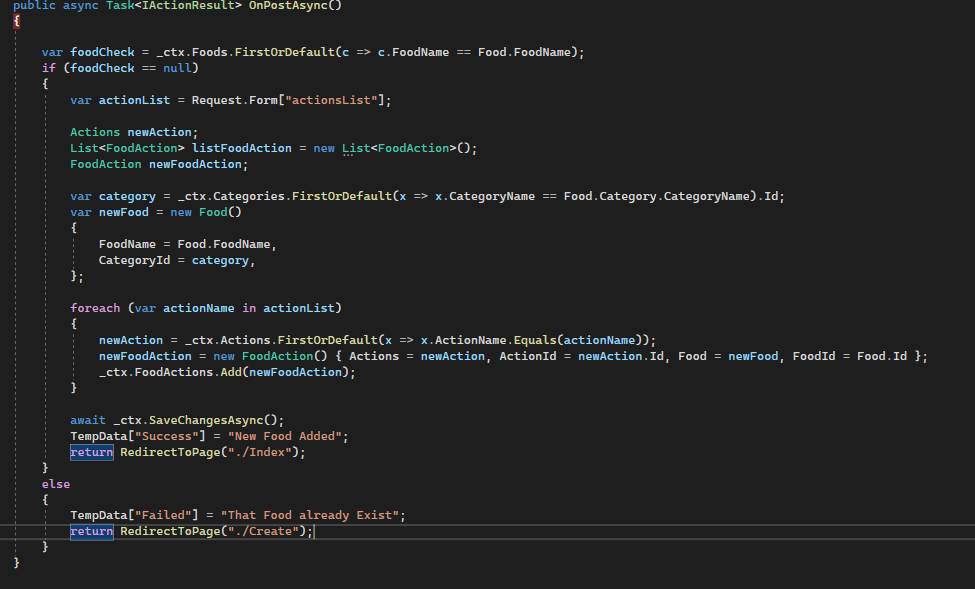
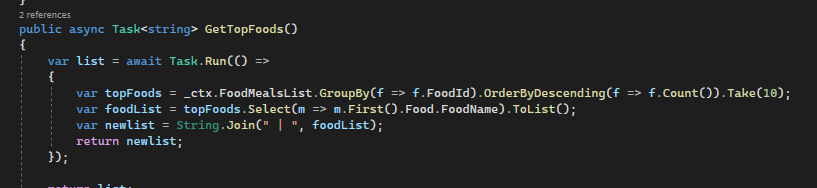
1. A aplicação deverá ter funcionalidades que só o administrador pode realizar e outras disponíveis aos utilizadores da aplicação.
   * Este ponto está definido antes dos requisitos da app no enunciado, mas eu decidi colocar dentro porque acho que faz sentido ficar aqui.
   * Este ponto juntamente com o ponto seguinte foram os pontos pelo qual comecei e fiquei preso durante um bom pedaço de tempo.
   * **Dificuldades:**
     + Como o Sérgio referiu e bem, não era suposto criar Roles, porque não abordamos o assunto nas aulas, mas no entanto ou por teimosia ou por querer provar a mim próprio que conseguia chegar lá por mim decidi então criar Roles, admito que não será a solução mais completa ou flexível, mas atendendo aos pontos antes mencionados não está nada mal para um novato.
     + Os passos foram os seguintes:
       - Scaffolding do Identity, selecionando o Login e o Register, o Login porque previamente tinha definido a class **ApplciationUser** com propriedades obrigatórias, FirstName e LastName, o que me obrigou a alterar o Registo e o Layout do Login ou então não seria possível criar novos Users.
       - Foi criada também a parte um class com propriedades estáticas só para definir as Roles, poderia ter mais para o que era pedido 2 chegavam
       - Dentro do Register.cshtml foi necessário injetar no construtor do Register o roleManager. E adicionar também o identityrole no Program.cs.
       - E por fim no OnPostAsync foram então adicionadas as novas linhas de código para receber o FirstName, LastName e criar as novas roles.
       - Também quero salientar que no layout do Index foi adicionado uma condição em que separa o menu consoante o Admin e o Utilizador comum.



* + - * **Raciocínio:** No final a parte mais complicada deste ponto foi compreender qual era a ordem dos procedimentos a seguir, porque em termos de raciocínio faz sentido que ao registar um utilizador lhe seja atribuída uma role, portanto na minha cabeça seria algures no registo, mas ao mesmo tempo queria destacar o Admin, e não sabendo fazer melhor decidi definir o Email como o diferenciador entre as roles.

1. ***RA1:*** *Deve ser possível importar uma lista de alimentos pré-compilada a partir de um ficheiro.csv. Tal é importante para que se possa inserir centenas de produtos de forma completamente automatizada.*
   * Sem dúvida um dos maiores desafios deste projeto para mim, talvez por ter sido logo ao início e ainda não estava no mindset correto, ou simplesmente por ser um ponto algo exigente.
   * Eu decidi deixar em comentário na parte final do ficheiro **LocalFileUploadService.cs** um cemitério de código que testei e voltei a testar até chegar a solução final. e como o Sérgio teve a oportunidade experienciar parte da minha dificuldade através das dezenas de questões que fui colocando em relação a este tópico
   * **Dificuldades:**
     + A maior dificuldade sem dúvida foi o algoritmo, o testing até chegar a conclusão correta. No entanto devo salientar que reparei num problema que estava a ter e felizmente foi ao início que depois me ajudou a ser mais prático no restante do projeto. O problema foi querer fazer tudo muito bonito e funcional antes de sequer estar a funcionar primeiro, com isto quero dizer por exemplo, se calhar devia ter gasto mais energia primeiro em ler bem o ficheiro em vez de tentar importar o ficheiro através de um botão que vai buscar o ficheiro e o ler.
     + No momento em que estou a escrever este relatório o único botão que existe, o que faz é importar os dados de um local para a base de dados. Nada de muito impressionante, mas 100% funcional.
     + **Raciocínio:**
       - A primeira parte (até ao sublinhado amarelo), muito graças a ajuda e debate na última sessão ajudou a chegar um entendimento de como devia ser feito, lembro que no início estava a tentar fazer tudo com listas, e pode ver isso no cemitério de código que referi, e agora olhando para o resultado final, este código final faz todo sentido e consigo entender porque não estava a conseguir na altura, mais uma vez não estava a ir por partes.
       - A parte abaixo do sublinhado e parte das verificações já foram obtidas através de tentativa-erro com o debug até chegar ao resultado que era pretendido.

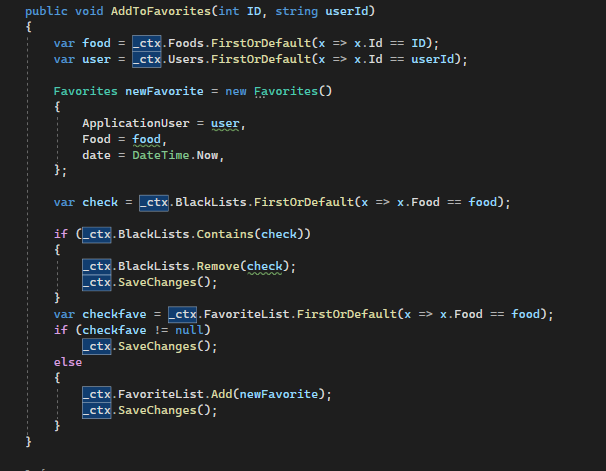
3**. RA2:** Devem ser criadas operações de CRUD para Ação, Alimento e Categoria;

* + Cada Ação corresponde a um dos 5 sistemas de defesa;
  + Categoria (Carne, Fruta, Vegetais, etc.)
  + Felizmente este ponto não foi tao pesado como os anteriores e as dificuldades foram poucas, no entanto eu não segui esta ordem durante o projeto e na altura saltei para os favoritos, a verdade é que 1 dos pequenos desafios que surgiu durante este ponto ajudou bastante em outros pontos a frente.
  + **Dificuldades:**
    - A parte de CRUD em si, é fácil e fácil e bastante rápida, mas, no entanto, e não sei se podemos considerar tudo o que vou referir como extras vistos que não está implicitamente no enunciado, nós quando adicionamos food, temos de poder adicionar as categorias e as ações associadas a essa food e tudo isso foi uma boa jornada.
    - Ignorando a parte das verificações e dos avisos, que fazem sentido, mas que não eram necessariamente obrigatórios, ou seja, extras, o maior desafio aqui foi do lado da view ter, 1 form com um input de text, outro select item e outro com multiple checkboxes. Sem dúvida que foi extremamente desafiante no início entender como iria passar tanta informação, mas também como receber e distribuir o que interessava.
    - A verdade é que eu aqui poderia ter recebido pelo menos a informação em 1 ou 2 objetos, mas durante os testes eu verifiquei que o OnPost juntamente com o Handler conseguia entender a parte referente a text box e ao select item list. Ou seja ⅔ do problema estavam resolvidos, o problema foi que durante um bom par de horas arrisco-me a dizer eu tentei várias maneiras de conseguir receber os dados da checkbox e só consegui com o Request.form. Que segundo o que vi é algo meio arcaico hoje em dia.
    - **Raciocínio:**
      * Neste caso o resultado foi atingido completamente por tentativa erro, várias pesquisas e muita dúvida existencial. Mas que mais uma vez abriu muitas portas para entender a passagem de dados que viriam a seguir.
* **RA3:** A app deve disponibilizar um dashboard onde seja possível visualizar o nº total de utilizadores e refeições registadas, os 10 produtos mais consumidos e o top 5 dos utilizadores mais ativos, i.e. que registaram mais refeições.
  + Este ponto não tive grandes dificuldades foi mais o processo de entender as funções Linq e quais as mais úteis para o que era pedido.
  + **Raciocínio:**
    - Neste caso optei por colocar tudo num serviço, UserService com InterFace IUserService, no sentido em que como isto era mais para o admin, e eram dados sobre os users, fazia sentido para mim que o nome tivesse a ver com users.
    - Dentro deste serviço temos 4 métodos que vão fazer exatamente o que foi pedido no enunciado, sendo o exemplo abaixo talvez o mais chato porque está a mostrar todos os alimentos juntos e então tive de formatar, ainda assim não como desejado, mas dentro das minhas capacidades.  
      

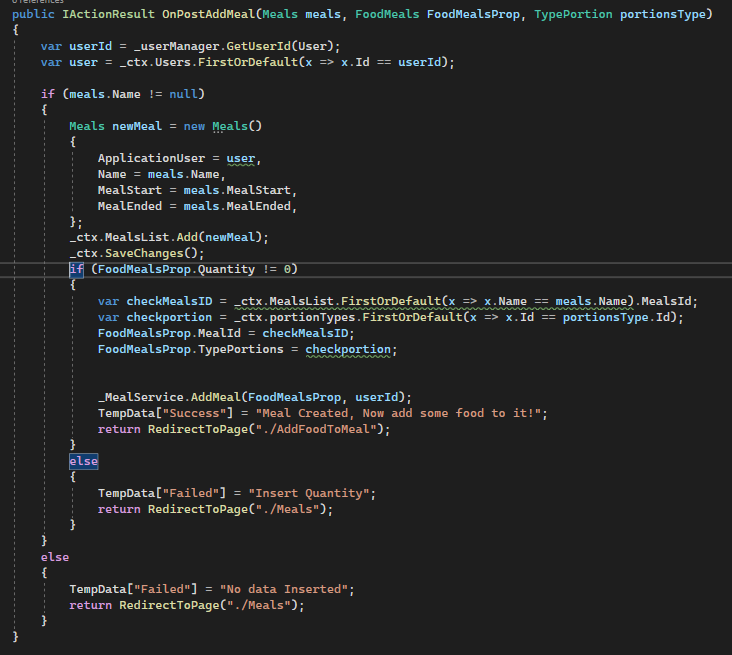
**Requisitos Utilizador:**

1. ***RU1 e RU2:*** *O utilizador deverá conseguir marcar os alimentos como favoritos. Guarde também a data/hora em que a operação foi feita.*
   * Como referi anteriormente este foi o primeiro ponto que fiz após a importação dos dados, isto pode explicar o porquê de algum código estar melhor que outro em vez de ficar progressivamente melhor.
   * Optei por juntar aqui o Favorites e Blacklist porque são bastante semelhantes tirando o aparte que os blacklist não serão sugeridos posteriormente para concluir o objetivo diário **(RU6**).
   * **Dificuldades:** 
     + Uma das maiores dificuldades foi em demonstrar na view da forma que era desejada, acho que cumpre, mas não esta como era desejado, a ideia seria conseguir filtrar parte da tabela a partir dos SelectList items, e de certa forma evitar que os alimentos demonstram todas as suas ações ou categorias sem terem que se repetirem. Mas como isto já envolvia Javascript, considero que não seja um incumprimento, mas apenas um extra que não aconteceu.
   * **Raciocínio:**
     + Eu quis simplificar o máximo possível, daí ter 3 views e apenas um “controller”, também tentei utilizar services para minimizar a repetição de código ou grandes trechos.
     + Eu queria que as views fossem interativas, pelo menos a de adicionar e remover favoritos/blacklisted e apesar de não ter conseguido o que mencionei em cima acho que consigo algo bastante decente.

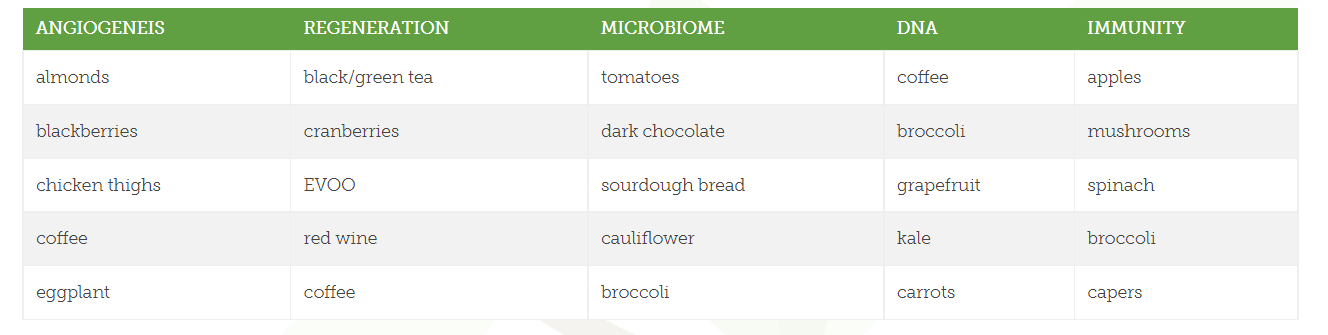
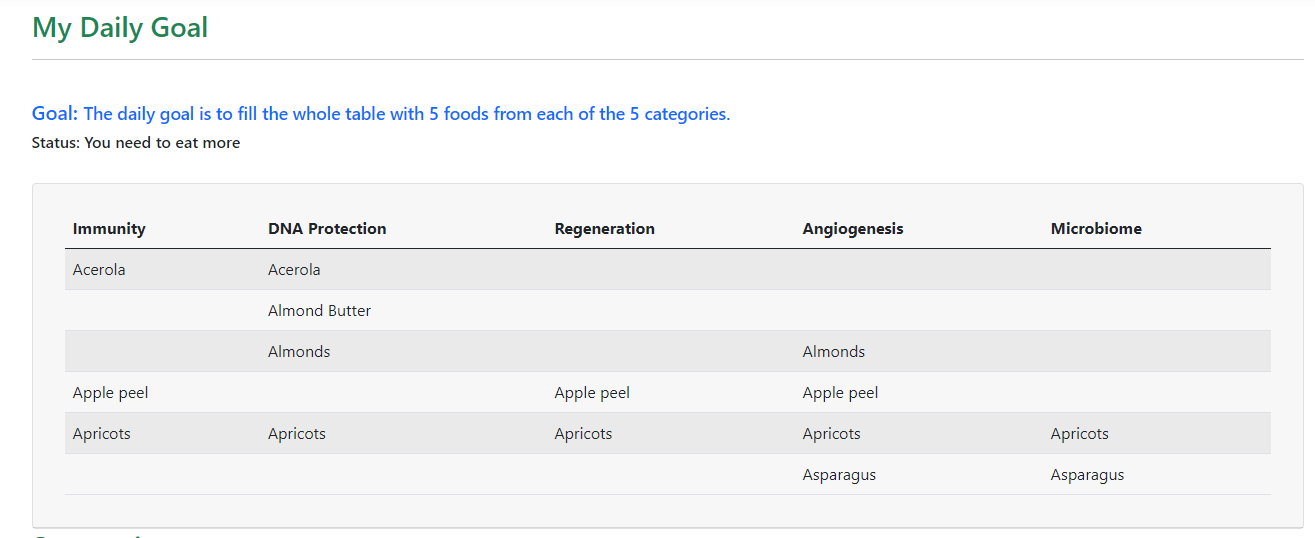
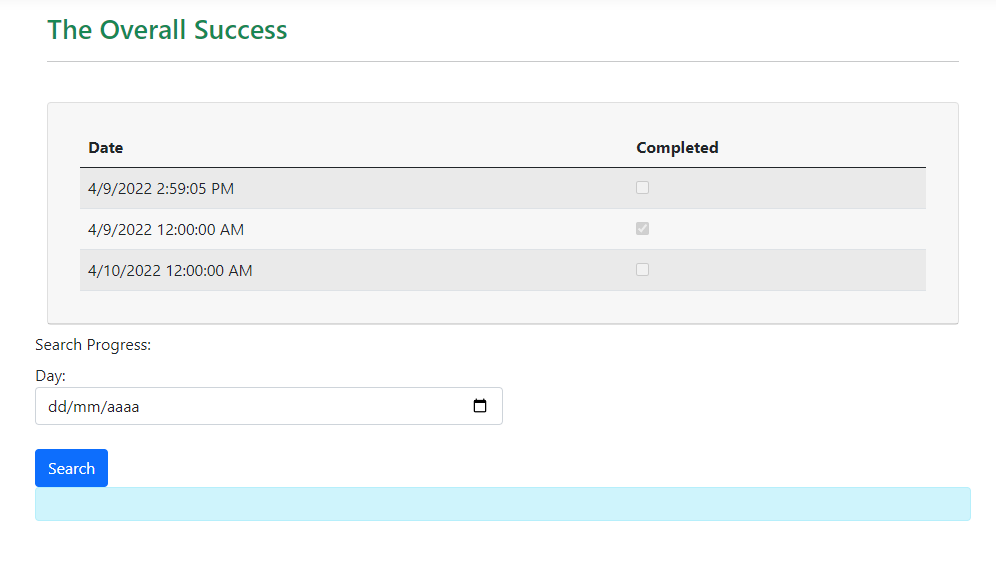
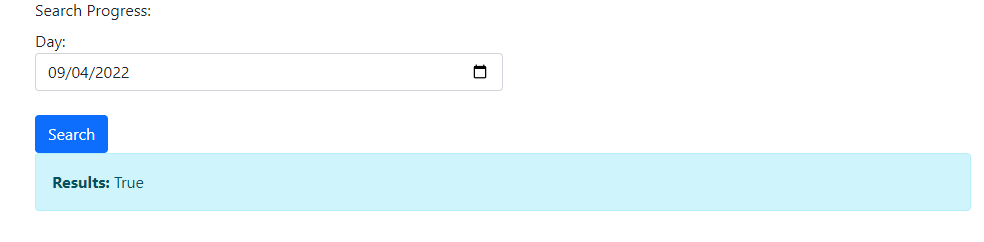


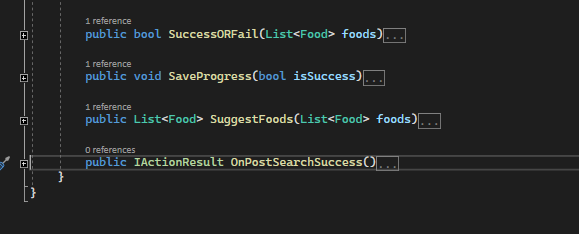
* + Código muito simples e limpo, utilizei o ID como parâmetro para comunicar entre a view e o controller.
  + E depois dentro do método no FoodService é tratado da seguinte forma
  + Aqui não há verificações porque trabalha logo a partir de uma lista que tem de existir previamente, no caso dos alimentos. Depois a verificação que é feita e se este alimento está na blacklist ou vice-versa e se sim então é removida e passado para a favorita e vice-versa, e se já estiver como favorito não faz nada.

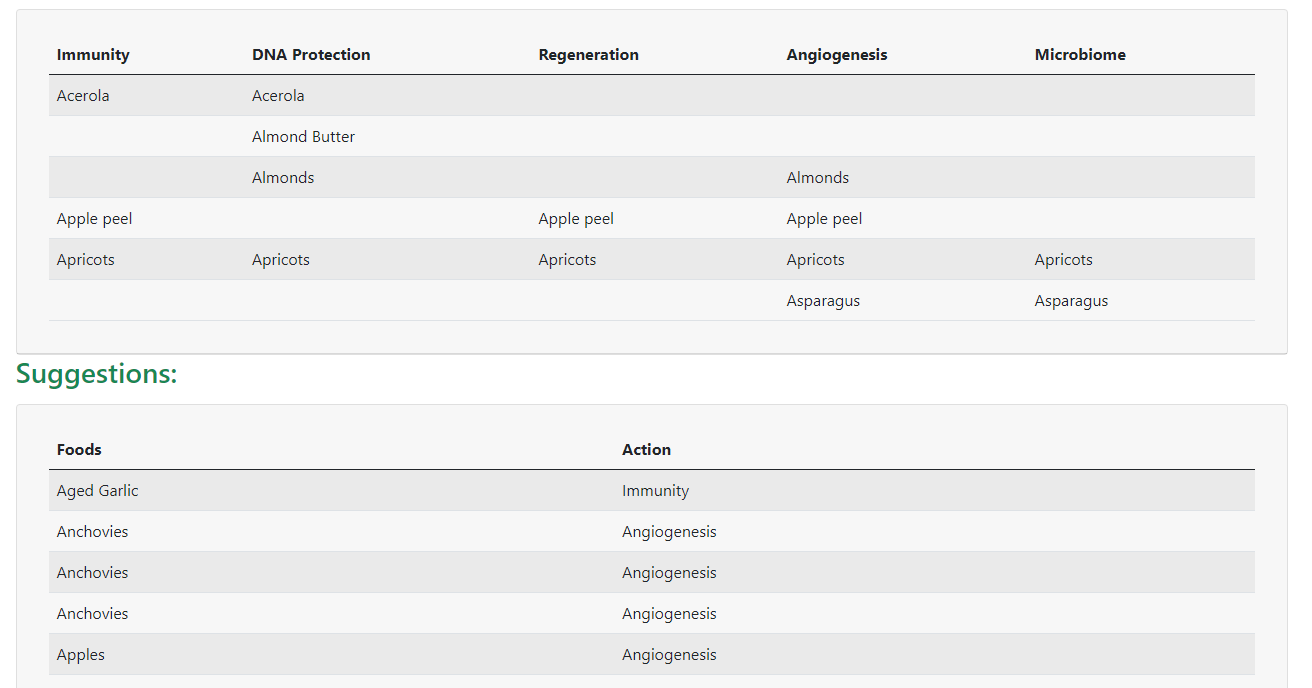
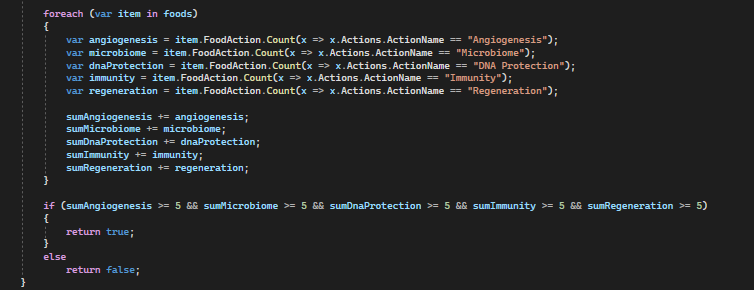
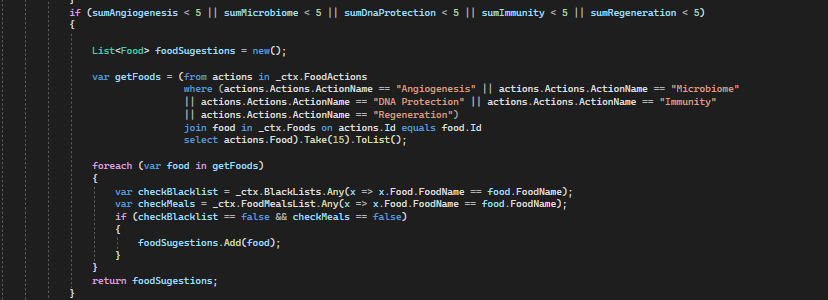
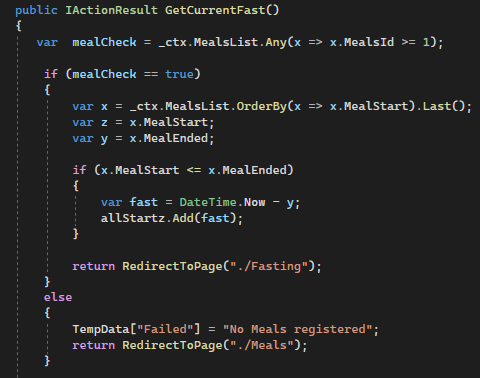
1. ***RU3:*** *O utilizador deverá poder marcar as várias refeições ao longo do dia. Cada refeição inclui 1 ou mais alimentos e também a porção de cada um deles, pelo que deve identificar a medida (gramas, unidades) e o valor associado, e.g. 1 unidade ou 100 gramas. É também necessário guardar o início e o fim da refeição.*
   * Sem dúvida um dos grandes desafios deste projeto, o facto de ter de criar uma meal com algumas propriedades e depois só na FoodMeals é que adicionava as restantes propriedades, admito que foi algo confuso inicialmente.
   * **Dificuldades:**
     + A dificuldade inicial estava mais uma vez em passar dados de forma a quando recebê-los ser capaz de os interpretar, para isso fiz com que o método recebesse 3 parâmetros (ver imagem abaixo), apesar de eu ter a ideia que não é uma forma elegante de programar neste caso e até ao momento foi a única forma que encontrei para resolver a primeira parte do problema.
     + A outra dificuldade aqui foi, também em parte por ter recebido 3 parâmetros, depois não soube organizar o código de forma a utilizar eficientemente o serviço que tinha criado para esta parte que era o MealService, mas pelo menos deixou-me com uma nota mental para ir rever o repository/service pattern quando o tempo permitir.

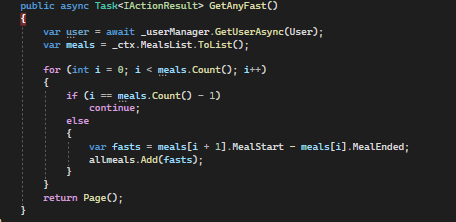


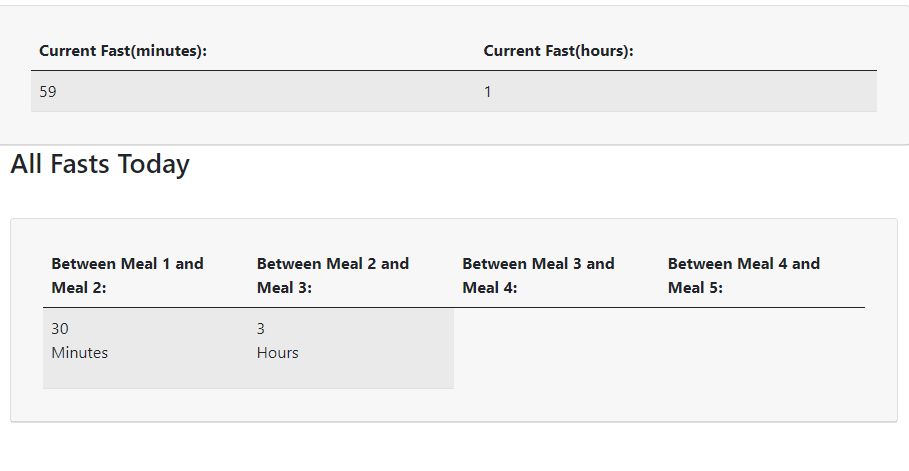
* + **Raciocínio:**
    - Como já disse em cima nas dificuldades a maior barreira estava em compreender que era necessário partir o problema em problemas mais pequenos e atacá-los individualmente (como foi mencionado num dos pontos iniciais), ou seja, nesta fase apesar de alguma pressão já sabia que era uma questão de tempo até começar a ver soluções.
    - Comecei por criar os objetos(meals) a ir testando e verificando se realmente passava os dados que eram pretendidos ao mesmo era criado uma FoodMeal que englobava todos os dados que eram interessantes e assim ficou resolvido um dos pontos mais importantes a meu ver do projeto.

1. ***RU4:*** *O utilizador só poderá usar a app após criar um registo e autenticar-se corretamente.*
   * **Dificuldades/Raciocínio:**
   * Neste ponto em particular não tive muita dificuldade, pois como já tinha conseguido criar Roles e o projeto em si utilizava autentication o que tive de fazer foi o seguinte:
   * Não sei se era só isto que era necessário tendo em conta o que já tinha feito antes, mas desta forma os utilizadores não conseguem aceder às páginas admin e quem não é utilizador também não consegue aceder à app.
   * Eu considerei o Admin apenas um moderador que tem funções muito específicas, daí eu não ter dado acesso às funcionalidades que um utilizador dito normal tem. No fundo quis separar bem todos os intervenientes, mas conseguiria facilmente dar poderes ao admin para aceder a todas as páginas que o user acede (usando a linha de código em cima na imagem)
2. ***RU5:*** *O utilizador deverá poder aceder a uma área com informação sobre o sucesso da incorporação da framework 5x5x5 na sua vida durante um determinado período temporal.*
   * Juntamente com o adicionar food e ponto seguinte eu considero estes os pontos que serão o coração da app, e como tal são bastante exigentes
   * Nesta fase do projeto o desgaste já acusa um bocado, e não como desculpa, mas como explicação de como a solução foi pensada.
   * **Raciocínio:**
     + Como referi muito brevemente nas linhas introdutórias deste ponto o cansaço já acusava um bocado e não era fácil pensar em formas de demonstrar o sucesso, ou era fácil, mas não de forma a que eu com as minhas capacidades atuais conseguisse chegar lá, então fui buscar inspiração ao próprio site do How to Beat Disease. <https://cholangiocarcinoma.org/eat-to-beat-disease/>
     + Esta tabela deu-me algo com que trabalhar e a partir daí o objetivo foi criar queries que permitissem ter todos os alimentos, dentro das categorias a que pertencem, e assim que a tabela tivesse preenchida a missão estava comprida.
     + Alem da tabela também demonstra o **Status**(em cima da tabela) se esta completo o objetivo por hoje ou se precisar de mais alimentos.
     + Depois também tenho outra página para o “Overall Success” que neste momento conta com 2 funcionalidades:
   * A primeira mostra todos os dias registados e se foi ou não completo com sucesso o protocolo.
   * Se quiser procurar 1 dia em específico também apresenta um resultado se foi completado com sucesso = True e falhou = false.
   * **Dificuldades:**
     + A maior dificuldade foi o raciocínio em geral, para a visualização do objetivo diário foi chegar a querie certa, e ainda assim acho que ainda tenho demasiada logica na view
     + Em relação a parte do “Overall Success” foi mais a criação da estrutura, e a filtragem por assim dizer dos dias bem-sucedidos ou dias que correu menos bem



1. ***RU6:*** *O utilizador deverá ter uma área onde lhe sejam sugeridos alimentos de forma a completar com sucesso os critérios da framework 5x5x5 no dia corrente.*
   * **Dificuldades:**
     + Talvez o ponto em estou menos satisfeito, não só pela execução, mas também por não ter conseguido pensar em uma melhor solução.
     + O meu objetivo era mostrar alguns alimentos atendendo as “Ações” que faltavam cumprir, só que da maneira que eu pensei e fiz o método ia ficar incrivelmente longo (Deixei parte do código em comentário porque vi que iria ser extremamente longo)
     + No final optei por deixar uma versão imperfeita, mas simplificada como mostro em baixo.
     + Como podemos ver nesta imagem, faltam alguns alimentos em todas as categorias, mas nas sugestões não aparecem todos (apenas 2), isto acontece porque para cada alimento só consigo mostrar 1 action, no entanto todos os alimentos listados ajudariam a atingir o objetivo diário.
   * **Raciocínio:**
     + A ideia foi ir as Meals do dia, contar as actions e quantas vezes cada action aparecia nas refeições.
     + Se todas estivessem a 5, então objetivo cumprido senão seria enviado para outro método para sugerir o que falta.
     + É aqui que entra também o restante código de qual não me encontro muito satisfeito, mas “resolve”:
     + Como tapar todas as hipóteses tornaria o código gigante, decidi ir pela opção mais “fácil”, como disse em cima. Não é perfeito, não fiquei satisfeito, mas resolve.
2. ***RU7:*** *O utilizador deverá ter acesso a uma área com estatísticas sobre os períodos de jejum ao longo de um determinado período.*
   * Este ponto é algo malandro, porque é resultado de dados já registado em colunas e ter de ir buscar esses mesmo dados e moldá-los de forma a tingir as nossas necessidades (que nesta fase do jogo ainda não é muito fácil)
   * Como o enunciado em si é algo ambíguo, decidi por demonstrar apenas 2 estatísticas, porque acho serem as mais relevantes e mais que isso poderia complicar bastante e não ser possível de completar antes da entrega.
   * **Dificuldades:**
   * A maior dificuldade foi sem dúvida começar a entender a linguagem de queries ao ponto de forma a cumprir os nossos objetivos, existe uma curva de aprendizagem neste ponto, pelo menos para mim.
   * Outro ponto que achei difícil foi a utilização do DateTime e a sua manipulação, fui me apercebendo que é pouco flexível.
   * **Raciocínio:**
   * Para conseguir o fasting atual o raciocínio foi, preciso de verificar a que horas terminou a última refeição e comprar com o tempo atual. Aqui verifiquei uma das dificuldades, a fraca flexibilidade do DateTIme. Felizmente conheci o tipo TimeSpan que ajudou a atingir o objetivo.
   * A imagem abaixo demonstra o código da solução a que cheguei, mas entendo que possa ser algo limitada.
   * Começo por verificar se existe alguma meal, senão o programa crashava aqui.
   * a segunda verificação nem eu tenho a certeza se faz sentido, porque a minha ideia aqui é garantir que mais nenhuma refeição começou
   * Por fim o que faz é calcular as horas atuais menos a horas em que a refeição terminou.
   * A outra funcionalidade implementada foi a possibilidade de ver o fast entre as várias refeições durante o dia.





Extras:

Nesta parte vou listar todas as funcionalidades que eu considero extra atendendo ao facto de não estarem explicitamente referidas no enunciado, mas que compreendo que estivesse subentendido.

1. Roles de Admin e Customer
2. Adicionar campos ao Registo do User
3. Gerir Users, como admin
4. RemoveMeal
5. RemoveFavorite
6. RemoveBlacklist

Dúvidas que gostaria de obter resposta:

Este ponto é uma forma de tentar melhorar um projeto após o término do módulo, pois tenciono continuar a aperfeiçoar a app à medida que vamos aprendendo algum Javascript no próximo módulo. Ao mesmo tempo aprender a otimizar o código que foi feito e outras funções que gostava de ter aplicado, mas na altura não sabia como fazer ou por onde começar.

Aqui ficam os pontos que gostava de obter feedback:

1. Como poderia ter organizado melhor o Projeto (No solution Explorer)
2. Como poderia ter otimizado a utilização de serviços
3. Exemplos onde poderia ter utilizado métodos async e como (porque tentei, mas não estava a conseguir)
4. Como poderia reduzir o código/logica na parte da view, com exemplos se possível
5. Como poderia ter resolvido o ponto: “***RU6:*** *O utilizador deverá ter uma área onde lhe sejam sugeridos alimentos de forma a completar com sucesso os critérios da framework 5x5x5 no dia corrente.”* De uma forma mais otimizada.
6. Por fim agradecia que anotasse outros detalhes que fosse importante eu ter em conta daqui para a frente com vista em melhorar os mesmos.