Patiëntenbeheer

# Installatie

1. **Start het script “C:\temp\laaddatabase.sq**l” in SQL Server Management Studio 2012

Het script laadt de tabellen en doet een **bulk import** van afdelingen, personeel, postcodes en prestatiesoorten die te vinden zijn onder C:\temp\data.

Dit script werd manueel aangepast om de constraints te verzekeren en geen inconsistenties kunnen ontstaan in de databank. Standaard stond dit uitgeschakeld (WITH NOCHECK)

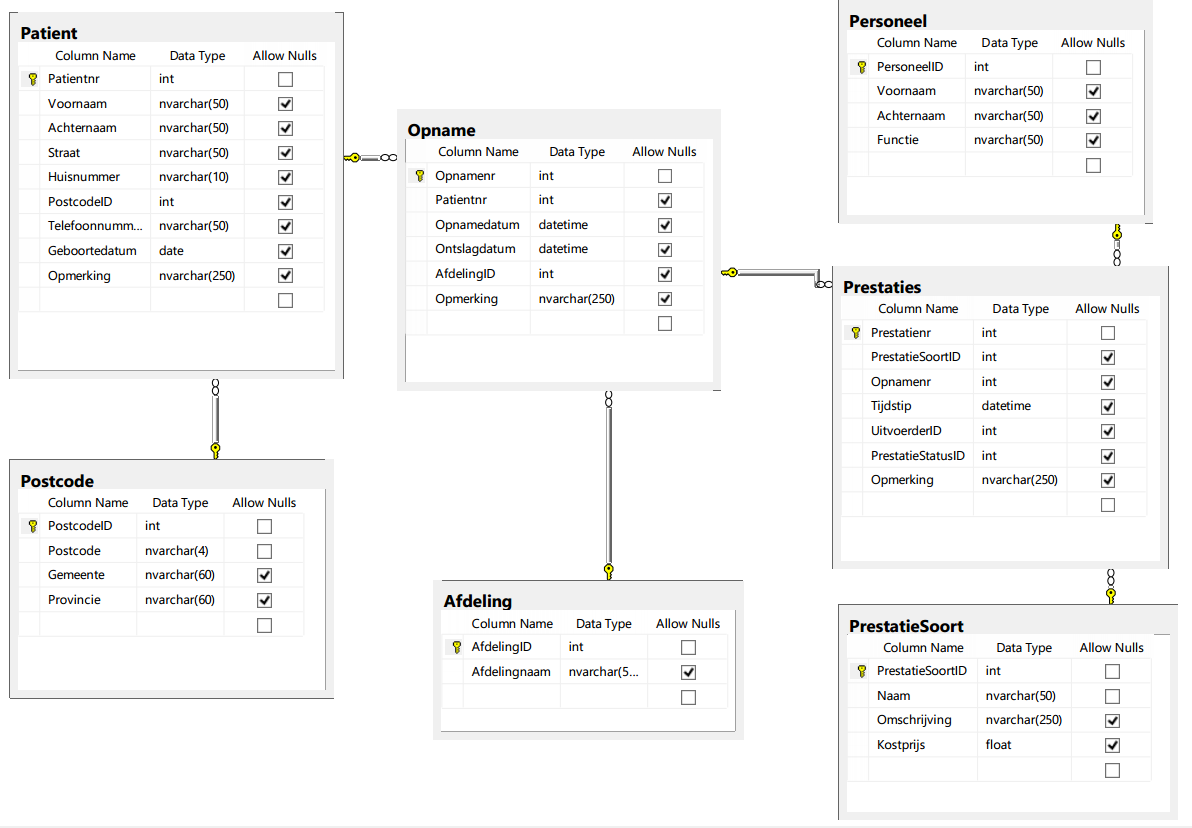
Onderaan de pagina is een diagram van de database en haar relaties terug te vinden.

1. **Laad het project C:\temp\PatientenBeheerPlus\PatientenBeheerPlus.sln** in Microsoft Visual Studio 2012. Het project werd getest met een Express editie van zowel de SQL server als de Visual Studio.
2. **Vullen van de database**

Het is aanbevolen de database eerst enigszins te vullen. Er zitten namelijk geen patiënten in het systeem.

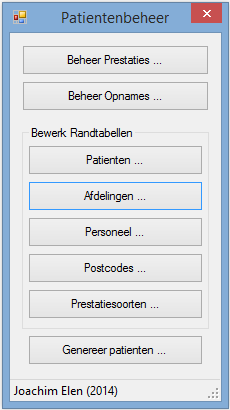
Gebruik daarvoor de **“Genereer patiënten”** toets onderaan het venster. Door een patiënt te genereren worden er eveneens opnames en prestaties toegevoegd met verschillende statussen voor testdoeleinden.

Er wordt voorgesteld 10 à 20 patiënten te genereren.



# Hoofdscherm / aanmelden

Normaal moet men aanmelden om in het juiste scherm terecht te komen, maar zover zijn we niet geraakt in de les, daarom werd dit aanmeldscherm vervangen door onderstaand hoofdscherm:



Voor het eenvoudig bewerken van de randtabellen werd gebruik gemaakt van gegenereerde datasets

Beheer van opnames werd aangevat maar werd niet uitgewerkt

Gevraagd onderdeel, beheer van prestaties

# Patiënten- en opnamebeheer

Geformatteerde lijsten dmv toegevoegde properties in elke tabel klasse.

Om het programma snel te kunnen testen werd er een generator toegevoegd die eenvoudig en willekeurig patiënten, opnames en prestaties genereert.

Overzicht van grootte database

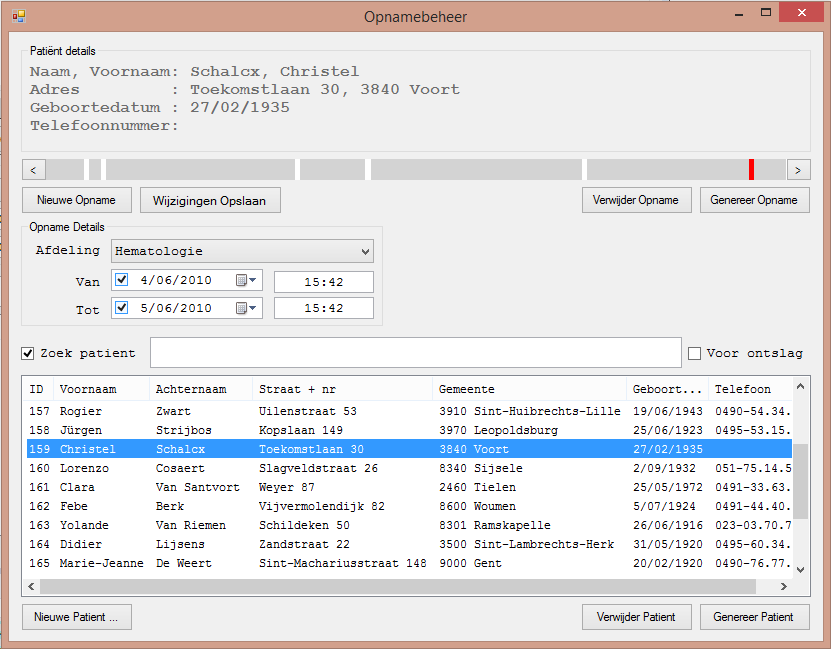
Hieraan werd begonnen, maar omdat dit geen onderdeel van de opdracht uitmaakte werd de uitwerking ervan niet voltooid. Dit onderdeel is dus niet 100% functioneel. Patiënten en opnames kunnen immers ook worden aangemaakt dmv de randtabellen.

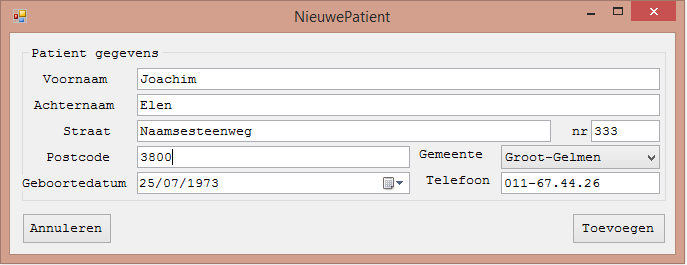
Deze module wordt gebruikt voor:

* Patiëntenbeheer (nieuwe aanmaken, verwijderen).
* Opnamebeheer (nieuwe aanmaken, wijzigen, verwijderen).

Onder de “Patiënt Details” staat een “levenslijn”. Hierop is onmiddellijk te zien wanneer patiënten zijn opgenomen. Elke opname wordt getoond als wit streepje. Selecteren van een opname gebeurt eenvoudig met klikken op de levenslijn.

Levenslijn bevat de opnames, de rode is de geselecteerd opname





# Prestatiebeheer

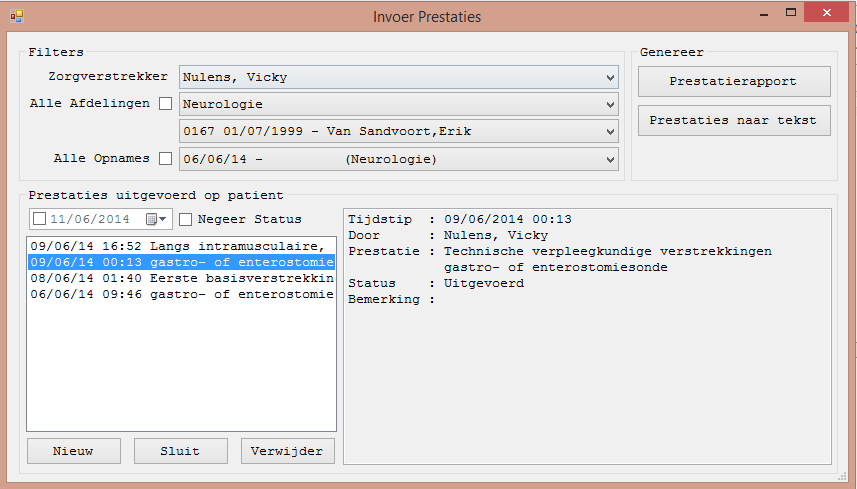
De zorgverstrekker die administratie moet doen moet niet gestoord worden met moeilijke applicaties. Zij zijn er om zorg te verstrekken, niet om met software te spelen. Daarom werd getracht om te denken vanuit hun standpunt.

De zorgverstrekker selecteert haar naam en afdeling. Automatisch krijgt ze de patiënten in alfabetische volgorde te zien die momenteel in de afdeling liggen *(indien aangemeld is de naam gekend en eventueel de afdeling, deze controls kunnen dan wegvallen)*. Ze kan indien gewenst met een vinkje ook de patienten van de andere afdelingen zien. Alle controls worden verduidelijkt door een tooltip.

Om het overzicht over de prestaties te kunnen overzien kan de gebruiker een datum selecteren. Er worden dan enkel prestaties van de die dag getoond. Hierdoor wordt het overzicht beter bewaard.

Bij uitbreiding is het mogelijk dat eerdere opnames worden bekeken door “alle opnames” aan te vinken. Je krijgt een historiek van alle opnames van die patiënt al dan niet op die afdeling.

Prestaties kunnen worden afgesloten door op de “Sluit” toets te drukken. Ze worden dan doorgegeven aan de facturatiedienst voor verdere verwerking. Hiermee doet men dus een **update** van de betreffende prestatie. **Nieuwe** prestaties worden ingegeven in een afzonderlijk venster; Het is uiteraard ook mogelijk prestaties te **verwijderen**.

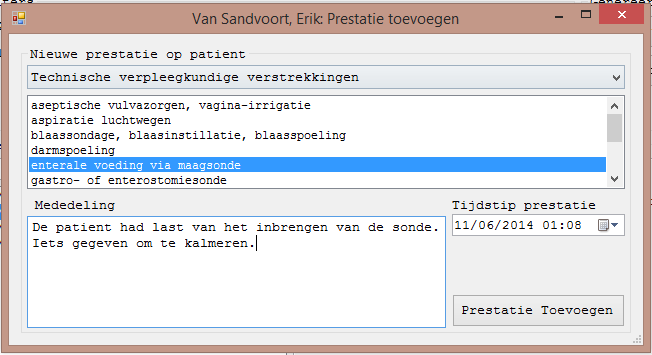


Rapport naar tekstbestand

.ToString()  
Klasse Override

Naam patient in titelbalk van “Nieuwe Prestatie” venster

Prestatiecategorieën



Prestaties werden ondergebracht in prestatiecategorieën. Dit om een overzicht te kunnen houden in de grote verscheidenheid van prestatiesoorten. Zo kan een prestatie vlugger worden gevonden. Hoewel de kostprijs is opgenomen in de database is dit niet relevant voor de zorgverstrekker en wordt niet getoond. Het tijdstip van de prestatie kan uiteraard enkel vallen in het tijdslot van de opname.

Mogelijkheid commentaar toe te voegen

# Rapporten

Om reden dat ik dit niet aan het werk kreeg was ik niet in de mogelijkheid om een rapport toe te voegen zoals bedoeld in de opgave.

De supplementaire vraag voor het schrijven van tekst naar een bestand werd dan beantwoord door het **exporteren van een gedetailleerd prestatieoverzicht van de patiënt tijdens de geselecteerde opname**. Op die wijze kon toch enigszins een rapport worden gegenereerd.

Werkwijze

# 3 Layer approach

De 3-layer approach werd gevolgd door van elke tabel in de database een klasse te maken en een klasseDB. Elke form biedt zijn functionaliteit door hiervan gebruik te maken.

* **Een klasse voor elke tabel uit de database** met fields en properties overeenkomstig de tabel. Om gemakkelijk de juiste lijstweergave te krijgen (voor een combobox bijvoorbeeld) werd aan meerdere klassen de property .ListInfo toegevoegd. Sommige klassen kregen ook andere bijkomende methods of properties toegekend die het programmeren konden vereenvoudigen. Er werd uitgegaan van het object moet maar aanleveren wat ik nogal eens gebruik. Voor detailinformatie werd een override op de .ToString() gebruikt van het object.
* **Voor elke tabelklasse werd er een klasseDB toegevoegd**. Deze heeft typisch volgende methods: Get(), Add(), Update(), Delete(), maar ook hun GetList(), DeleteList(), … varianten waar nuttig.  
  Delete() en DeleteList() zijn uiteraard zo opgemaakt dat er geen residu kan blijven zitten in de database. Het verwijderen van een patiënt betekent dat ook zijn opnames en elke prestatie per opname wordt verwijderd.
* **Formulieren gebruiken de tabelklassen en klassenDB’s voor toegang tot de database**. Zij maken waar mogelijk gebruiken een lokale List<klasse> lijst die wordt gevuld met klasseDB methods om netwerktraffiek te beperken. Bewerkingen op lokale Lists worden meestal doorgevoerd met **Linq**. Zo zijn lijsten eenvoudig te koppelen aan list- en comboboxen.