Examentraining 1

Realiseren

hoofdstuk

4

Samenwerken binnen GIT

Algemene informatie

Onderwerp	Samenwerken binnen GIT
Leerdoel(en)	1. De student kan een GIT commit en push uitvoeren 2. De student kan een merge conflict oplossen
Vereiste voorkennis	1. De student heeft een voorkennis van commits 2. De student heeft een voorkennis van push
Kwalificatiedossier	☐ B1-K1-W1: Plant werkzaamheden en bewaakt de voortgang ☐ B1-K1-W2: Ontwerpt software ☒ B1-K1-W3: Realiseert (onderdelen van) software ☐ B1-K1-W4: Test software ☐ B1-K1-W5: Doet verbetervoorstellen voor de software ☒ B1-K2-W1: Voert overleg ☐ B1-K2-W2: Presenteert het opgeleverde werk ☐ B1-K2-W3: Reflecteert op het werk

Inhoudsopgave

Introductie

Je hebt in alle voorgaande realiseren lessen geleerd hoe je een goedwerkende applicatie kunt bouwen waarin de gebruiker alle functionaliteiten beschreven kan uitvoeren.

Echter missen we een laatste en belangrijk deel van onze applicatie: Authenticatie van gebruikers en het bepalen van hun rechten (wie mag wat?). Zo mag een reguliere gebruiker bijvoorbeeld geen settings aanpassen en mag een administrator dit juist weer wel.
Wellicht mag een caissière alleen producten scannen en mag een baliemedewerker weer de prijzen van een product aanpassen? Dit zijn allemaal dingen die je kunt programmeren.

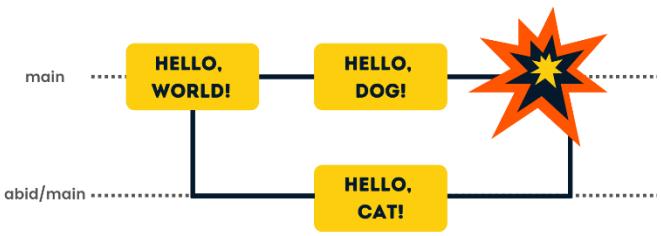
Een gebruiker moet dus inloggen voordat de applicatie gebruikt kan worden. Daarna wordt er per scherm bepaald of de gebruiker deze specifieke actie wel of juist niet mag gebruiken.

Het maken van een loginscherm en daaromheen toegangscontrole noemen we een authenticatiesysteem

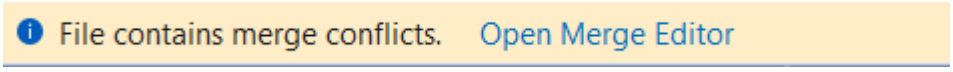
Inhoud

1. Merge conflicts

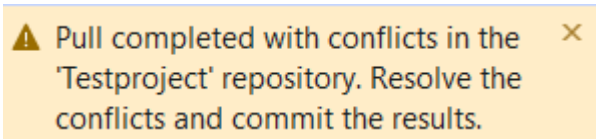
Een merge-conflict ontstaat wanneer twee of meer personen wijzigingen aanbrengen in hetzelfde bestand op verschillende branches en vervolgens proberen deze branches samen te voegen. Azure DevOps kan de wijzigingen niet automatisch samenvoegen omdat het onduidelijk is welke versie behouden moet worden.



Als je wilt jouw wijzigingen wil committen en pushen krijg je een melding dat dit niet kan. Je kan dan kiezen voor pull before push (dus eerst de versie van de server halen en dan alles naar de server sturen. Echter als je met meer dan 1 persoon wijzigingen in een bestand hebt aangebracht krijg je dit boven een file in Visual studio:

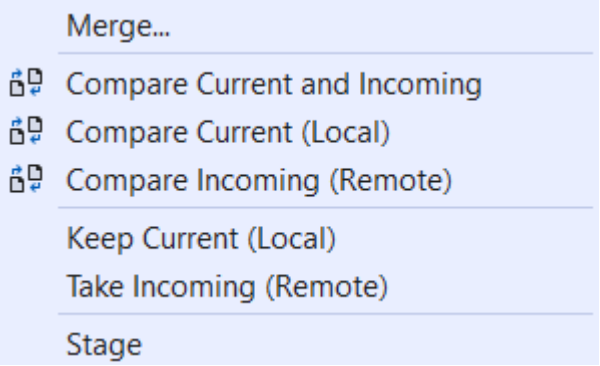


In je git changes tabblad verschijnt deze tekst:

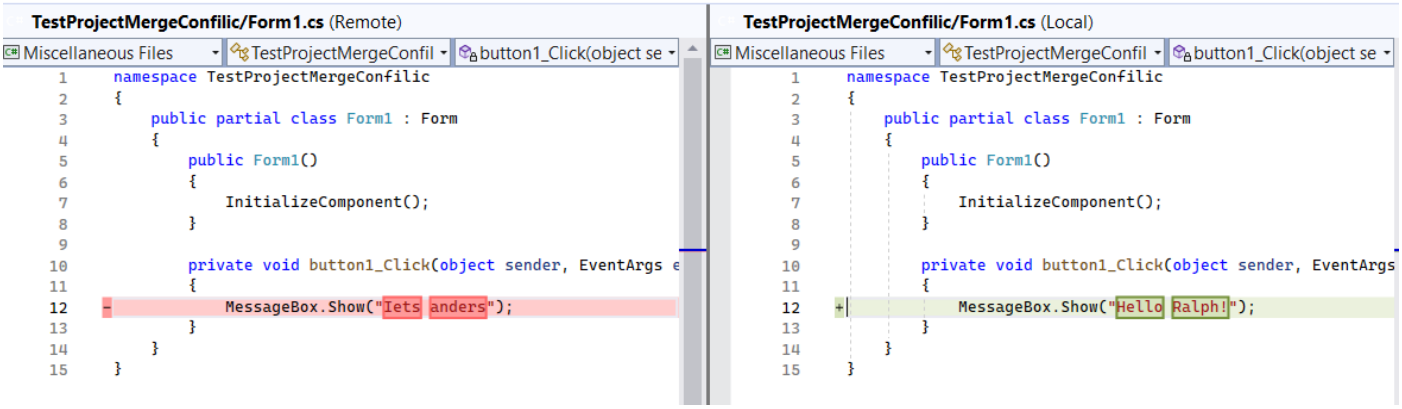


Ofwel het is zaak om je wijzigingen handmatig samen te voegen. Jij (SAMEN MET JE TEAM) bepaalt dan welke code je wilt bewaren.

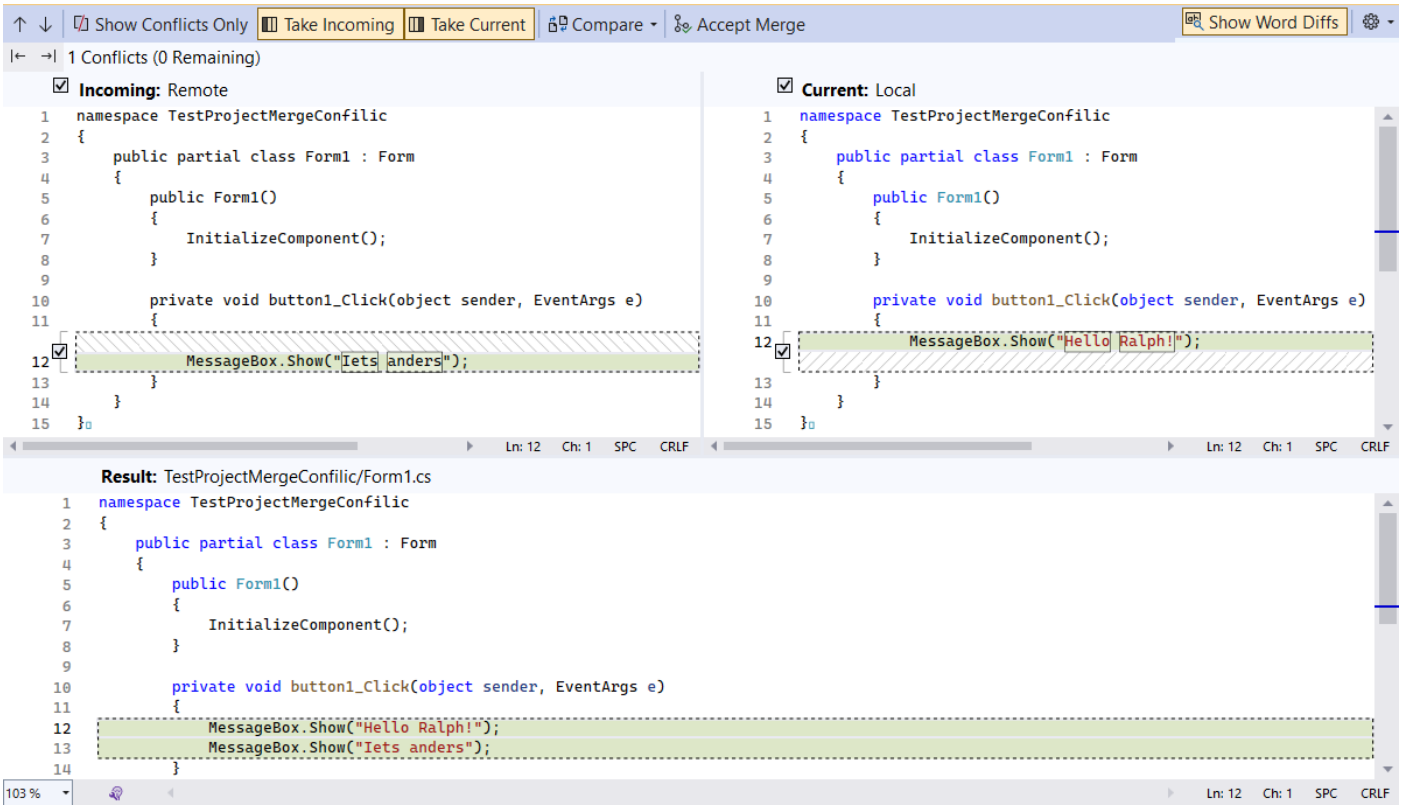
In je git changes kun je per bestand het volgende contextmenu oproepen:



Belangrijke opties hierin zijn natuurlijk het vergelijken van jou versie (current) met degene op de server (incoming, je bent immers aan het pullen als het verkeerd gaat…).



Een andere optie die je kunt toepassen is de Merge Editor. Je krijgt dan onderstaand scherm te zien. Het mooie aan dit scherm is dat je hierbij de mogelijkheid heb om jouw aanpassingen (Take Current) EN de aanpassingen (Take Incomming) van een ander teamlid allebei zou kunnen behouden. Je krijgt in het scherm onderin dan de code die te zien die het uiteindelijk wordt.



Merge-conflicten kunnen enkele uitdagingen met zich meebrengen tijdens het ontwikkelproces. Hier zijn enkele veelvoorkomende uitdagingen waarmee je te maken kunt krijgen bij het oplossen van merge-conflicten:

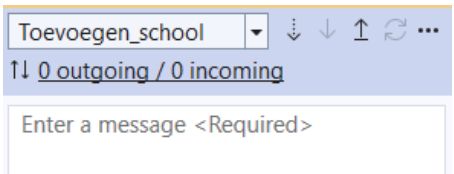
1. **Complexiteit van conflicten**: Merge-conflicten kunnen complex zijn, vooral wanneer er meerdere conflicten optreden in verschillende delen van een bestand of wanneer meerdere bestanden conflicterende wijzigingen hebben. Het vereist vaak een grondige analyse en begrip van de wijzigingen om de beste oplossing te vinden.
2. **Afhankelijkheden en impact**: In complexe projecten kunnen conflicten in één bestand gevolgen hebben voor andere delen in je project. Het oplossen van een merge-conflict in een bestand kan dus invloed hebben op de functionaliteit van andere delen van het project. Dit vereist zorgvuldige aandacht om ervoor te zorgen dat de oplossing geen onbedoelde neveneffecten heeft.
3. **Tijdsdruk**: Merge-conflicten kunnen vertragingen veroorzaken in het ontwikkelproces, vooral als ze complex zijn en extra tijd en inspanning vergen om op te lossen. Dit kan de deadlines van het project beïnvloeden en druk leggen op het ontwikkelteam.
4. **Communicatie en samenwerking**: Merge-conflicten komen vaak voor in teams waar meerdere ontwikkelaars aan hetzelfde project werken. Het effectief communiceren en samenwerken om conflicten op te lossen is essentieel. Het kan uitdagend zijn om duidelijkheid te krijgen over de bedoelingen van verschillende ontwikkelaars en om tot een consensus te komen over de beste oplossing.
5. **Complexe projecten**: In grote projecten met veel bestanden en branches kunnen merge-conflicten complexer worden. Het kan uitdagend zijn om een goed overzicht te krijgen van alle conflicten en om te begrijpen hoe de wijzigingen in verschillende delen van het project met elkaar samenhangen.

Om deze uitdagingen aan te pakken, is het belangrijk om een gestructureerde aanpak te volgen bij het oplossen van merge-conflicten. Dit omvat het begrijpen van de wijzigingen, communiceren met andere ontwikkelaars, het zorgvuldig analyseren van de impact van conflicten en het zorgvuldig testen van de oplossingen om ervoor te zorgen dat er geen onbedoelde gevolgen zijn.

2. Werken met verschillende brances

Het werken met aparte branches in Azure DevOps en Visual Studio 2022 stelt ontwikkelaars in staat om parallel aan het hoofdproject te werken zonder de hoofdcode (master brance) te beïnvloeden. Hier zijn de stappen om in aparte branches te werken:

1. Branch maken: Open Visual Studio 2022 en ga naar het "Git Changes" venster. Klik op "..." en vervolgens op "New Branch". Geef de nieuwe branch een naam die relevant is voor jouw werk, bijvoorbeeld een specifieke functie of bugfix.
2. Switchen naar de nieuwe branch: In het Git Changes kun je kiezen in welke branche je werkt:



3. Wijzigingen aanbrengen: Maak wijzigingen in de code, voeg nieuwe functies toe, pas bestaande code aan, etc. in Visual Studio 2022.

4. Commit wijzigingen: Zodra je tevreden bent met jouw wijzigingen, gebruik je het "Git Changes" venster om de wijzigingen te committen naar jouw lokale branch. Geef een beschrijvende commit-boodschap om de aard van de wijzigingen aan te geven.

5. Push naar de remote repository: Om jouw lokale branch naar de remote repository te pushen, klik je op "Sync" in het "Team Explorer" venster en vervolgens op "Push". Hierdoor worden jouw wijzigingen gedeeld met andere ontwikkelaars die toegang hebben tot de repository.

Voor- en nadelen van werken met aparte branches:

Voordelen:

- **Parallelle ontwikkeling**: Aparte branches stellen ontwikkelaars in staat om tegelijkertijd aan verschillende functies of bugfixes te werken zonder elkaar in de weg te zitten.
- **Experimentatie**: Je kunt nieuwe ideeën en experimenten uitvoeren in een aparte branch zonder de stabiele hoofdcodebase te verstoren.
- **Gecontroleerde releases**: Je kunt aparte branches gebruiken om nieuwe functies te ontwikkelen en testen voordat ze worden samengevoegd met de master branch, waardoor een betere controle over releases ontstaat.

Nadelen:

- **Merge-conflicten**: Bij het samenvoegen van branches kunnen merge-conflicten ontstaan als meerdere ontwikkelaars wijzigingen hebben aangebracht in hetzelfde deel van de code. Deze conflicten moeten handmatig worden opgelost.
- **Overhead**: Werken met meerdere branches kan extra beheer- en coördinatiewerk met zich meebrengen, omdat ontwikkelaars moeten afstemmen welke wijzigingen wanneer worden samengevoegd met de hoofdbranch.
- **Complexiteit**: Het werken met meerdere branches kan de complexiteit van het ontwikkelproces vergroten, vooral als er veel branches zijn en de afhankelijkheden tussen branches ingewikkeld zijn.

 Je kunt nu **oefening 4.1** maken.