Projeto Final de Desenvolvimento Web .NET

Conteúdo deste documento: Competências; Proposta de projeto final; Entregas; Equipes.

Competências

1. Elaborar uma modelagem a partir de um problema.
2. Criar uma classe para cada tabela ao usar o EF.
3. Criar um contexto do EF.
4. Definir relações 1-N entre entidades.
5. Definir relações N-N entre entidades.
6. Utilizar migrações para criar o banco de dados.
7. Salvar objetos no banco usando EF.
8. Usar Include para fazer Joins usando EF.
9. Criar UseCases com seus devidos Requests e Payloads.
10. Aplicar atributos de validação em Payloads.
11. Criar implementações auxiliares em serviços com interface e implementação.
12. Configurar serviços com Transient, Scoped e Singleton.
13. Aplicar criptografia de senha ao criar um usuário.
14. Criar sistema de autentificação de usuário com JWT.
15. Configurar claims no JWT.
16. Habilitar autentificação baseado no JWT nos Endpoints corretos.
17. Extrair claims do JWT e usar em operações.
18. Configurar o CORS para um Frontend específico.
19. Definir Endpoints com URL e métodos adequados.
20. Implementar Endpoints com serviços/use cases.
21. Retornar status code corretamente dos Endpoints.
22. Habilitar Swagger.
23. Definir testes automatizados para algumas operações complexas.

Proposta de projeto final

Você deverá implementar um rplace. Um sistema onde os usuários podem criar uma conta e criar um quadro onde os jogadores adicionados podem desenhar pixeis de tempos em tempos.

O projeto será feito em várias entregas que devem ser concluídas uma a uma até que o projeto esteja concluído.

Abaixo uma descrição do cliente do projeto dizendo exatamente o que ele quer, toda a informação do que é necessário para a construção do projeto:

- O sistema permite que o usuário crie uma conta com nome de usuário, que seja único com ao menos 6 caracteres, e-mail, que seja único deve ser um e-mail de fato, senha com letras maiúsculas, minúsculas, caracteres especiais, números e ao menos 8 caracteres. Um campo de repetir senha deve ser incluso. O usuário pode passar o link para uma imagem na web e esse link será salvo para usar como imagem de perfil. Uma descrição do perfil também deve ser inclusa. Espera-se que a senha seja armazenada nos padrões da indústria. Um usuário ainda tem um dado adicional importante que é o plano do sistema e até quando ele é válido, que pode ser um dos seguintes: Gratuito, Gold e Platinum. Considere que isso pode mudar no futuro e mais planos podem ser incluídos.
- O sistema permite que o usuário faça login no sistema usando seu nome de usuário ou e-mail e sua senha. O sistema usa padrões de autentificação da indústria.
- Todos os usuários podem ver o perfil de outro usuário que é sempre público, neste perfil todos os dados exceto e-mail e senha do usuário está disponíveis. Isso inclui o plano do usuário.
- Usuários logados podem editar seus dados básicos, exceto a senha. Para alterar os dados é necessário fornecer novamente a senha como medida de segurança, mesmo que o usuário já esteja logado.
- Usuários logados podem ver planos disponíveis. Para aderir a um plano ele precisa de um código válido para esse plano. No caso, a geração desses códigos será feita manualmente pelo desenvolvedor e colocados no banco de dados, ou seja, não existe nada no sistema para criação de códigos, eles são criados manualmente e vendidos em *Gift Cards*. Mas deve ser possível que o usuário entre com um código e atualize seu plano. Um código não pode ser usado mais de uma vez.

Cada código ainda tem um tempo associado com a duração do plano; Isso é importante porque um usuário Gold pode comprar mais duração ao seu plano e estender sua validade. É recomendado que o usuário e seu jwt tenham o último plano e sua data de expiração. Caso o plano não seja mais válido, considera-se que o usuário é um usuário Gratuito.

- Usuários logados podem ver todas as suas salas de desenho com seus respectivos nomes e quem criou aquela sala.
- Usuários logados podem ver, aceitar ou rejeitar convites para participar de uma sala de desenho.
- Usuários logados podem criar salas de desenho. Para isso definem o nome da sala e seu tamanho. Existem limites de tamanho com base no tipo de usuário. Usuários com plano Gratuito podem ter salas de até 64x64. Usuários com plano Gold podem ter salas de até 128x128. Usuários com plano Platinum podem ter salas de até 256x256.
- Um usuário logado tem permissões sobre uma sala de desenho. Isso significa que cada usuário terá permissões diferentes sobre cada sala de desenho. Uma sala de desenho é como um grupo e o usuário poderá pedir para ver os participantes da sala bem como seus níveis de permissão que podem ser: Dono, Administrador, Pintor e Plateia. Nos próximos tópicos serão discutidos as funções da sala de desenho e a permissão necessária para realizar essas operações.
    - Convidar: Donos e Administradores podem digitar o nome de um usuário e adicioná-lo na sala. Convites são enviados e o usuário poderá aceitar ou recusar.
    - Remover: Donos e Administradores podem remover outros usuários. Donos podem remover Administradores, mas não o contrário.
    - Promover: Donos e Administradores podem mudar permissões de outros usuários, isso é mudar entre Dono, Administrador, Pintor ou Plateia. Contudo, eles não podem nunca prover a algo acima de seu cargo nem afetar alguém com um cargo igual ou superior. Ou seja, um Dono não pode transformar outro Dono em Plateia. Um Administrador não pode promover alguém a Dono ou reduzir o nível de outro Administrador.
    - Pintar: Um Dono, Administrador ou Pintor pode escolher pintar um pixel com uma posição (x, y) com uma cor qualquer RGB. Mas cada usuário só pode pintar um pixel a cada 60 segundos. Usuário Gold podem pintar a cada 50 segundos e usuários Platinum a cada 40 segundos.

- Ver: Todos os usuários podem pedir todos os pixels pintados da tela para ver como ela está.

Dica: Com cerca de 15 endpoints é possível implementar todo o sistema explicado acima.

Entregas

Nesta secção será discutidas as entregas do projeto. É extremamente recomendável que cada entrega seja feita e validade pelo professor antes de seguir para a próxima entrega. Recomenda-se também ler todas as entregas antes de começar a primeira. Recomenda-se um trabalho em equipe afim de terminar tarefas mais rapidamente ao invés de separar o projeto. Essa decisão fica a cargo da equipe contudo é importante saber que todo conteúdo precisa ser dominado por todos os integrantes.

1. Documentação e Organização do Projeto: Crie um git compartilhado entre você e sua equipe com um projeto Web criado. Você também pode criar um Notion ou similar onde colocará anotações e diagramas. Esses recursos poderão ser consultados na prova da disciplina, e apenas eles. Essa etapa não precisa ser validada ficando a cargo de cada equipe.

2. Modelo Entidade Relacionamento: Proponha um diagrama ou descrição textual que levante todas as tabelas que serão necessárias no projeto, suas relações e colunas.

3. Modelo do Entity Framework: Escreva as classes que representam as tabelas já com Navigations e um DbContext.

4. Migrations: Configure o DbContext como Serviço, adicione a *string* de conexão como variável de ambiente e rode as migrations criando o banco de dados.

5. Definir Use Cases (sem implementação): Defina a lista de todos os Use Cases que serão necessários para implementação do sistema. Você pode já escrever o código não implementado dos Use Cases ou mostrar uma lista de Use Cases. Faça também a entrada e retorno de cada Use Case que é fundamental para implementação futura.

6. Definir Endpoints: Defina os endpoints (seja textual ou por código) que são necessários para que o sistema funcione completamente.

7. Configurar Autentificação: Configure a autentificação com Json Web Tokens. Defina o que será enviado nas suas claims. Se pergunte: O que é útil que eu tenha em fácil acesso para fazer implementações depois? Não inclusa nesta etapa a parte de geração do JWT apenas de configurar autentificação e recebimento do mesmo.

8. Definir Serviços (interfaces): Discuta com seus colegas quais serviços são úteis para implementação dos Use Cases. Dica: De fato, com apenas 4 serviços, você terá mais que o necessário, considerando que você pode implementar todo o resto usando seu

DbContext. Ou seja, 4 partes do código que repetem muito e você pode implementar em um lugar separado. Mas você pode propor mais se quiser.

9. Escreva os Testes: Você escreverá uma função para testar alguns dos serviços que você definiu. Para isso crie classes sem implementação dos serviços e escreva seus testes. Quais serviços serão testados será informado a você quando chegar nesta etapa.

10. Implementar Serviços: Implemente os serviços que você propôs. Depois rode os testes e veja se todos funcionam.

11. Configurar Serviços e Use Cases: Configure no sistema de injeção de dependência todos os serviços e use cases que você criou.

12. Implementar Use Cases e Endpoints: Implemente os Use Cases que você propôs usando seus serviços e o DbContext, também implemente seus Endpoints usando seus Use Cases; Nesta etapa você pode dividir o trabalho entre os integrantes da equipe e, além disso, mostrar e testar cada implementação separadamente no Postman ou Swagger e ir validando seu trabalho aos poucos.

13. Conectando a um Frontend [Extra]: No final do projeto um frontend será entregue a vocês tentem ajustar para ver seu sistema funcionar ao vivo e a muitas cores.

Equipes

Anna Beatriz Guerra – Thayna Schaeffer – Ana Julia Pereira

Thais Michel – Ketlyn Sofia – Bruna Elohá

Heloise Zelma – Maria Luiza Geraldo – Lasnine Miranda

Fernanda Klechowicz – Rebeca Silveira Ianz – Júlia Carolina Gabriel

Juliana Stadler – Kessyane De França – Joyce Nascimento

Jhenifer Halma – Lays Arceles De Souza – Leticia Burlinski