ARTESIS PLANTIJN HOGESCHOOL ANTWERPEN

Portfolio Programmeren in .NET

ASP.NET MVC app voor

1 Kadering

Voor je portfolio dien je met behulp van Entity Framework en ASP.NET MVC een app te maken die een kmo de mogelijkheid geeft om de facturatie van het bedrijf te beheren.

De app is niet publiekelijk toegankelijk. Medewerkers dienen zich dus te authentiseren met een gebruikersnaam en paswoord. Afhankelijk van de rol die je krijgt zal je bepaalde acties mogen uitvoeren en anderen niet.

2 User story's

2.1 Security en profiel

2.1.1 Gebruiker aanmaken

Als gebruiker met een administrator rol kan ik gebruikers aanmaken en hen 1 van de 2 rollen (administrator of medewerker) toekennen. Hou er rekening mee dat er in de toekomst nog rollen zouden kunnen bijkomen, en een gebruiker meerdere rollen kan combineren.

2.1.2 Mijn gegevens beheren

Als een geregistreerde gebruiker kan ik te allen tijde mijn gegevens bewerken, zoals naam, voornaam, email, adresgegevens en eventuele andere relevante gegevens. Ik heb ook de mogelijkheid om mijn wachtwoord te wijzigen. Het is vanzelfsprekend niet de bedoeling dat ik mijn rollen kan beheren.

2.1.3 Wachtwoord resetten

Als een geregistreerde gebruiker heb ik steeds de mogelijkheid om mijn wachtwoord te resetten in geval ik dit vergeten ben. Ik kan daartoe op een hyperlink klikken op het aanmeldingsscherm. Hierdoor zal ik een email met een hyperlink toegestuurd krijgen op mijn geregistreerd email adres, langs waar ik mijn wachtwoord opnieuw kan instellen. De link mag slechts 24u beschikbaar blijven.

2.1.4 Rollen beheren

Als gebruiker met een administrator rol kan ik nieuwe rollen definiëren, de naam ervan aanpassen en verwijderen. Enkel de administrator rol is fixed en kan niet gewijzigd of verwijderd worden.

2.2 Facturatie

2.3 Klanten beheren

Als medewerker kan ik klanten toevoegen, bewerken en verwijderen. Opgelet, klanten mogen niet letterlijk verwijderd worden uit het systeem, maar mogen niet meer zichtbaar zijn in die delen van de website waar klanteninformatie gebruikt wordt voor andere zaken zoals bv. voor het toevoegen van facturen. Bij het raadplegen van bestaande facturen dienen de gegevens vanzelfsprekend wel getoond te worden.

2.3.1 Verkoopfacturen beheren

Als medewerker kan ik een verkoopfactuur toevoegen, bewerken en eventueel verwijderen. Het spreekt voor zich dat deze gekoppeld dient te worden aan een klant. Je voorziet de minimaal vereiste velden zoals factuurdatum, unieke oplopende code bestaande uit [yyyy][mm]-[teller]. De teller bestaat daarbij uit 4 cijfers die oplopen binnen die maand. Bij een nieuwe maand start de teller terug van 0. Onze eigen bedrijfsgegevens en die van de klant dienen op het factuur getoond te worden. Opgelet, een factuur mag enkel verwijderd worden zonder meer indien er nog geen detaillijnen voorzien zijn. Van zo gauw er detaillijnen bestaan, dient verplicht een reden voor verwijdering opgegeven te worden. Het factuur dient dan gemarkeerd te worden als verwijderd, maar niet effectief verwijderd.

Om ervoor te zorgen dat een factuur niet meer kan gewijzigd worden achteraf houden we ook een status bij die aangeeft of het factuur als afgewerkt wordt beschouwd.

Het is hierbij uiteraard de bedoeling om vanuit een overzichtslijst, inclusief filtering, te kunnen vertrekken om de gewenste acties uit te voeren. Hiertoe behoort ook het afdrukken van een factuur.

Let op berekende gegevens!

2.3.2 Detaillijnen verkoopfactuur beheren

Als medewerker kan ik aan een bestaand factuur detaillijnen toevoegen, aanpassen en verwijderen zolang het factuur niet als afgewerkt gemarkeerd staat.
Volgende velden moeten zeker aanwezig zijn

- Item (kan een product of dienst zijn waarover het gaat en is manuele ingave. Je dient dus geen extra tabellen te voorzien om producten te beheren)
- Eenheidsprijs
- Korting
- Aantal
- BTW percentage

3 Technische vereisten

Bij het uitwerken van deze app dien je rekening te houden met volgende zaken:

3.1 Databank

1. Je ontwerpt zelf een databank waarin de gebruiker de nodige gegevens kan bewaren.
2. Je database dient minimaal uit 4 tabellen te bestaan waarbij minstens 1 één-op-meer relatie bestaat tussen deze tabellen.
3. Je gebruikt **SQLServer** als RDBMS.
4. Je zorgt bij je portfolio voor een SQL-script dat alle statements bevat om de initiële database aan te maken op eender welke computer. Dit script dien je te genereren met behulp van Entity Framework. Je werkt dus code-first!

3.2 Applicatie

1. Je voorziet minimum een eenvoudige, duidelijke menu structuur om elke bewerking in je app te kunnen uitvoeren.
2. Je voorziet een basis CRUD-gebruikersinterface voor elke entiteit in je applicatie.
3. Je werkt een gebruiksvriendelijke gebruikersinterface uit. Hou hierbij rekening met volgende zaken:
    a. Duidelijk
    b. Eenvoudig in gebruik
    c. Consistent
    d. Validatie van gebruikersinput

3.3 Technisch

1. Werk een duidelijke professionele architectuur uit voor je applicatie (bv. aparte laag voor entiteiten, aparte laag voor data access).
2. Werk je data laag uit in functie van een disconnected model.
3. Voorzie minstens 1 zelf uitgewerkte inline query in je data laag, gebruik makend van Entity Framework (geen pure ADO.NET gebruiken!).
4. Voorzie minstens 1 stored procedure call in je data laag, gebruik makend van Entity Framework (geen pure ADO.NET gebruiken!).

4 Portfolio

Je portfolio moet bestaan uit volgende onderdelen:

1. Je ERD.
2. Een SQL-script dat de database kan aanmaken op eender welke SQLServer instance.
3. Een zip bestand met daarin je volledige solution.
4. Een bondige maar duidelijke gebruikershandleiding van je applicatie.
5. Een begeleidend document waarin je toelicht in welke bestanden je gebruik hebt gemaakt van een inline query en stored procedure call.

Voor de oplevering van je portfolio gelden volgende afspraken:

1. **De deadline voor oplevering wordt tijdig meegedeeld!**
2. Inleveren doe je door al je bestanden in 1 zip bestand beschikbaar te maken via WeTransfer (wetransfer.com).
   RAR-bestanden worden NIET aanvaard!
3. Voorzie een duidelijke naam voor al je bestanden!