Event Tracker 1.0

Specificaties

Evert De Schryver

2019

Inhoudstabel

[Probleemdomein 3](#_Toc10996222)

[Omschrijving Probleemdomein 3](#_Toc10996223)

[Authenticatie & Authorisatie 4](#_Toc10996224)

[Procesdiagram 5](#_Toc10996225)

[Use Case Diagram 6](#_Toc10996226)

[Use Cases Overzicht 7](#_Toc10996227)

[Activity Diagrams 8](#_Toc10996228)

[Login 8](#_Toc10996229)

[Reset Password 9](#_Toc10996230)

[Change Password 10](#_Toc10996231)

[View Upcoming Events 11](#_Toc10996232)

[View My Events 11](#_Toc10996233)

[(Un)subscribe 12](#_Toc10996234)

[Add Event 13](#_Toc10996235)

[Edit Event 14](#_Toc10996236)

[Cancel Event 15](#_Toc10996237)

[Delete Event 16](#_Toc10996238)

[View All Users 17](#_Toc10996239)

[Add User 18](#_Toc10996240)

[Edit User 19](#_Toc10996241)

[Delete User 20](#_Toc10996242)

[User Interface 21](#_Toc10996243)

[User Interface Mockups 21](#_Toc10996244)

[Login 21](#_Toc10996245)

[Forgot Password 22](#_Toc10996246)

[Reset Password 23](#_Toc10996247)

[Change Password 24](#_Toc10996248)

[Upcoming Events 25](#_Toc10996249)

[Add Event 26](#_Toc10996250)

[Edit Event 27](#_Toc10996251)

[Users Overview 28](#_Toc10996252)

[Add User 29](#_Toc10996253)

[Edit User 30](#_Toc10996254)

[Storyboards 31](#_Toc10996255)

[Login 31](#_Toc10996256)

[Forgot Password 32](#_Toc10996257)

[Change Password 33](#_Toc10996258)

[Sidebar Navigation 34](#_Toc10996259)

[Event Management 35](#_Toc10996260)

[User Management 36](#_Toc10996261)

[Ontwikkelingstack 37](#_Toc10996262)

[Doelplatform 37](#_Toc10996263)

[Technologieën 37](#_Toc10996264)

[Front end 37](#_Toc10996265)

[Back end 37](#_Toc10996266)

[Tools 37](#_Toc10996267)

[Architectuur 38](#_Toc10996268)

[Application Layers 38](#_Toc10996269)

[Data Access Layer 38](#_Toc10996270)

[De Models Layer 39](#_Toc10996271)

[De Services Layer 40](#_Toc10996272)

[De Web Application Layer 40](#_Toc10996273)

[Class Diagram 43](#_Toc10996274)

[ErrorController 43](#_Toc10996275)

[AccountController 44](#_Toc10996276)

[EventsController 45](#_Toc10996277)

[UserProfilesController 46](#_Toc10996278)

[Database Diagram 47](#_Toc10996279)

[Technologische Uiteenzettingen 48](#_Toc10996280)

[Mobile First 48](#_Toc10996281)

[Extensions 51](#_Toc10996282)

[Alerts 51](#_Toc10996283)

[Validators 53](#_Toc10996284)

[Vraagstukken Tijdens Development 54](#_Toc10996285)

[Timeframes toevoegen 54](#_Toc10996286)

[Overposting 58](#_Toc10996287)

# Probleemdomein

## Omschrijving Probleemdomein

Event Tracker is een responsive, ‘mobile-first’ webapplicatie waarmee een gebruiker, afhankelijk van zijn niveau van autorisatie, evenementen kan creëren, beheren, annuleren en/of verwijderen. Alle gebruikers kunnen zich voor een evenement inschrijven. Door zich in te schrijven, geven zij aan te zullen participeren in de organisatie van het evenement.

Een evenement kan worden gecreëerd door een gebruiker met de daartoe voorziene autoriteit en beschikt over volgende zaken:

* Een naam
* Één of meerdere tijdspannes bestaande uit een datum met daaraan een start- en einduur gekoppeld
* Een locatie bestaande uit een stad en een provincie
* Een korte omschrijving
* Een minimum aantal gewenste participanten
* Een lijst van reeds ingeschreven gebruikers

Gebruikers moeten zich inloggen alvorens ze gebruik kunnen maken van de applicatie. Zij kunnen ook aangeven of ze wensen dat de applicatie hun gebruikersprofiel via een coockie onthoudt. Hierdoor kunnen zij het inlog proces in de toekomst vermijden. Zelf kan men geen gebruikersprofiel aanmaken. Dit gebeurt door een administrator. Eens een nieuw gebruikersprofiel is aangemaakt, stuurt de applicatie een confirmatiemail naar het email adres dat gebruikt is tijdens het aanmaken van de gebruiker. Een gebruiker kan zich niet inloggen zolang hij zijn profiel niet heeft geactiveerd door de instructies in de confirmatiemail te volgen.

Een gebruikersprofiel beschikt over volgende zaken:

* Voornaam
* Familienaam
* Email
* Paswoord
* Gebruikersrol

Een gebruiker krijgt na het inloggen een overzicht te zien van alle toekomstige evenementen. Evenementen uit het verleden worden niet weergegeven door de applicatie. Alle gebruikers kunnen zich voor elk getoond evenement in- of uitschrijven tenzij het evenement geannuleerd is. Er staat geen maximum op het aantal inschrijvingen. De applicatie zal wel visueel aanduiden wanneer het gewenst aantal participanten is bereikt. Een gebruiker kan ook een overzicht krijgen van alle evenementen waarop hij zich heeft ingeschreven.

Een evenement kan door een superuser geannuleerd worden. Wanneer dit het geval is, zal de applicatie dit visueel aanduiden. Een gebruiker kan zich in dit geval niet meer inschrijven voor het evenement. Ook kan een superuser de details van een geannuleerd evenement niet langer aanpassen of het evenement verwijderen. Een geannuleerd evenement kan door een superuser wel terug beschikbaar worden gesteld. Eens dit is gebeurd, kunnen alle functionaliteiten terug worden toegepast.

Een verder overzicht van de verschillende gebruikersrollen met de daaraan gekoppelde autorisatie kan samen met een diepere omschrijving van het authenticatieproces gevonden worden in het hoofdstuk ‘[Authenticatie & Authorisatie](#_Authenticatie_&_authorisatie)’.

## Authenticatie & Authorisatie

Een gebruikersprofiel kan alleen worden aangemaakt door een administrator. Een voor de applicatie ongekende gebruiker kan zelf geen gebruikersprofiel aanmaken. Iemand die toegang wenst tot applicatie moet een aanvraag indienen bij een administrator. Hierbij geeft men een geldig emailadres dat men wenst te koppelen aan het gebruikersprofiel. Dit emailadres wordt tevens gebruikt als unieke gebruikersnaam. Dit betekent dat de applicatie geen nieuw gebruikersprofiel zal aanvaarden waarvan het opgegeven emailadres reeds gekoppeld is aan een ander gebruikersprofiel.

De administrator voorziet de nieuwe gebruiker van een paswoord. Dit paswoord is onderworpen aan volgende vereisten:

* Minimum 8 karakters
* Minimum 1 uppercase karakter
* Minimum 1 lowercase karakter
* Minimum 1 cijfer
* Minimum 1 punctuatie- of symboolkarakter

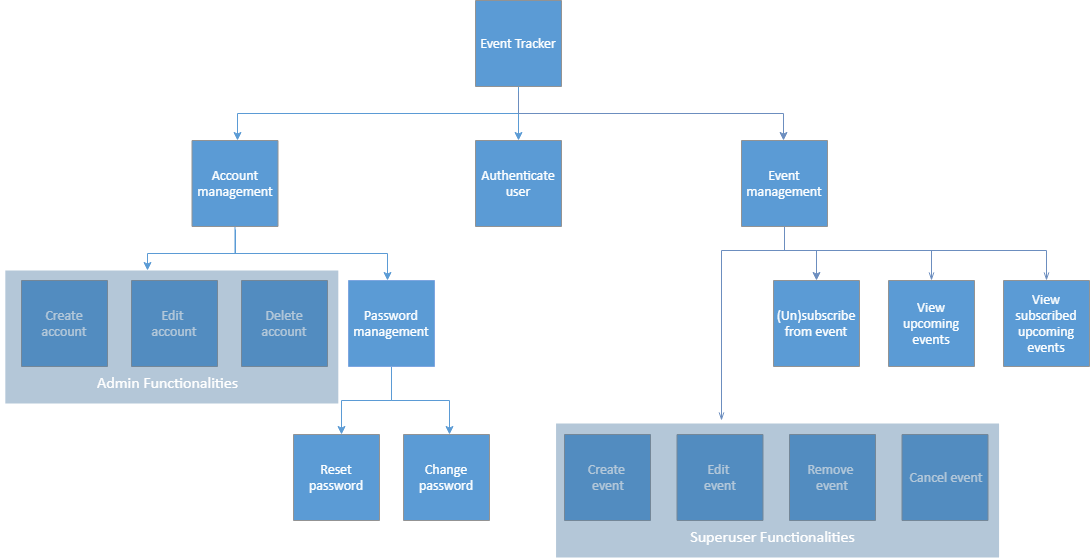
Eens een gebruikersprofiel is aangemaakt, zal de applicatie een confirmatiemail sturen naar het opgegeven email adres. Het doel van deze mail is om er zeker van te zijn dat het opgegeven emailadres daadwerkelijk bestaat. De mail bevat een confirmatie link. Wanneer de gebruiker op deze link klikt, zal zijn gebruikersprofiel geactiveerd zijn. De link is 24 uur geldig, hierna vervalt hij en kan hij bijgevolg niet meer gebruikt worden om het gebruikersprofiel te activeren. Een gebruiker kan zich pas inloggen indien zijn gebruikersprofiel geactiveerd is.

Een administrator voorziet een gebruiker tenslotte van een gebruikers. Volgende rollen kunnen worden toegewezen:

* Basic user:
  + Zij kunnen zich inloggen
  + Zij kunnen een overzicht krijgen van alle toekomstige evenementen
  + Zij kunnen een overzicht krijgen van alle toekomstige evenementen waarop zij zich hebben ingeschreven
  + Zij kunnen zich in- of uitschrijven voor evenementen
  + Zij kunnen het paswoord gelinkt aan hun profiel wijzigen of resetten
* Superuser:
  + Zij kunnen alles wat een basic user kan
  + Zij kunnen evenementen creëren, aanpassen, annuleren en verwijderen
  + Zij kunnen voor elk evenement deelnemers verwijderen
* Administrator:
  + Zij kunnen alles wat superusers kunnen
  + Zij kunnen gebruikersprofielen creëren, aanpassen en verwijderen

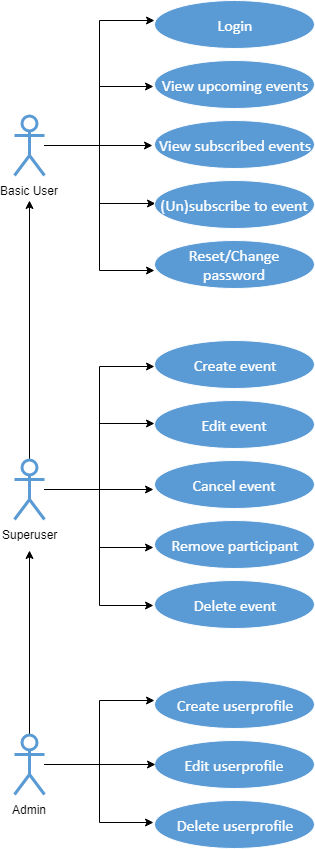
## Procesdiagram

Onderstaand diagram schets een beeld van alle processen die plaats vinden binnen de context van de applicatie.



## Use Case Diagram

Onderstaand diagram schets een beeld van alle use cases die plaats vinden binnen de context van de applicatie:



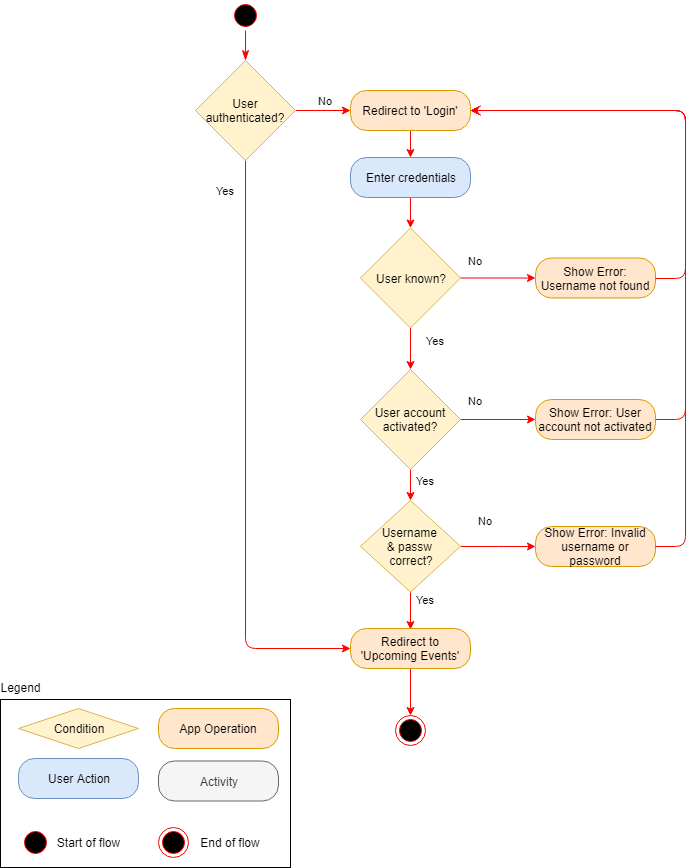
## Use Cases Overzicht

Onderstaand overzicht schets een beeld van alle use cases die plaats vinden binnen de context van de applicatie.

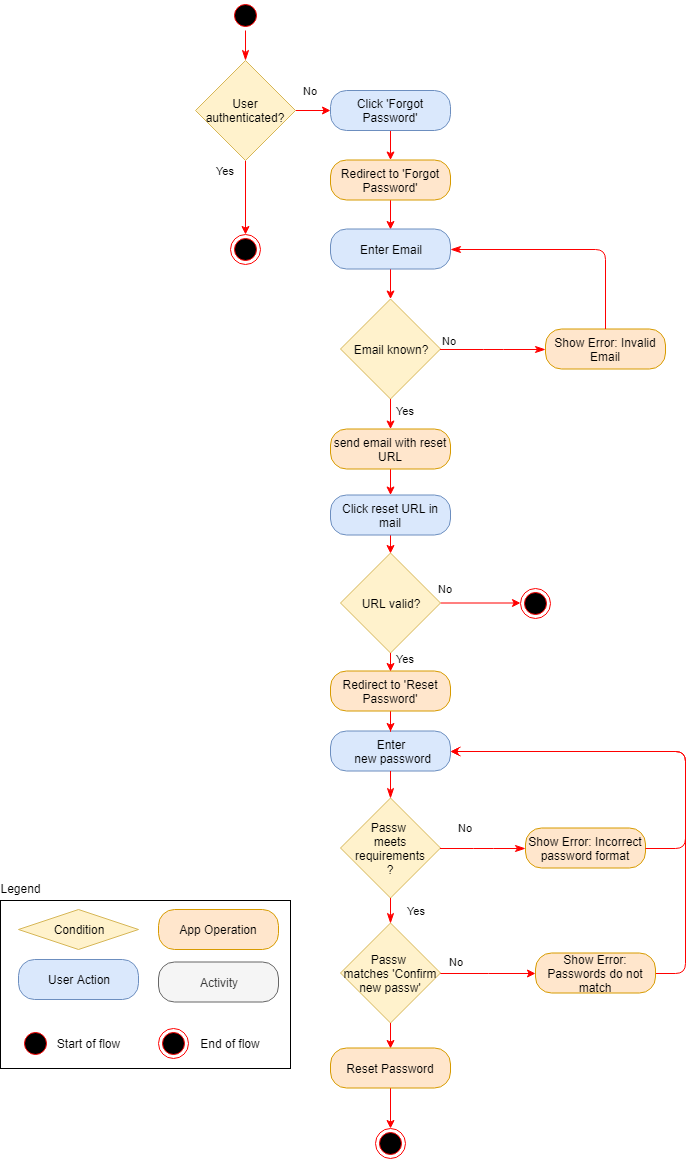
|  |  |  |
| --- | --- | --- |
| Als een… | Zou ik graag… | Zodat ik… |
| Basic user | Inloggen | Gebruik kan maken van de applicatie |
| Basic user | Een lijst krijgen van alle toekomstige evenementen | Kan zien welke evenementen er gepland staan |
| Basic user | Een lijst krijgen van alle toekomstige evenementen waarvoor ik me heb ingeschreven | Een overzicht heb van alle toekomstige evenementen waar mijn aanwezigheid wordt verwacht |
| Basic user | Mezelf inschrijven voor een evenement | Mijn aanwezigheid kenbaar kan maken |
| Basic user | Mezelf uitschrijven voor een evenement | Mijn afwezigheid kenbaar kan maken |
| Basic user | Mijn paswoord wijzigen | Goede beveiligingsnormen kan hanteren |
| Basic user | Mijn paswoord resetten | Een nieuw paswoord kan instellen indien ik het vergeten ben |
| Superuser | Evenementen aanmaken | Nieuwe evenementen beschikbaar kan stellen voor gebruikers |
| Superuser | Evenementen aanpassen | De praktische info van een evenement kan wijzigen |
| Superuser | Evenementen verwijderen | Onnodige evenementen kan verwijderen |
| Superuser | Evenementen annuleren | Gebruikers op de hoogte kan stellen wanneer een evenement niet doorgaat |
| Admin | Gebruikersprofielen aanmaken | Aan gebruikers de mogelijkheid geef gebruik te maken van de applicatie |
| Admin | Gebruikersprofielen verwijderen | Bepaalde gebruikers de toegang tot de applicatie kan ontzeggen |
| Admin | Gebruikersprofielen aanpassen | De gebruikersrol, email of naam van een gebruiker kan wijzigen |

## Activity Diagrams

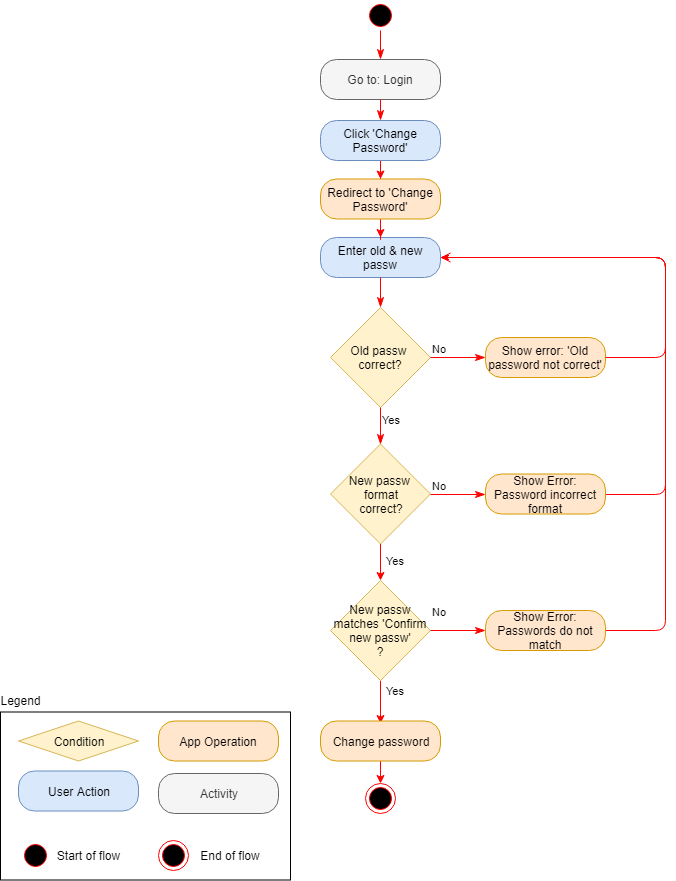
##### Login



##### Reset Password



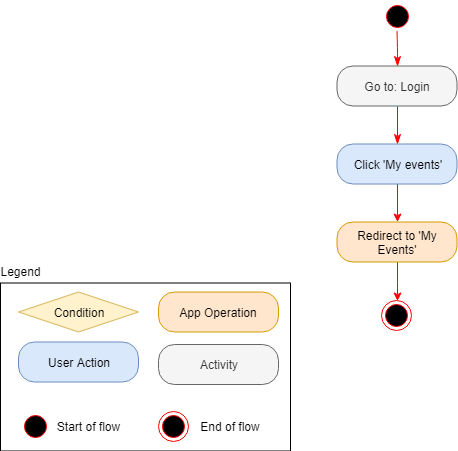
##### Change Password



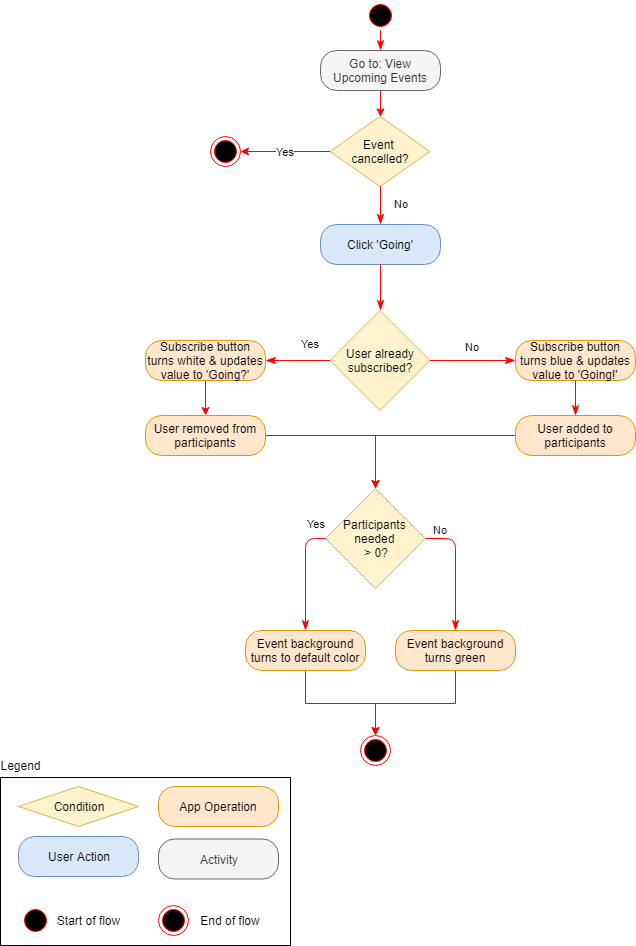
##### View Upcoming Events



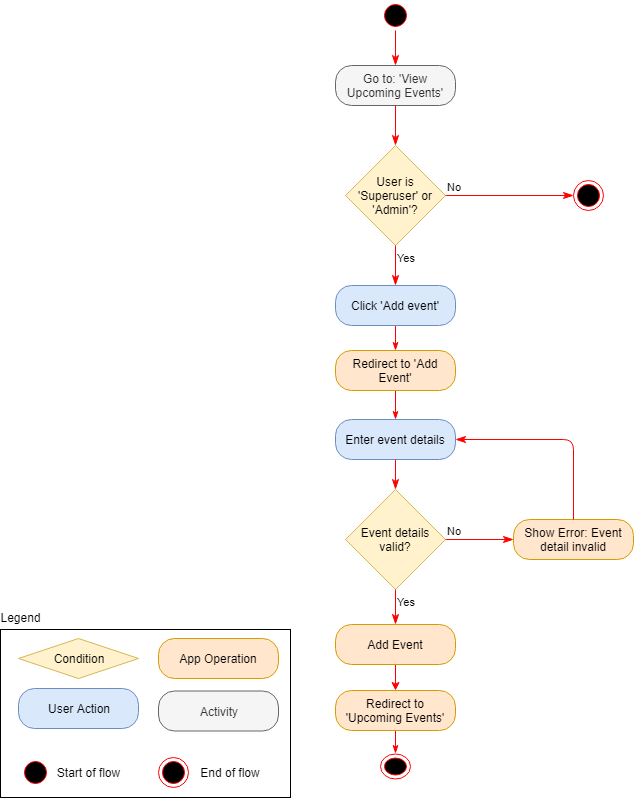
##### View My Events



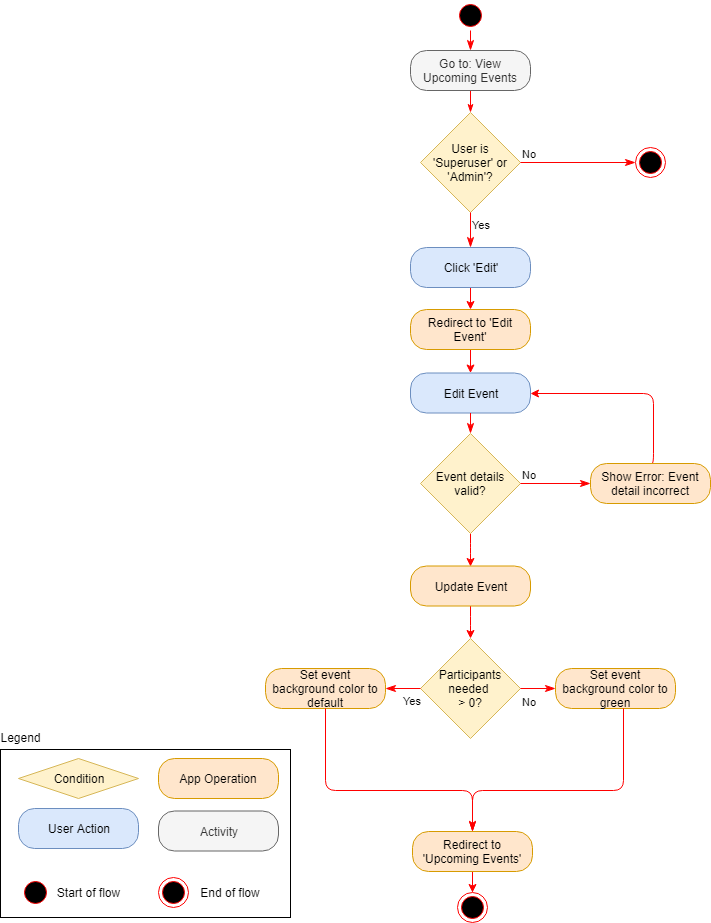
##### (Un)subscribe



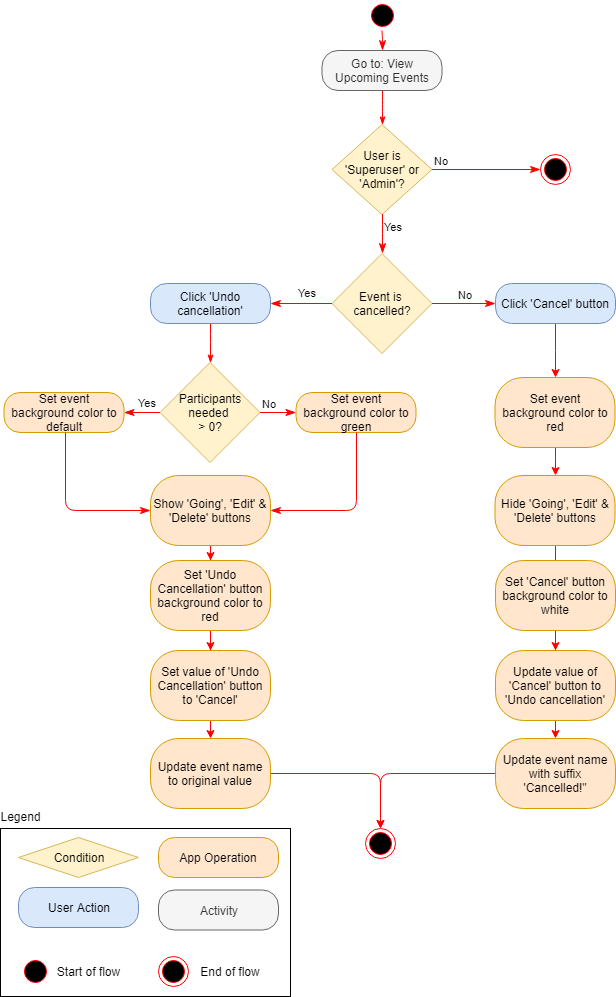
##### Add Event



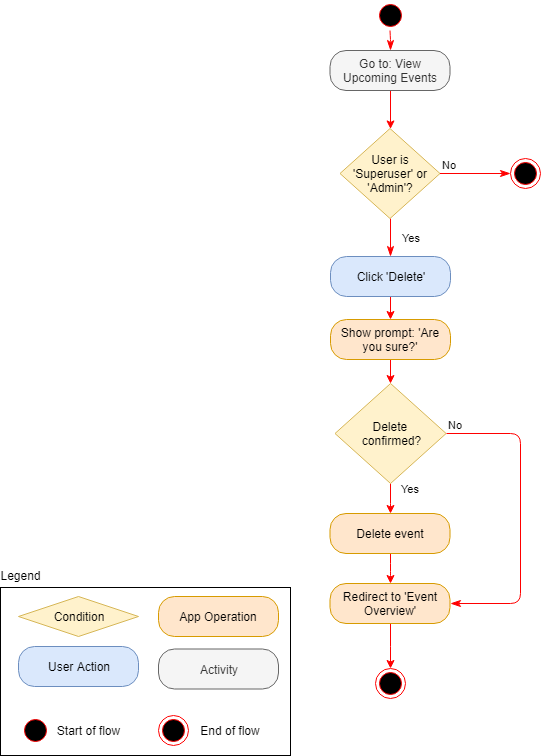
##### Edit Event



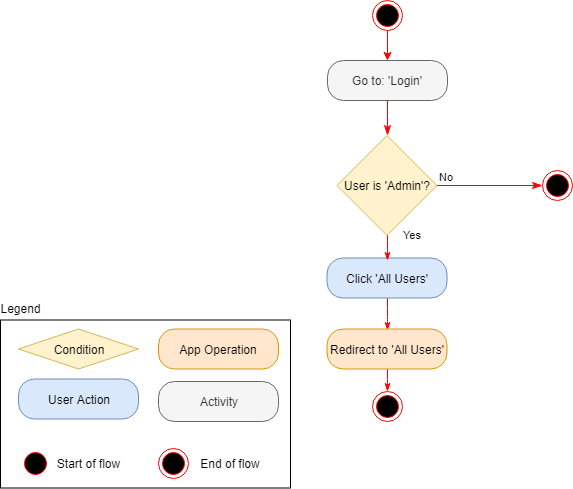
##### Cancel Event



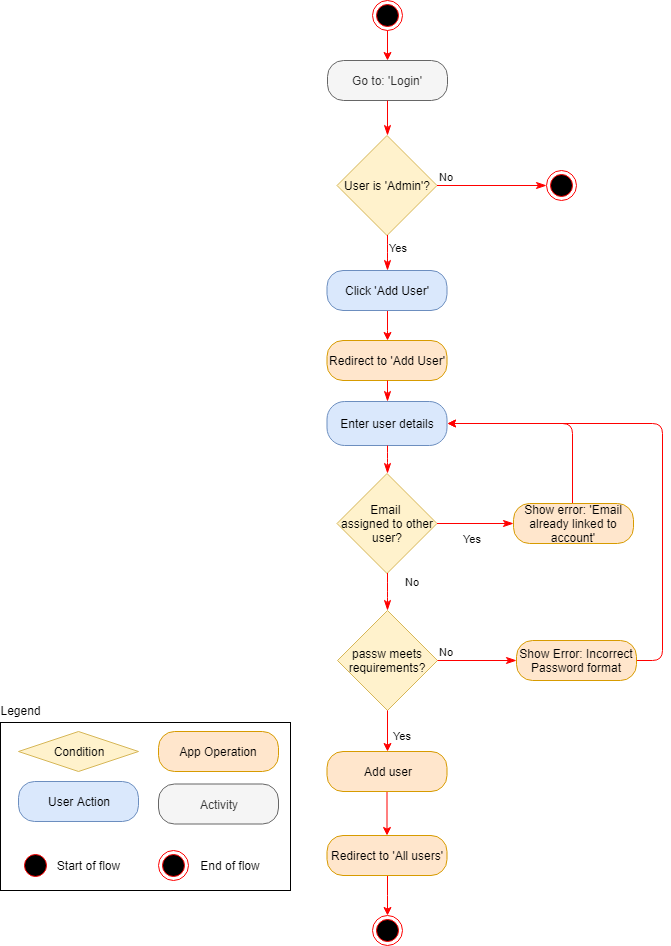
##### Delete Event



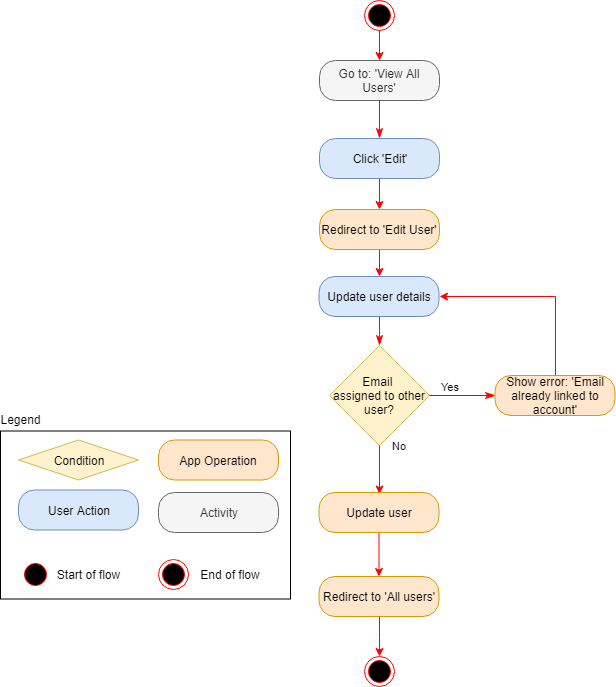
##### View All Users



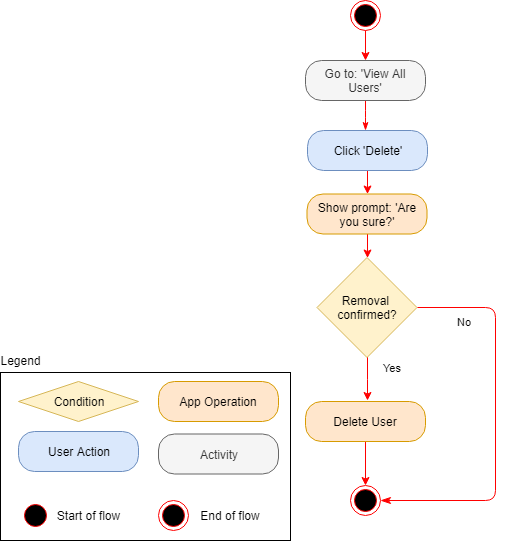
##### Add User



##### Edit User



##### Delete User

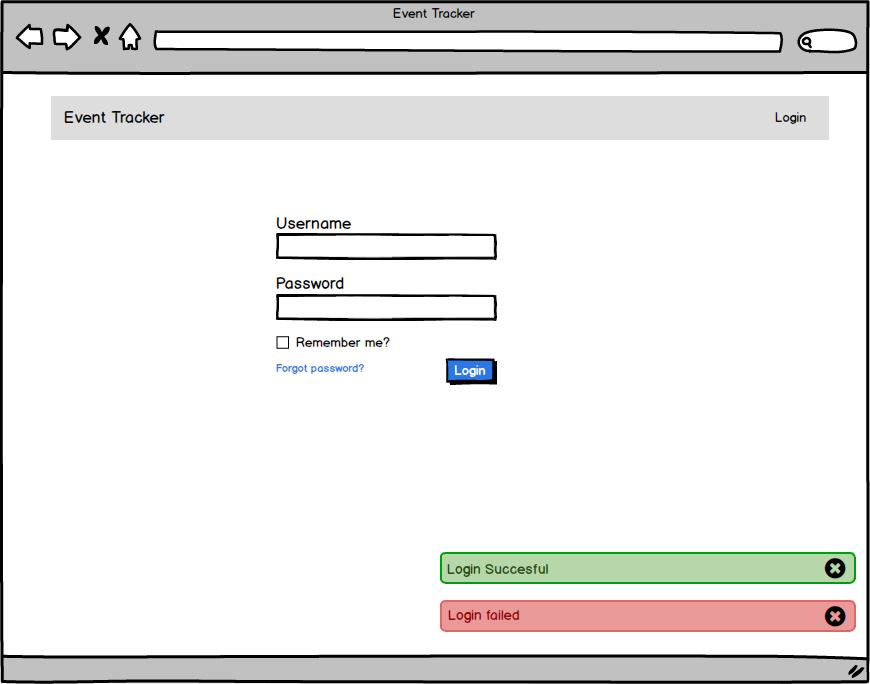


# User Interface

## User Interface Mockups

##### Login

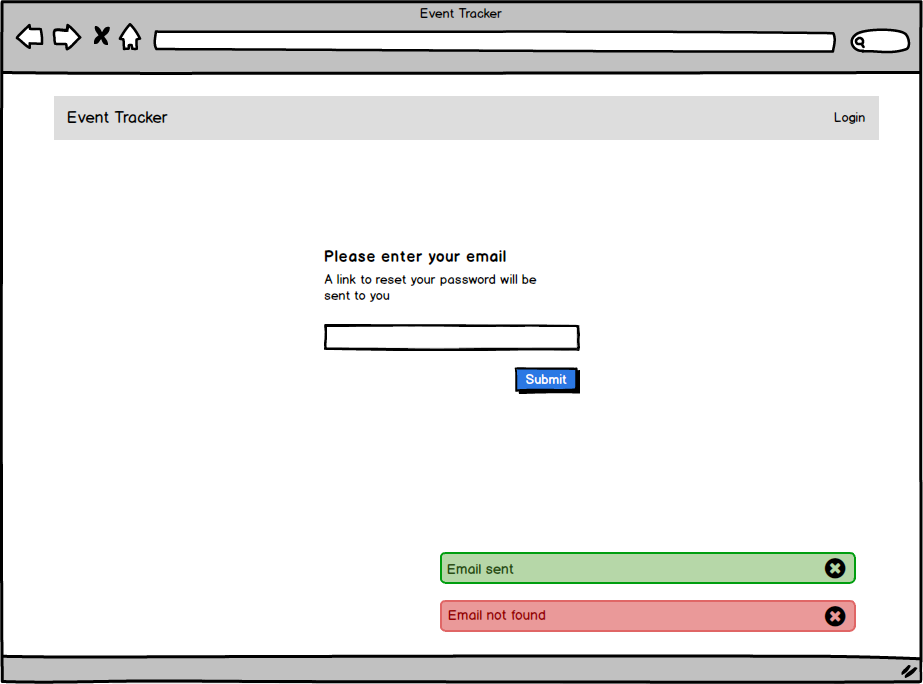
Het login scherm is beschikbaar voor alle gebruikers.





##### Forgot Password

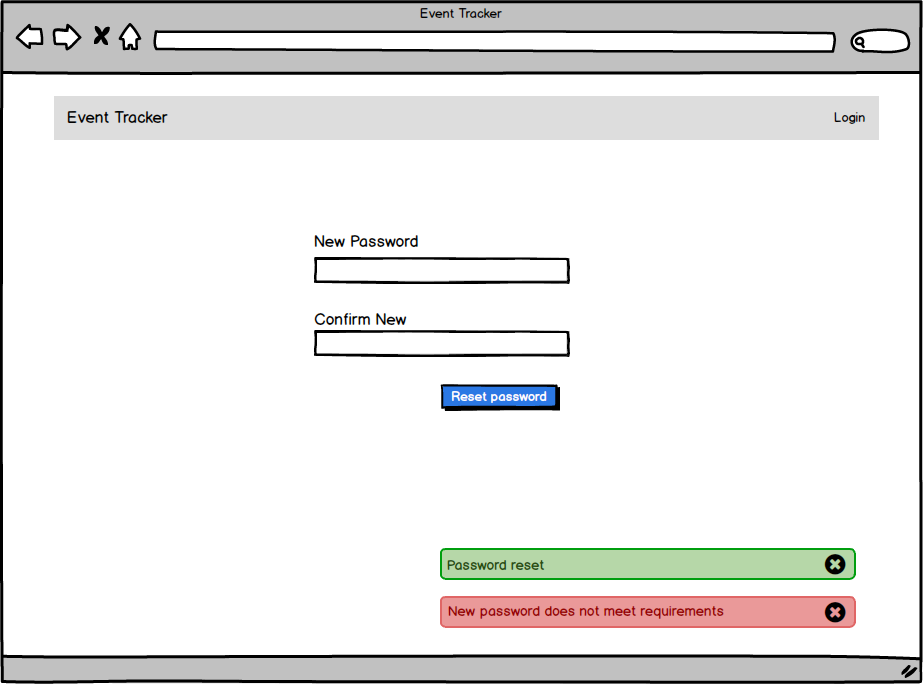
Het ‘Forgot password’ scherm is beschikbaar voor alle gebruikers.

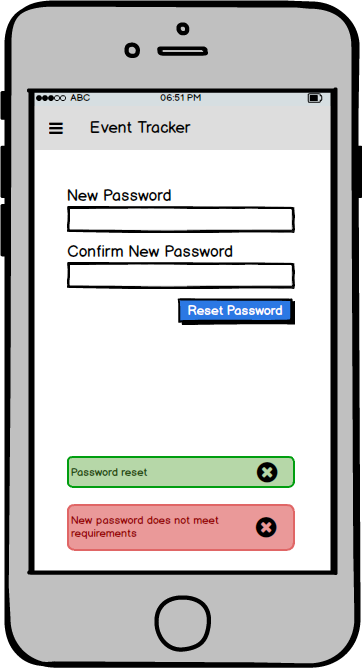




##### Reset Password

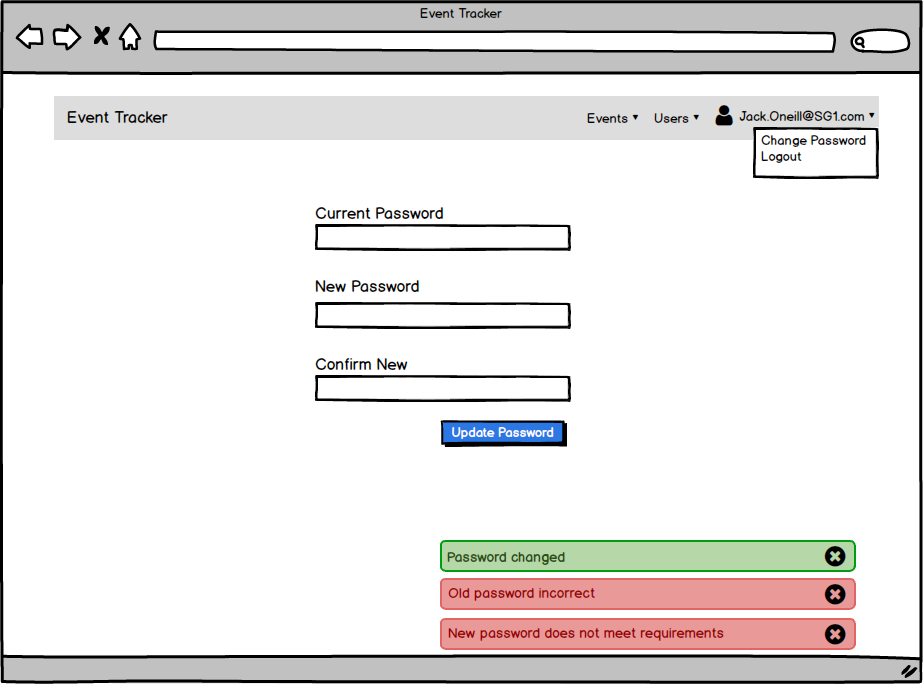
Het reset password scherm is toegankelijk voor alle gebruikers. Dit scherm kan enkel bereikt worden via een door de applicatie gegenereerde URL die per mail naar de user wordt verstuurd nadat die een geldige email heeft ingevoerd in het ‘Forgot Password’ scherm.

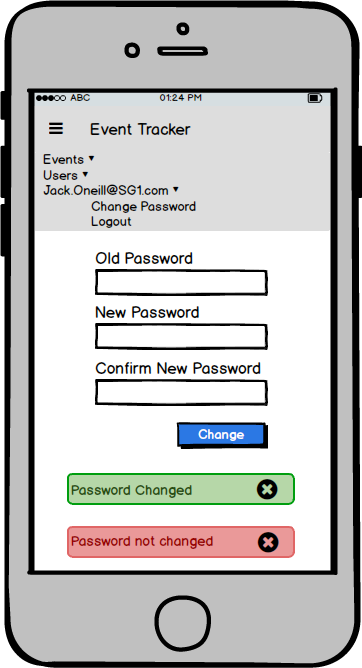




##### Change Password

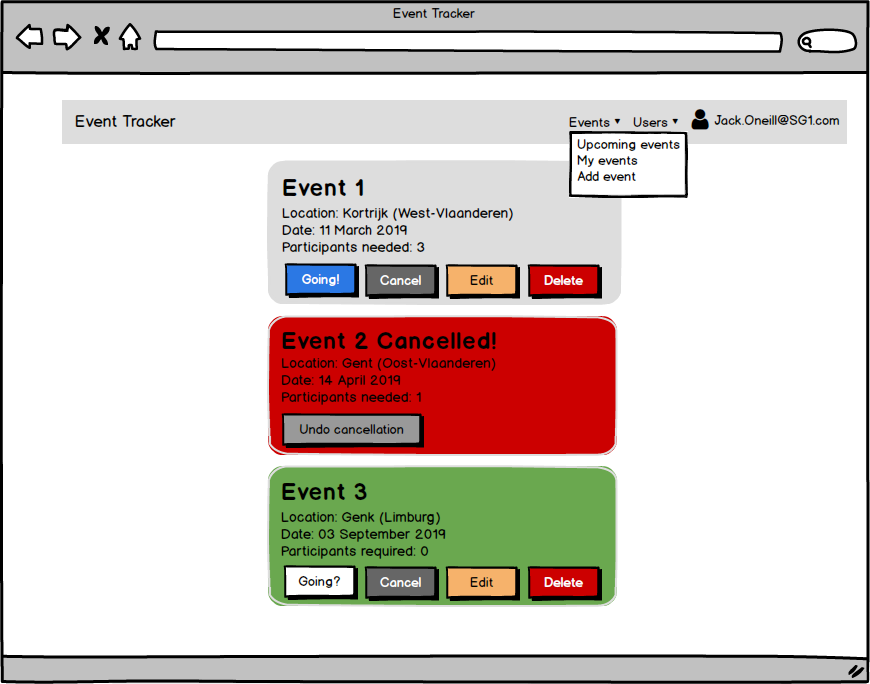
* Het ‘Change password’ scherm is beschikbaar voor alle gebruikers.
* Het dropdown menu ‘Users’ is enkel zichtbaar voor administrators.





##### Upcoming Events

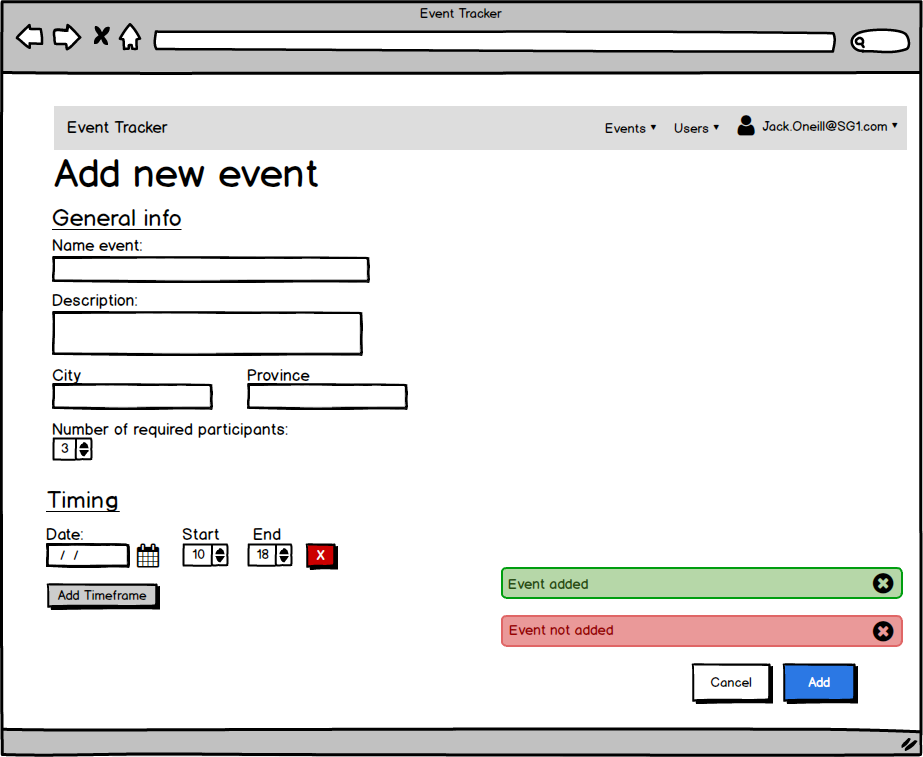
* Het ‘Upcoming Events’ scherm is beschikbaar voor alle gebruikers.
* Het dropdown menu ‘Users’ is enkel zichtbaar voor administrators.
* Deze lay-out wordt gehanteerd bij zowel ‘Upcoming Events’ als bij ‘My Events’. Enkel de buttons bij een evenement zullen in het ‘My Event’ scherm niet verschijnen.
* De link ‘Add event’ en de buttons ‘Cancel’, ‘Edit’, ‘Delete’ & ‘Undo cancellation’ zijn enkel zichtbaar voor superusers & administrators

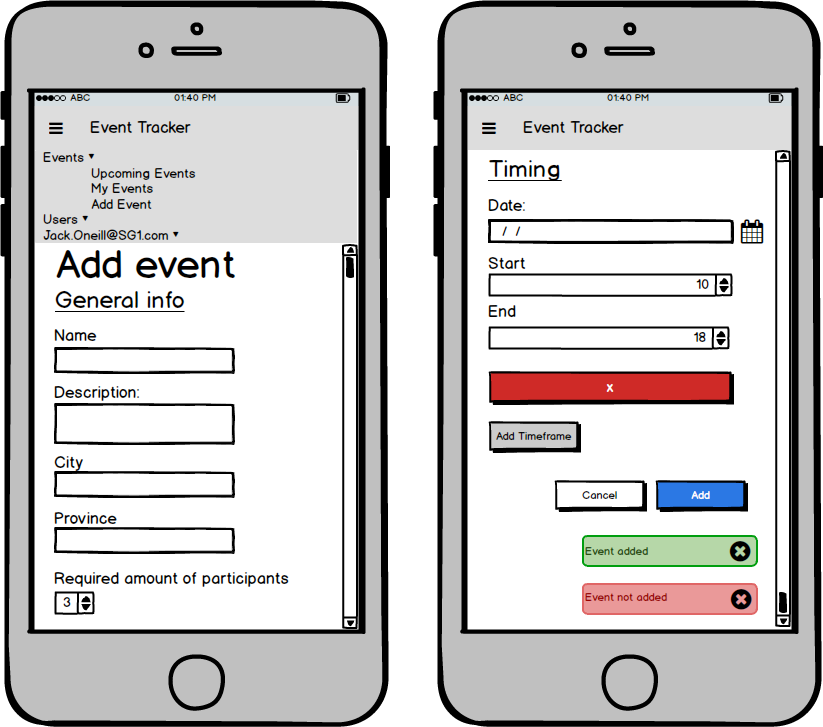




##### Add Event

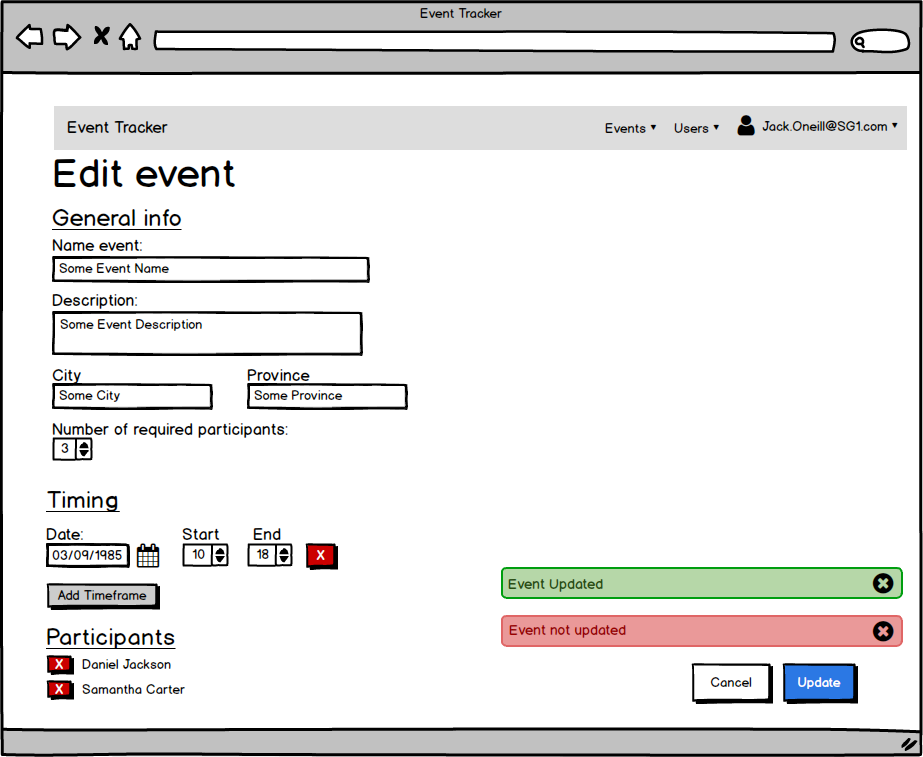
* Het ‘Add Event’ scherm is enkel beschikbaar voor superusers en administrators.
* Het dropdown menu ‘Users’ is enkel zichtbaar voor administrators.

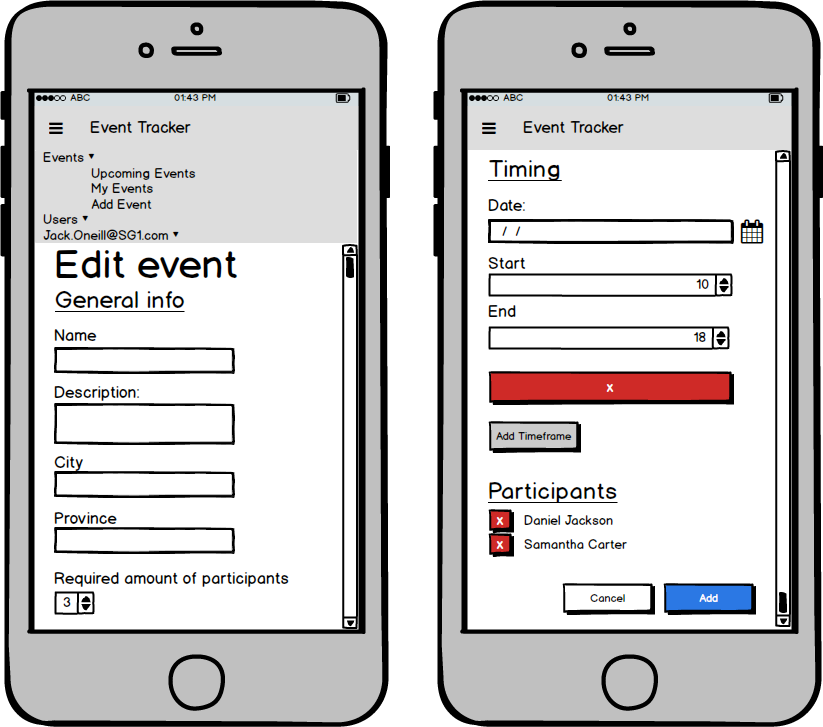




##### Edit Event

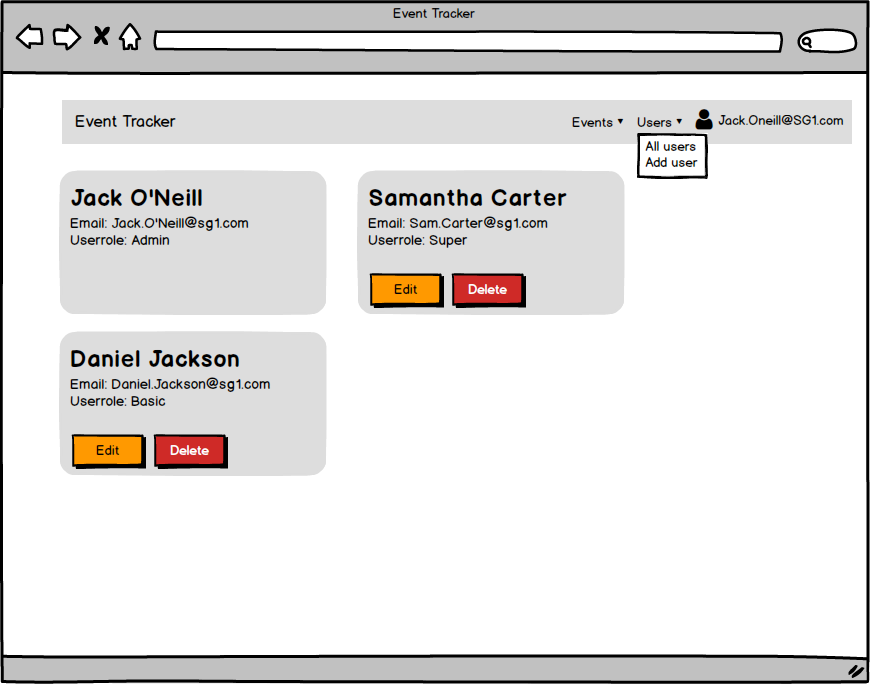
* Het ‘Add Event’ scherm is enkel beschikbaar voor superusers en administrators.
* Het dropdown menu ‘Users’ is enkel zichtbaar voor administrators.

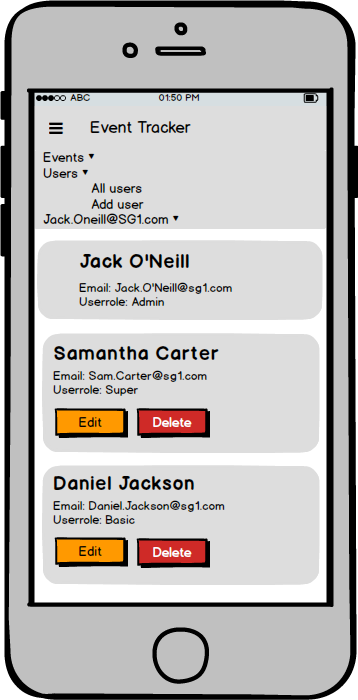




##### Users Overview

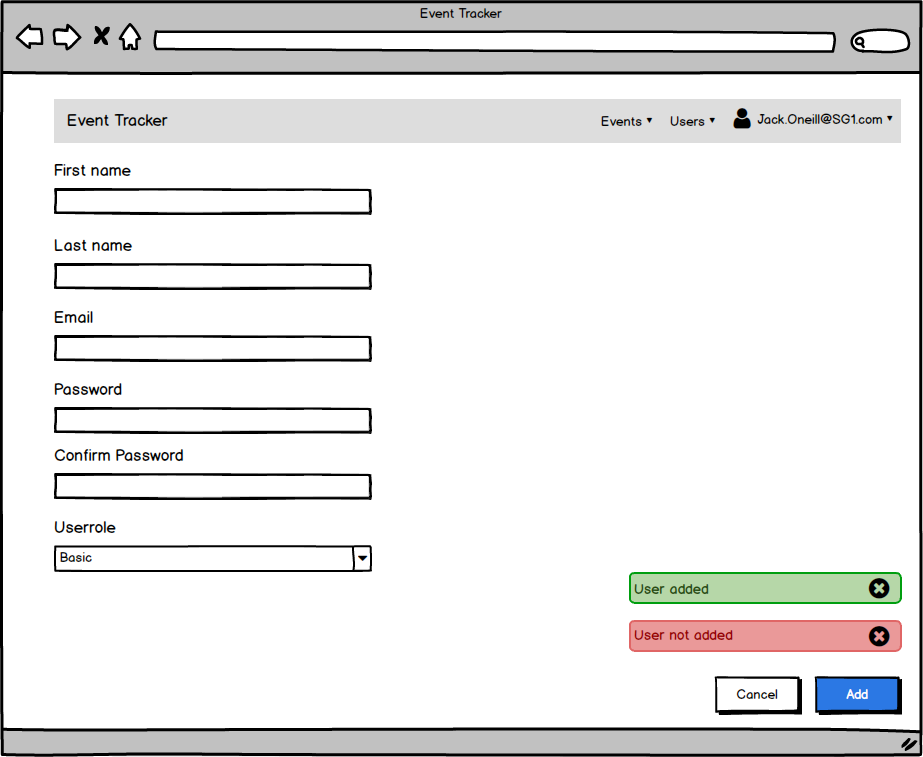
* Het ‘Users Overview’ scherm is enkel beschikbaar voor administrators.
* Het dropdown menu ‘Users’ is enkel zichtbaar voor administrators.
* Een administrator kan zijn eigen profiel niet verwijderen of aanpassen. De nodige buttons hiervoor verschijnen immers niet.

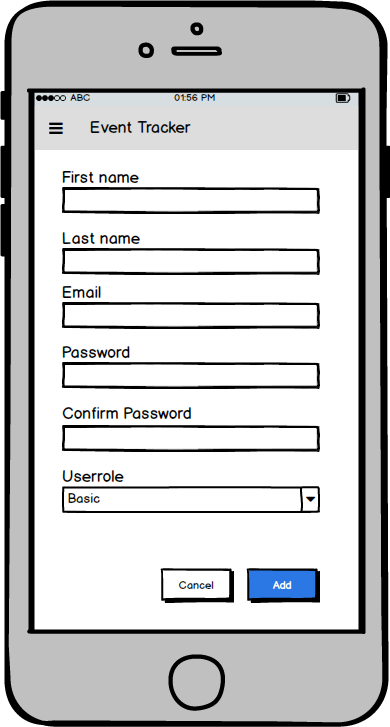




##### Add User

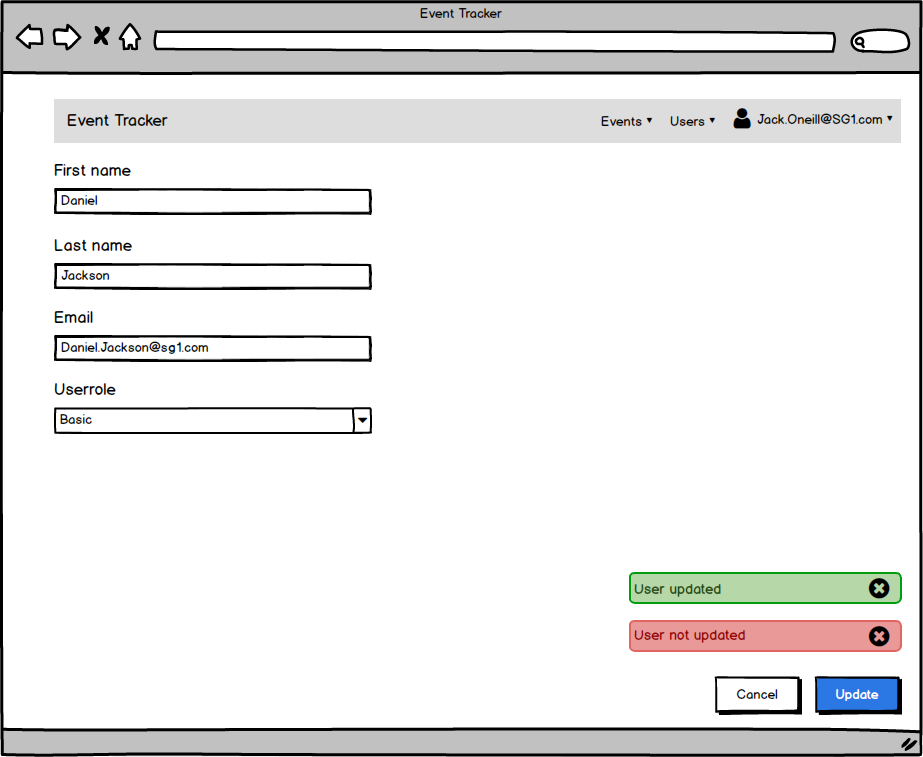
* Het ‘Add user’ scherm is enkel beschikbaar voor administrators.
* Het dropdown menu ‘Users’ is enkel zichtbaar voor administrators.

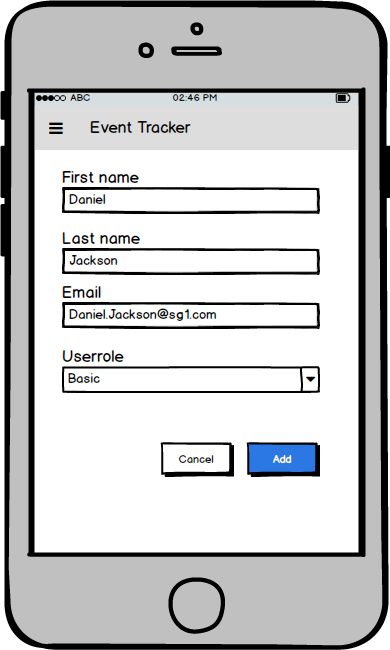




##### Edit User

* Het ‘Edit user’ scherm is enkel beschikbaar voor administrators.
* Het dropdown menu ‘Users’ is enkel zichtbaar voor administrators.

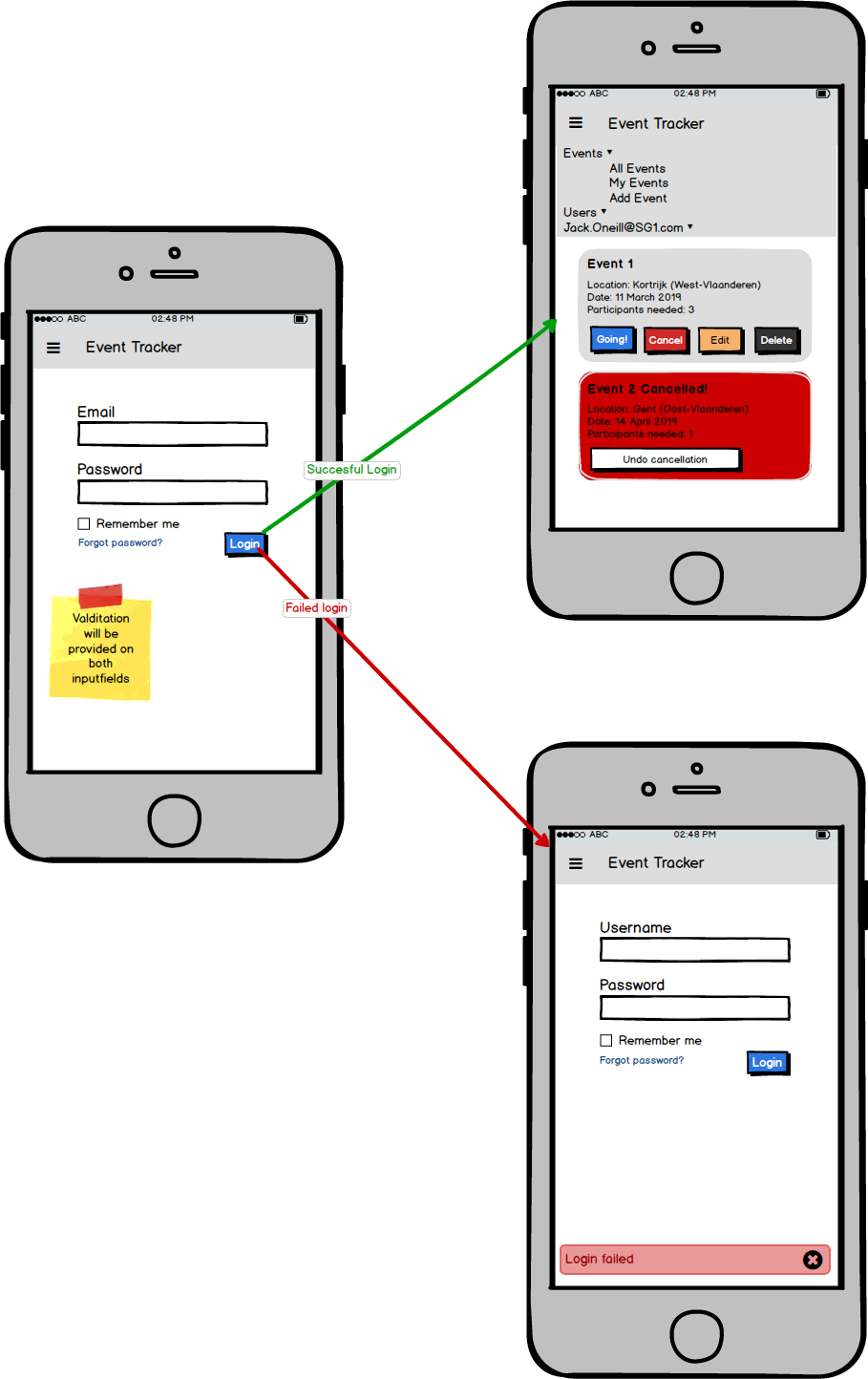




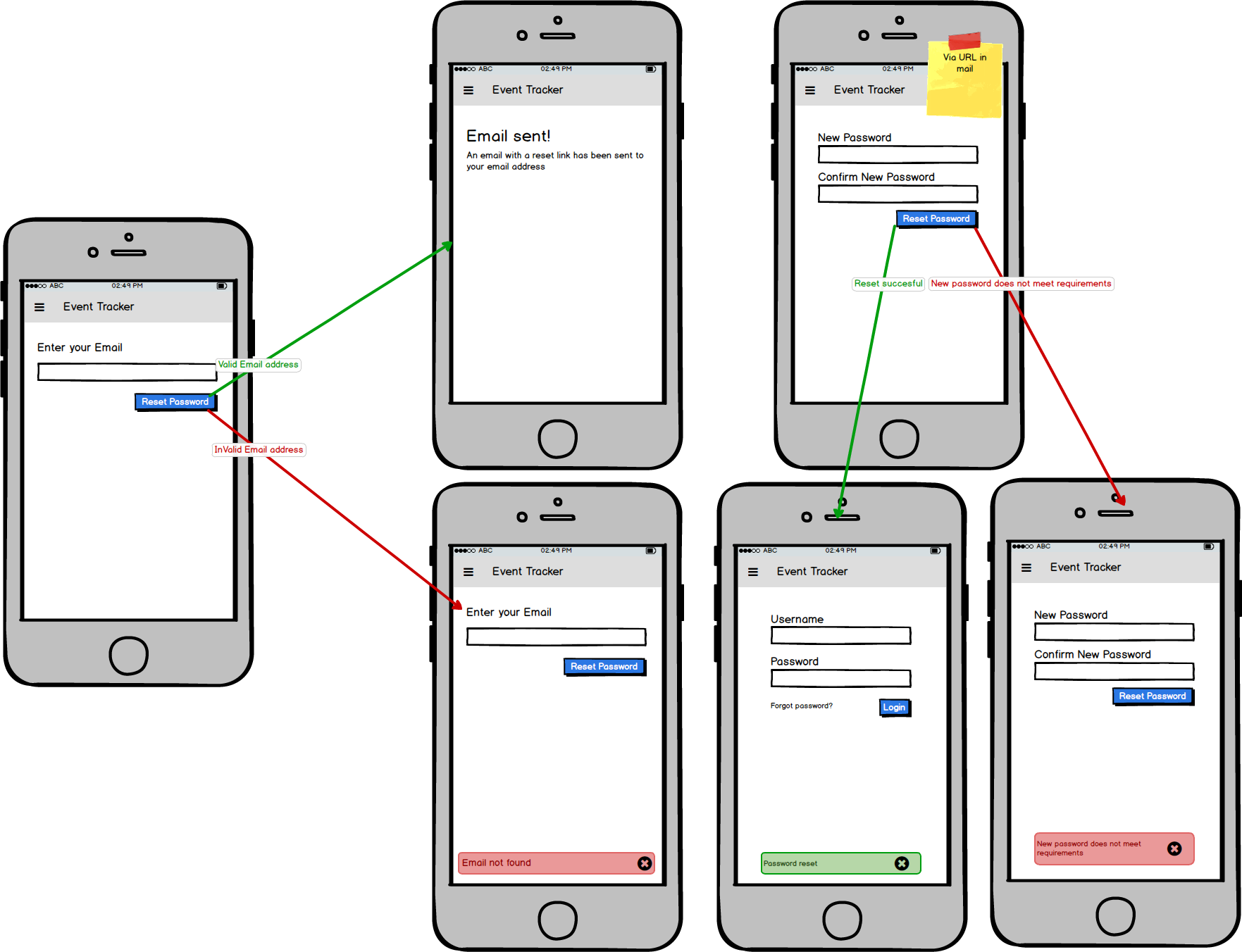
## Storyboards

Op vlak van navigatie is er geen wezenlijk verschil tussen de mobiele UIs en desktop UIs. Om het simpel te houden, wordt bijgevolg enkel de mobiele UI gebruikt bij onderstaande storyboards. Tevens bevatten alle UIs in onderstaande storyboards alle functionaliteiten die de applicatie aanbiedt. Sommige controls of UIs zijn dus niet toegangkelijk voor alle gebruikers. Dit is waar nodig gespecifieerd.

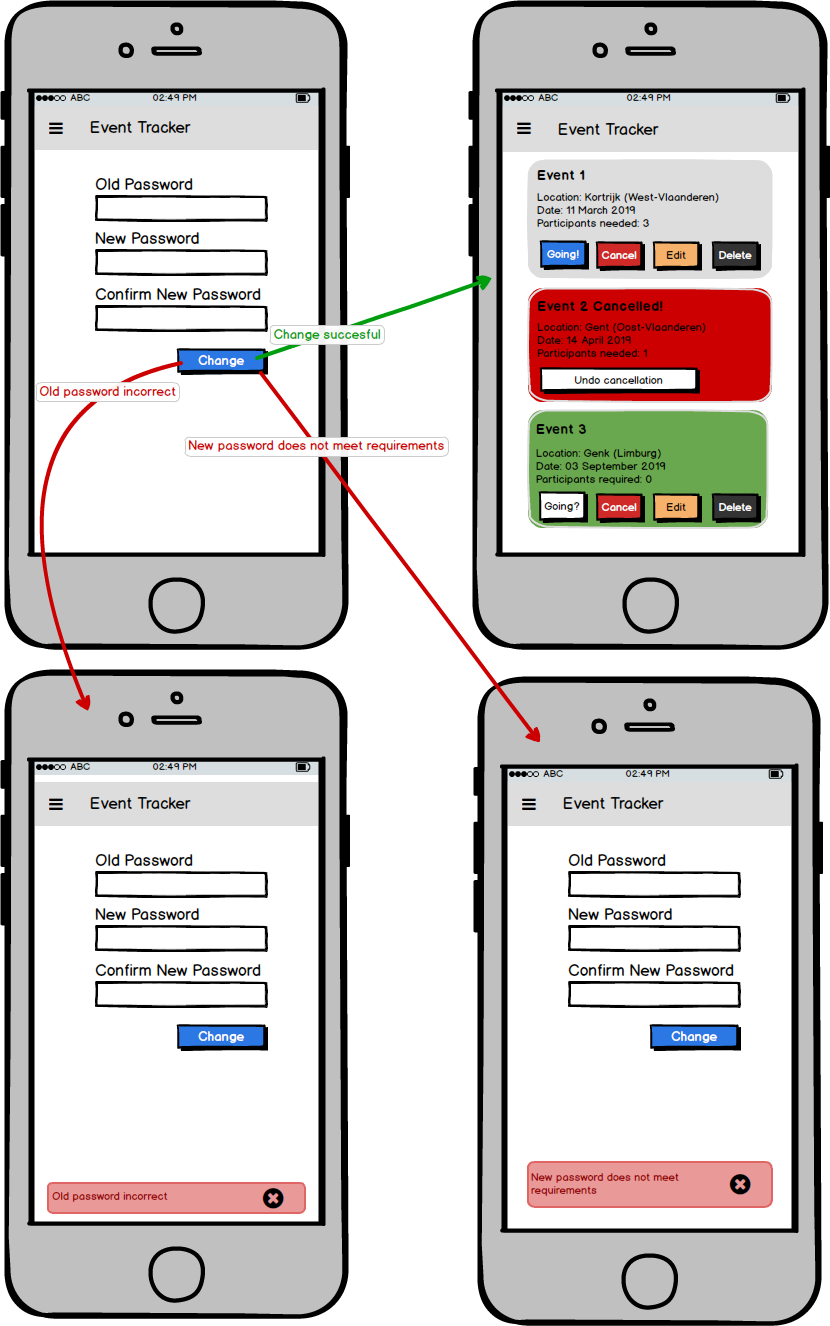
##### Login



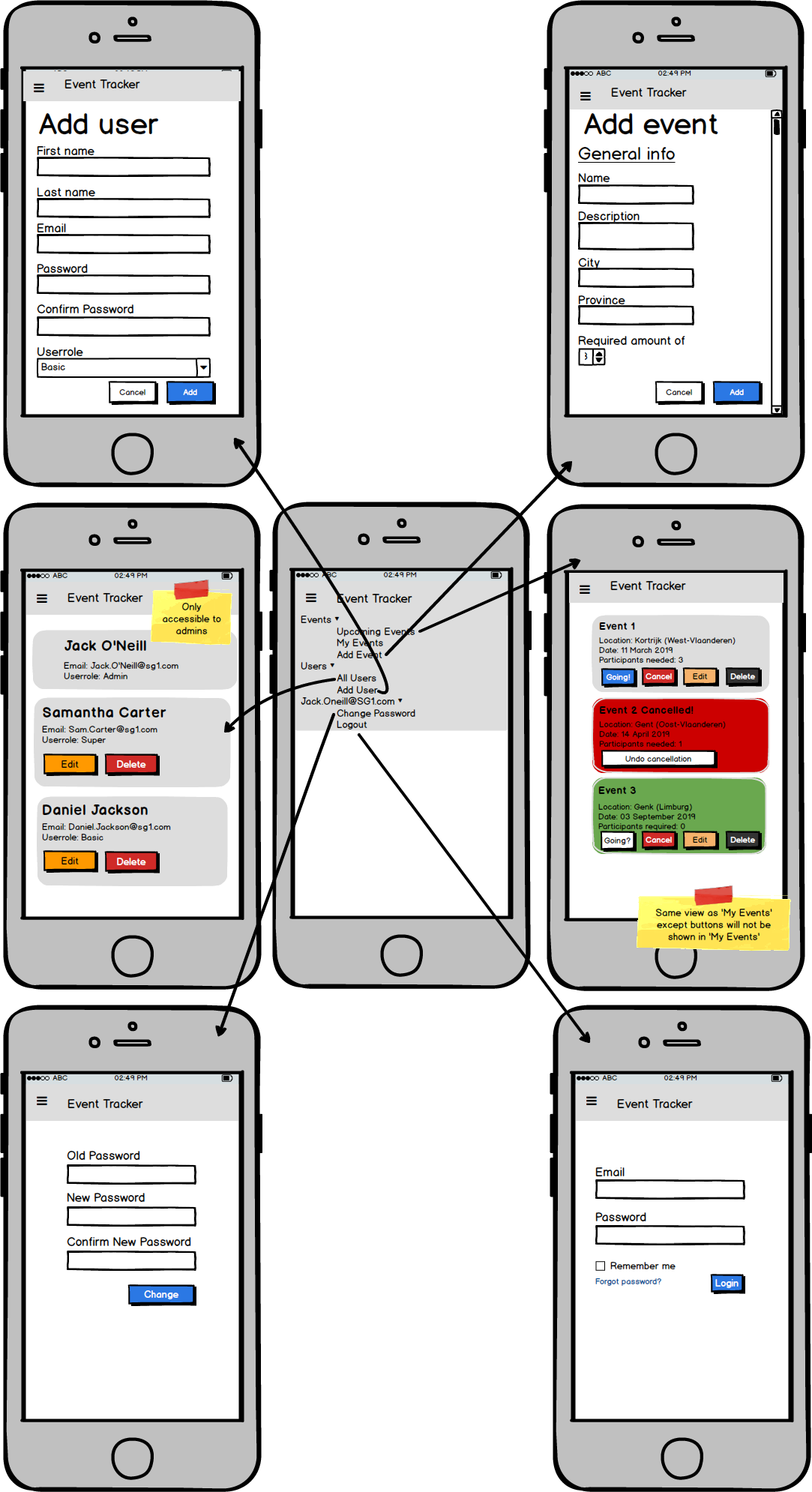
##### Forgot Password



##### Change Password

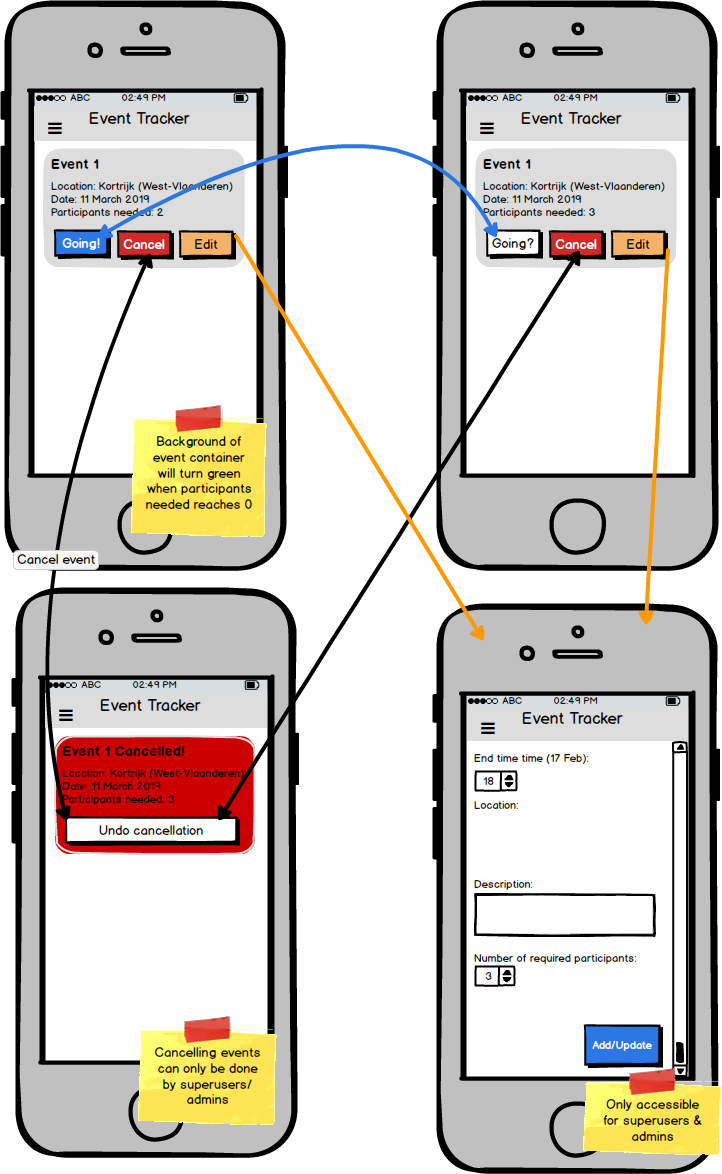


##### Sidebar Navigation

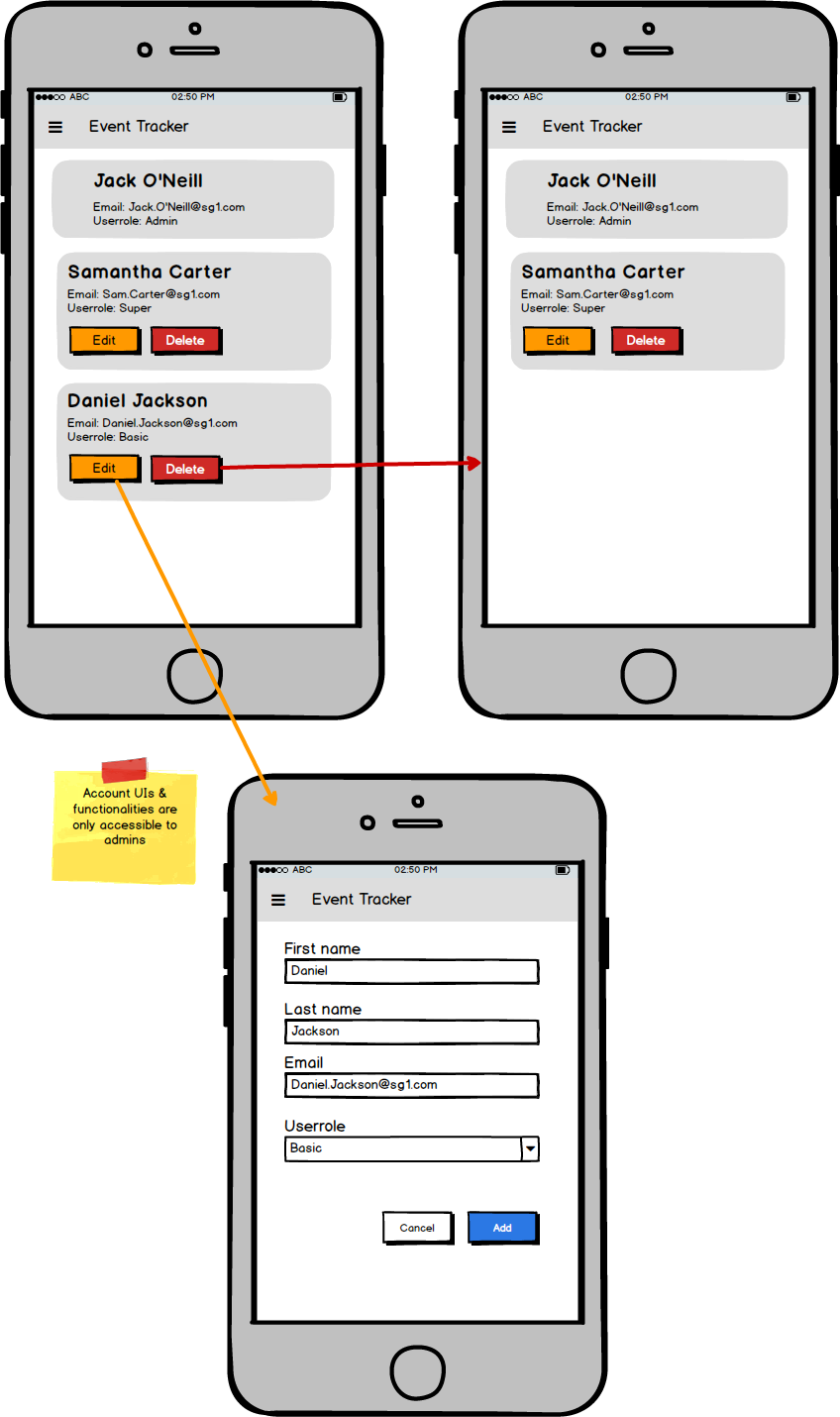


##### Event Management

Gezien onderstaand storyboard reeds veel interacties kent, is de ‘Delete event’ functionaliteit hier niet opgenomen. De gebruiker wordt na het verwijderen van een evenement simpelweg teruggestuurd naar een opgedateerd overzicht van de nog resterende evenementen.



##### User Management



# Ontwikkelingstack

## Doelplatform

* OS: Cross Platform
* Device: All devices
* Doel is een zo breed mogelijk publiek aan te spreken met een zo beperkt mogelijke code base
* Focus op mobile-friendly user experience door gebruik van Bootstrap 4 (Mobile First)

## Technologieën

##### Front end

* [HTML 5](https://www.w3.org/TR/html5/)
* [CSS 3](http://www.css3.com/)
* [SaSS](https://sass-lang.com/)
* [JavaScript](https://www.javascript.com/)
* [Jquery](https://jquery.com/) - v3.4.0
* [Jquery Validation](https://jqueryvalidation.org/) - v1.19.0
* [Jquery Validation Unobtrusive](https://www.nuget.org/packages/Microsoft.jQuery.Unobtrusive.Validation) - v3.2.11
* [Popper.js](https://popper.js.org/) - v2.0.0-next.4
* [Bootstrap](https://getbootstrap.com/) - v4.3.1
* [Font Awesome](https://fontawesome.com/) - v4.7.0

##### Back end

* [ASP.NET Core MVC](https://docs.microsoft.com/en-us/aspnet/?view=aspnetcore-2.2#pivot=core) - v2.1
* [ASP.NET Core Identity](https://www.nuget.org/packages/Microsoft.AspNetCore.Identity/) - v2.1.1
* [Entity Framework Core](https://docs.microsoft.com/en-us/ef/#pivot=efcore) - v2.1.8
* [MailKit](https://www.nuget.org/packages/MailKit/) - v2.1.5.1
* [BeginCollectionItemCore](https://www.nuget.org/packages/BeginCollectionItemCore) – v1.0.8

## Tools

* IDE: [MS Visual Studio 2019 Community Edition](https://visualstudio.microsoft.com/)
* UI Mockups & Storyboards: [Balsamiq Mockups 3](https://balsamiq.com/)
* Diagrams: [Draw.io](https://www.draw.io/)
* Visual Studio Tooling: [Bundler & Minifier](https://marketplace.visualstudio.com/items?itemName=MadsKristensen.BundlerMinifier) ([Mads Kristensen](https://marketplace.visualstudio.com/publishers/MadsKristensen)) – Version 2.9.406

# Architectuur

## Application Layers

Event Tracker bestaat uit 4 verschillende projecten. Elk project heeft zijn eigen verantwoordelijkheid.

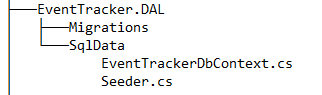
De 4 lagen zijn

* de Data Access Layer (EventTracker.DAL)
* de Models Layer (EventTracker.Models)
* de Service Layer (EventTracker.Services)
* de Web Application Layer (EventTracker.Web)

##### Data Access Layer

De Data Access Layer is verantwoordelijk voor het persisteren van de verschillende entities naar de onderliggende database. Hierbij wordt gebruik gemaakt van Entity Framework Core (v2.1.8).

Het project bestaat uit volgende files & folders:



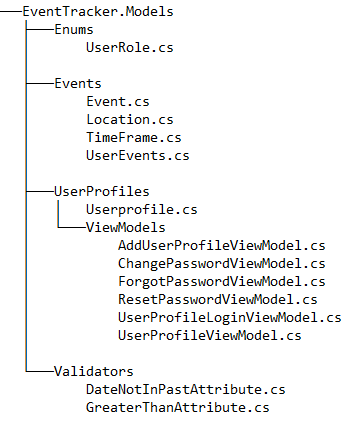
De Migrations folder wordt gebruikt door Entity Framework en zorgt ervoor dat de structuur van de onderliggende database compatibel blijft met de entities in de applicatie. Telkens wanneer de structuur van deze entities wordt aangepast in de code, moet een nieuwe migration ervoor zorgen dat de database mee evolueerd.

De SqlData folder bevat twee files:

* EventTrackerDbContext.cs: Hier wordt Entity Framework geconfigureerd door de verschillende entities en hun onderlinge relaties te verduidelijken. Deze entities en hun onderlinge relaties worden verder omschreven in het hoofdstuk ‘[Database Diagram](#_Database_Diagram)’
* Seeder.cs: Een class die verantwoordelijk is voor het populeren van de database met testdata. Deze logica wordt enkel uitgevoerd in de ‘Development’ omgeving en wanneer de applicatie detecteert dat bepaalde tables van de database leeg zijn.

##### De Models Layer

De Models Layer is verantwoordelijk voor het beheren van zowel de EntityModels als de ViewModels. Het project bestaat uit volgende files & folders:



Er worden in essentie slechts twee ‘hoofdentities’ gebruikt die op hun beurt enkele ‘subentities’ bevatten:

* Event:
  + Deze omvat alle details van een evenement.
  + Hierbij wordt gebruikt gemaakt de ‘subentities’ ‘Location’, ‘TimeFrame’ en ‘UserEvents’.
* UserProfile:
  + Deze omvat alle details van een gebruiker van de applicatie.
  + Deze klasse erft alle eigenschappen over van de ‘IdentityUser’ klasse, een klasse die gebruikt wordt door ASP.NET Core Identity, een framework die authenticatie & autorisatie beheert.

Belangrijk om hierbij te vermelden:

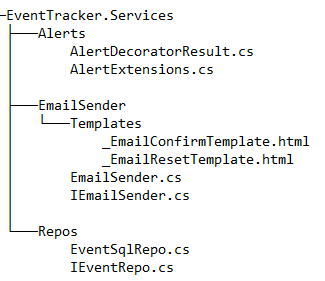
* Zowel de entity ‘Event’ als ‘UserProfile’ maken gebruik van ‘UserEvents’. Er bestaat namelijk een many-to-many relatie tussen deze entities. (Zie ook hoofdstuk ‘[Database Diagram](#_Database_Diagram)’).
* Voor de entity ‘UserProfile’ bestaan er een resem aan ViewModels. Voor de entity ‘Event’ bestaat er geen enkel ViewModel. De reden hiertoe is, kort door de bocht uitgelegd, dat de ‘Event’ entity niet gevoelig is voor overposting. Dit wordt verder verduidelijkt in het hoofdstuk ‘[Overposting](#_Overposting)’.

Naast de twee ‘hoofdentities’ en hun ‘subentities’ zijn er nog twee ‘model helpers’.

* UserRole: Een simpele enumeratie om de autorisatieniveaus (basic, super, admin) te regelen.
* Validators: Twee uitbreidingen op ‘Attributes’ die helpen bij het valideren van ‘TimeFrames’
  + DateNotInPastAttribute.cs: Zorgt ervoor dat een nieuw aangemaakt of een geopdateerd evenement geen datum in het verleden kan hebben. Het houdt immers geen steek om evenementen uit het verleden aan te maken.
  + GreaterThanAttribute.cs: Zorgt ervoor dat het startuur van een ‘TimeFrame’ niet groter is dan het einduur

##### De Services Layer

De Services Layer is verantwoordelijk voor het aanleveren van enkele services. Het project bestaat uit volgende files & folders:



* De ‘Alert‘ service is verantwoordelijk voor het beheren van alle notifications doorheen de applicatie. Het bouwt verder op de Bootstrap Alert functionaliteit. Deze service wordt verder uitgelegd in het hoofdstuk ‘[Alerts](#_Alerts)’
* De ‘Emailsender’ service is verantwoordelijk voor het versturen van zowel de ‘Confirmation Email’ bij het aanmaken van een nieuwe gebruiker als de ‘Reset Email’ bij het aanvragen van een paswoord reset.
* De ‘Repos’ service is veranwoordelijk voor alle CRUD functionaliteit wat betreft de entiteit ‘Event’. De entiteit ‘UserProfile’ heeft geen nood aan dergelijke service gezien deze gebruik maakt van het Identity framework voor database persistentie.

##### De Web Application Layer

De Web Application Layer is veruit de meest omvattende layer. Het is verantwoordelijk voor onder andere:

* het aanbieden van statische content via de wwwroot folder
* het aanleveren van de UI via Controllers & Views
* beperkte business logica
* de startup logica van de applicatie
* configuratie details
* het beheren van NuGet dependencies
* het bundlen & minifyen van javascript & css files

Het project bestaat uit volgende files & folders:



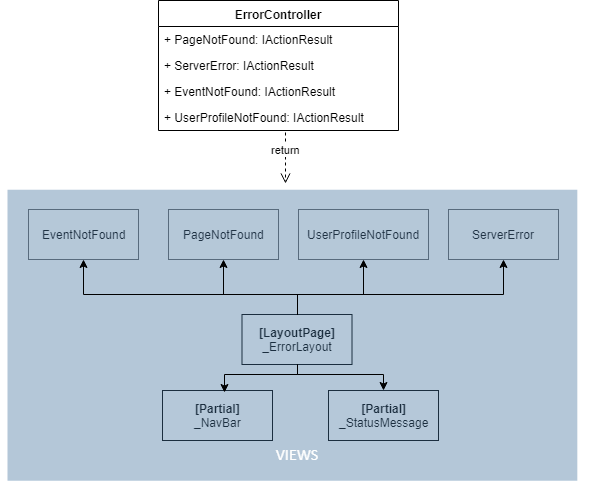
Gezien de uitgebreide collectie aan folders & files, volgt hieronder een eerder beperkte uitleg. Waar nodig wordt dieper ingegaan op bepaalde zaken of doorverwezen naar een ander hoofdstuk.

* .Json files:
  + Appsettings.json: configuratie van connectionstring & smtpclient voor productieomgeving
  + appsettings.Development.json: configuratie van connectionstring & smtpclient voor development omgeving
  + bundleconfig.json: bundles & minifies javascript files
  + compilerconfig.json: converteert scss naar css
  + package.json: basis applicatie configuratie + NuGet dependencies beheer
* Program.cs & Startup.cs:
  + Startup logica voor applicatie (toevoegen van gebruikte services + configuratie van die services)
* Controllers
  + Verantwoordelijk voor beperkte business logica + aanbieden van views
  + Controllers voor
    - Account: authenticatie & autorisatie logica
    - Error: userfriendly errorhandling
    - Events: alle logica omtrent evenementen
    - UserProfile: CRUD logica omtrent users
* Extensions
  + ServiceExtensions: Helperklasse om configuratie van bepaalde services te groeperen & te extrapoleren uit de ‘Startup’ klasse
  + UrlHelperExtensions: Helperklasse om account confirmatie & reset links te creëren
* Views
  + Alle UI logica
  + Verder verduidelijkt in het hoofdstuk ‘[View Diagram](#_View_Diagram)’
* wwwroot
  + Statische files die aan gebruiker worden aangeboden
  + css files, javascript files & fonts

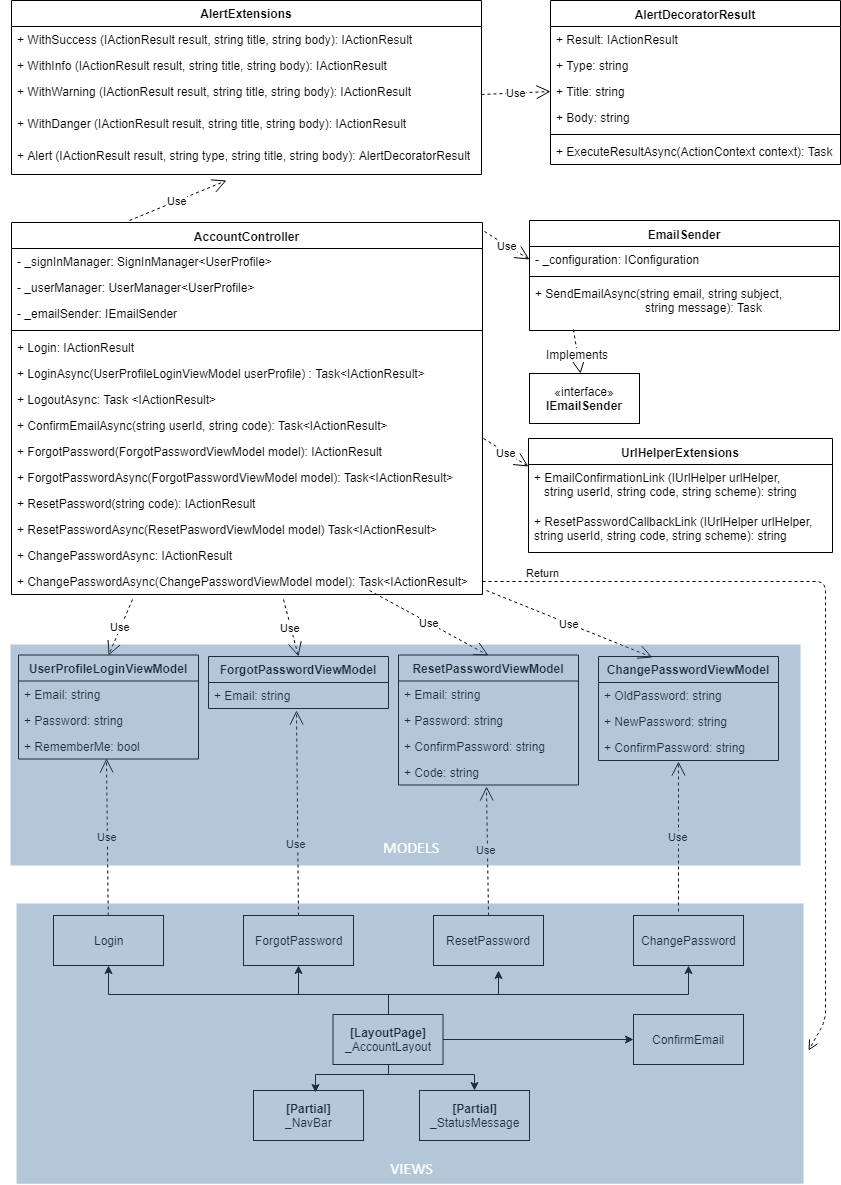
## Class Diagram

Een klassediagram opstellen van de gehele applicatie vertaalt zich al snel in een complex & chaotisch diagram. Een klassediagram per controller, die hieronder te vinden zijn, lijkt een betere keuze. Enkele elementen zijn echter alsnog verwijderd om de complexiteit verder te beperken. Deze elementen zijn onderdelen van frameworks zoals de UserManager klasse of SignInManager klasse van het Identity Framework of de Program & Startup klasse van ASP.NET Core. Een overzicht van de fields en methods van deze klasses kunnen in de Microsoft documentatie worden teruggevonden.

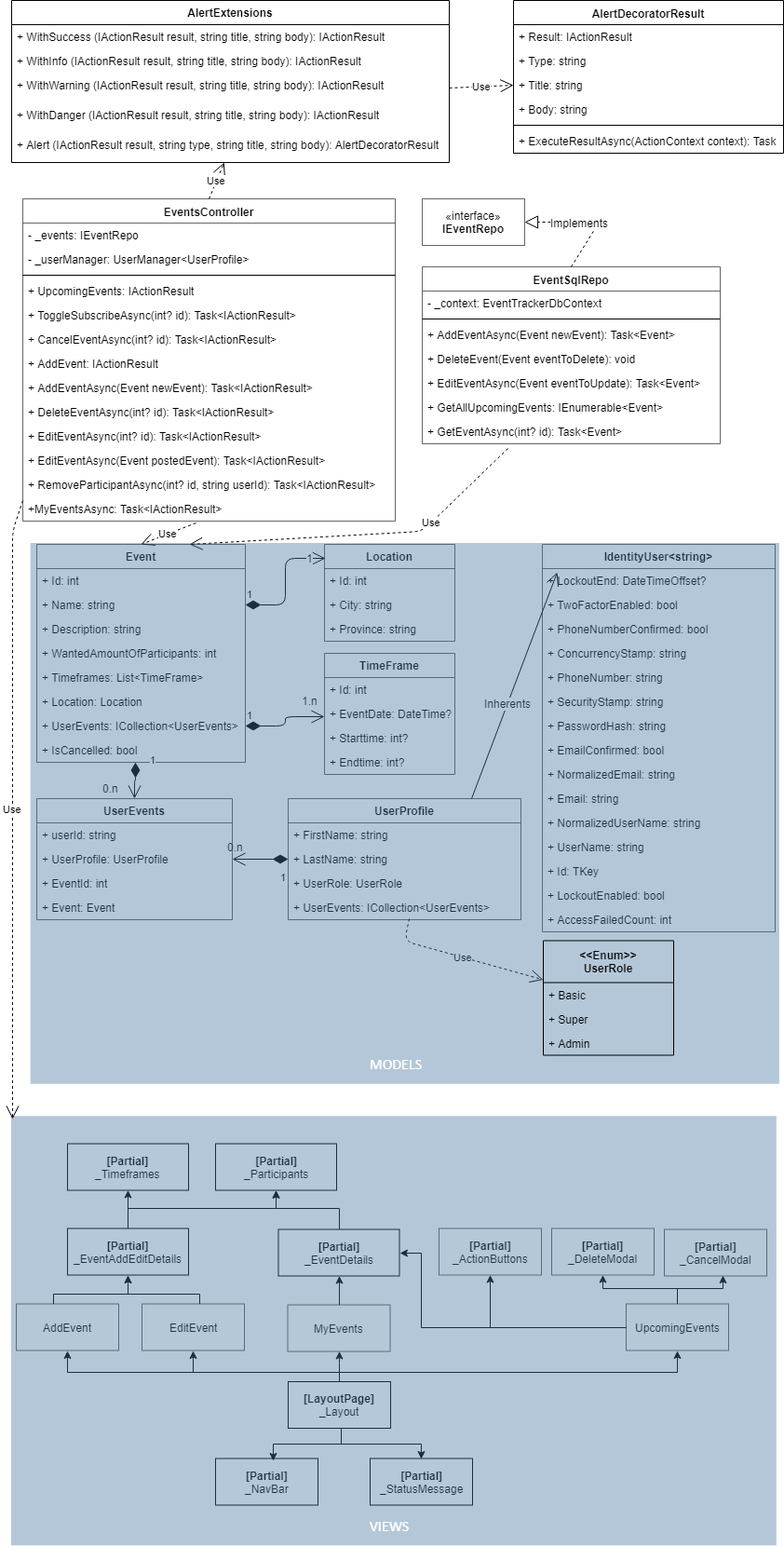
##### ErrorController



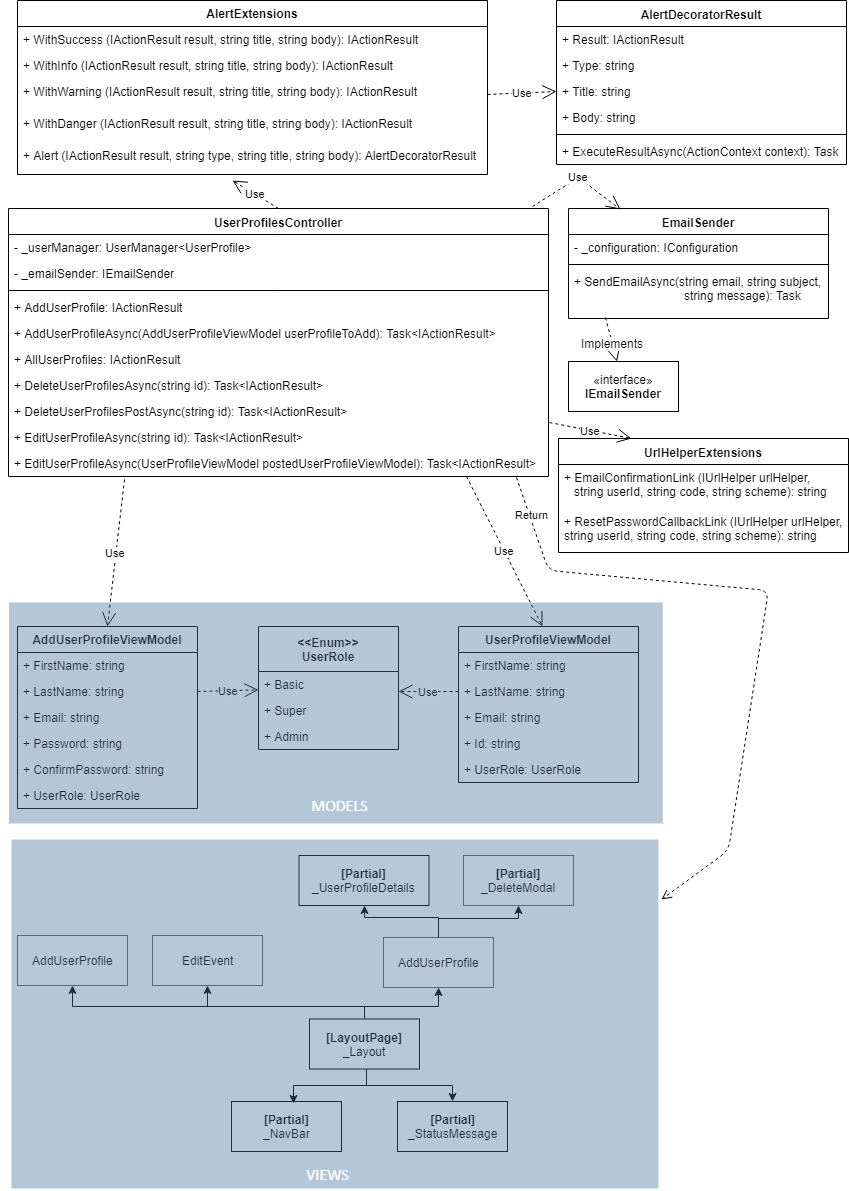
##### AccountController



##### EventsController

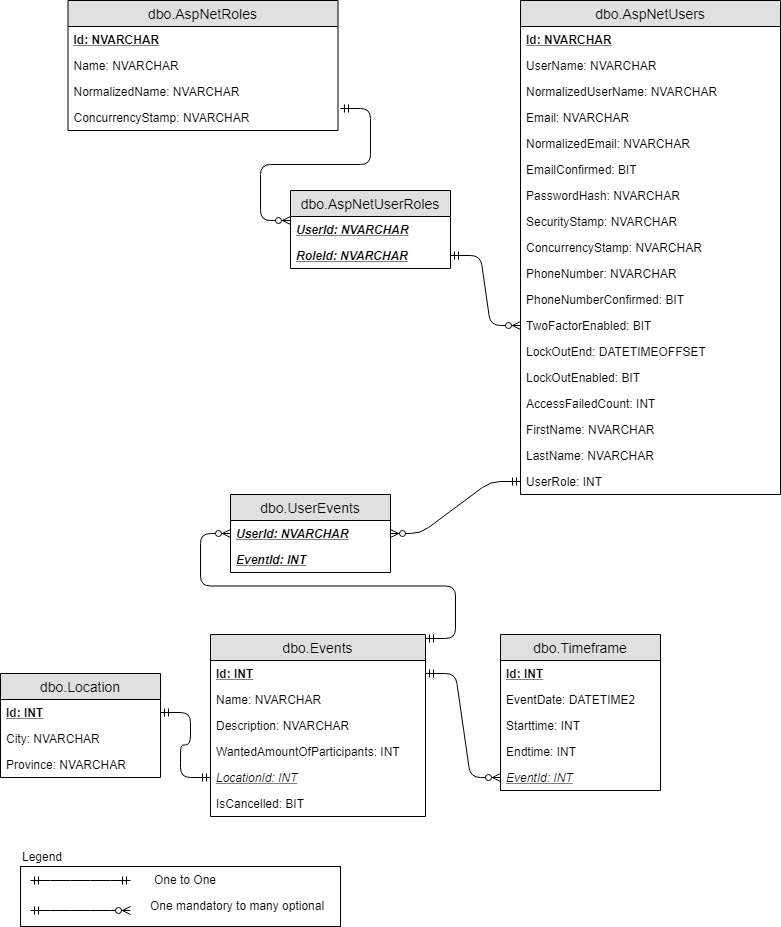


##### UserProfilesController



## Database Diagram

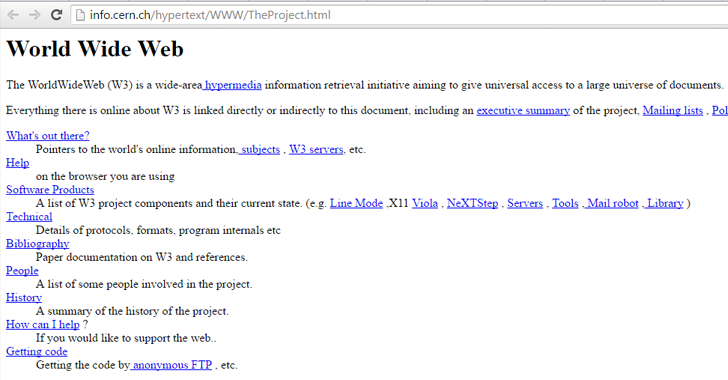
Hieronder een overzicht van de verschillende tables in de onderliggende database. De door Entity Framework gegenereerde tables die niet worden gebruikt & bijgevolg geen data bevatten, alsook de table die voor het beheer van de Migration history wordt bijgehouden, zijn niet opgenomen in dit diagram.



# Technologische Uiteenzettingen

## Mobile First

Op de dag van dit schrijven, vierde het wereldwijde web zijn 30e verjaardag. Bij zijn ontstaan werden we verwelkomd in een wereld met ogenschijnlijk geen limiet op zijn eigen potentieel. Maar de eerste webpagina’s zagen er niet bepaald sexy uit.



De vraag naar gebruiksvriendelijke en mooie websites kwam er toen steeds meer mensen hun weg vonden naar het wereldwijde web. Dit creëerde nieuwe uitdagingen. Naast de inhoud werd nu ook de lay-out belangrijk. En er veranderde nog meer. De manier waarop we vandaag websites bezoeken, is niet dezelfde als 30 jaar geleden. Waar websites vroeger voornamelijk via desktops of laptops werden bezocht, zien we vandaag hoe smartphones en tablets deze rol overnemen.

Als antwoord op deze trends, is men via heel wat tussenhaltes gekomen tot ‘responsive webdesign’. Websites oogden nu fris en modern en waren vlot toegankelijk vanop ieder device. Maar er was ruimte voor verbetering. Men ging er namelijk van uit dat websites vooral via een device met groot scherm werden bezocht. De lay-out van een webpagina werd bijgevolg berekend op de dimensies van een groot scherm. Pas als de website zou detecteren dat ze werd geraadpleegd via een klein scherm, zou de lay-out worden herberekend op basis van de dimensies van dit scherm.

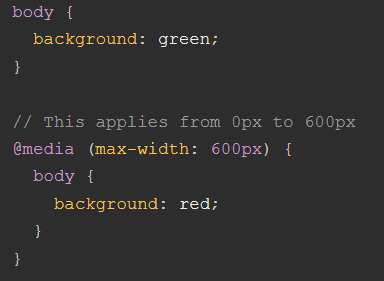
Een jammere zaak omwille van twee redenen:

1. Door de opkomst van smartphones, worden steeds meer websites bezocht door middel van devices met klein scherm.
2. Smartphones of tablets hebben doorgaans minder rekenkracht als desktops of laptops.

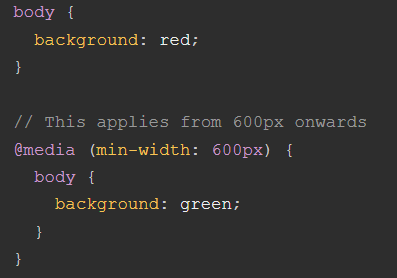
Een lay-out voor een groot scherm wordt dus vaak onnodig berekend. Als antwoord hierop, ontstond de ‘Mobile First’ filosofie waarbij devices met kleiner scherm net bevoordeeld worden. De lay-out wordt in eerste instantie berekend op kleinere schermen. Pas als de website detecteert dat een device met groot scherm de website raadpleegt, wordt de lay-out herberekend naar een groter formaat. Hierbij kan gebruik gemaakt worden van de doorgaans grotere rekenkracht van een device met groot scherm. Men draait de boel dus om.



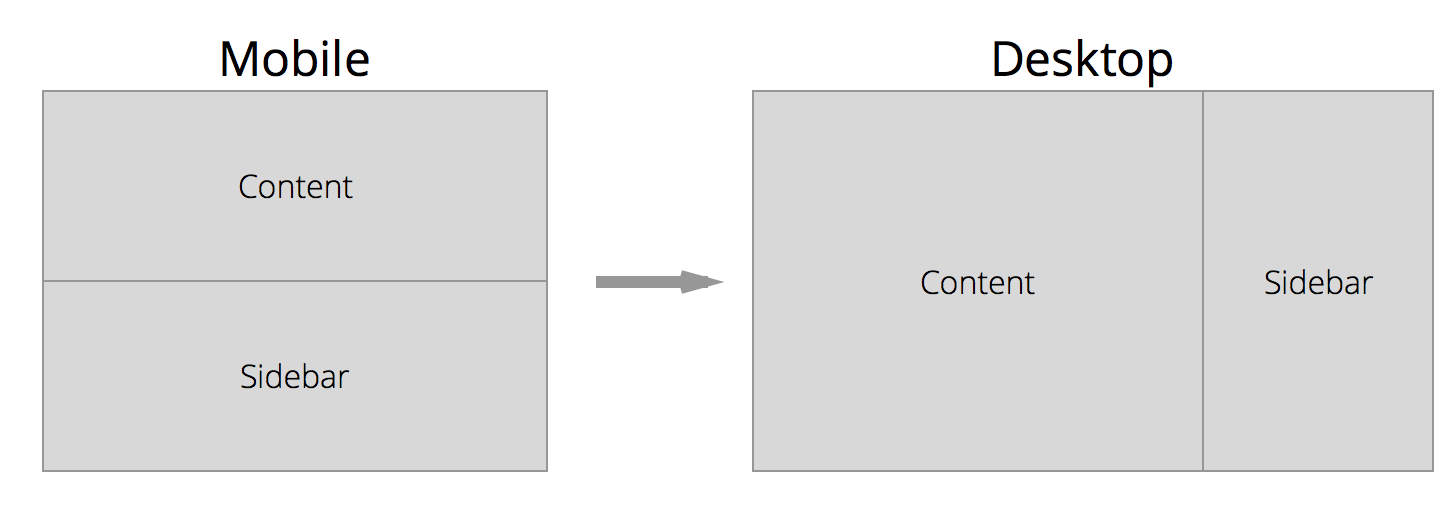
Als we vervolgens in de technische details duiken, moeten we het eerst hebben over media queries. Aan de hand van een media query wordt bepaald welke CSS gebruikt wordt afhankelijk van de dimensies van het scherm. Zo zal de achtergrond van een website door onderstaande code rood kleuren indien deze bezocht wordt door een scherm dat smaller is dan 600px.



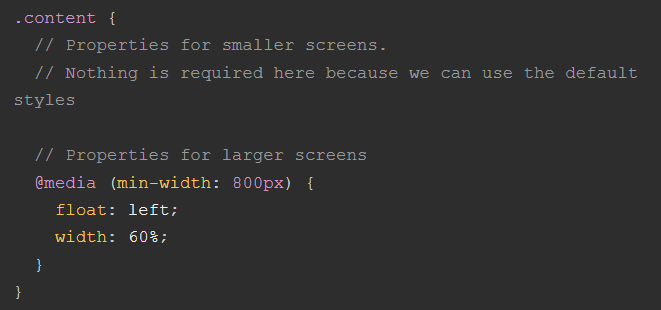
Dit is een voorbeeld van ‘Desktop First’. Bij ‘Mobile First’ keert men dit proces om. De CSS zal eerst de lay-out voor een kleiner scherm berekenen. Pas bij een breedte van 600px zal een andere lay-out berekend worden.



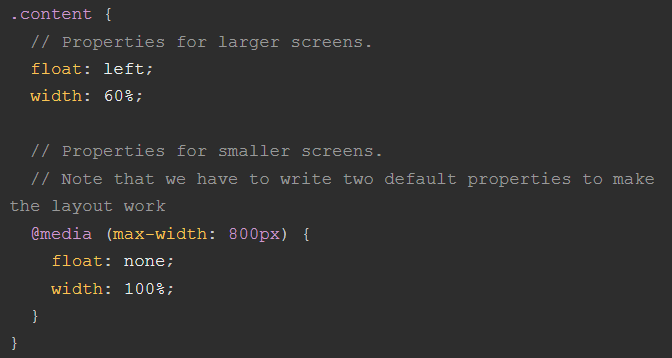
‘Mobile First’ biedt ook een voordeel voor de ontwikkelaar. Het is namelijk zo dat men bij het ontwikkelen van een mobiele user interface sneller gebruik kan maken van de default waardes van bepaalde elementen. Stel dat we een website maken waar er naast de hoofdinhoud ook een sidebar is. Het is op een kleiner scherm interessanter om deze sidebar onder de hoofdcontent weer te geven:



In het geval van een kleiner scherm gebruikt de sidebar 100% van de schermbreedte. In het geval van een groter scherm, komt de sidebar naast de hoofdinhoud te staan en zal deze pakweg 40% van het scherm innemen. Bij een ‘Mobile First’ aanpak kunnen we profiteren van de default waarde van de width property van een <div>:



Bij ‘Desktop First’ ontstaat er al sneller een situatie waar we een default waarde moeten ‘resetten’:



De ‘Mobile First’ approach bespaart ons hier 2 lijnen code. Als we deze performantie winst kunnen doortrekken naar een complexe website, kunnen we ook hier concluderen dat ‘Mobile First’ een mooi voordeel biedt.

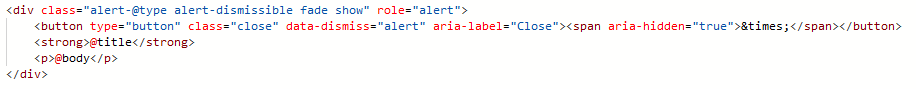
## Extensions

##### Alerts

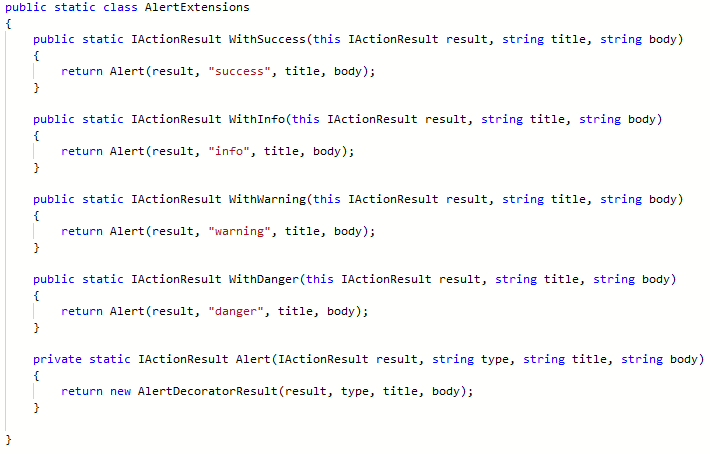
Doorheen de applicatie wordt gebruik gemaakt van notifications in de vorm van ‘Alerts’. Dit is een door Bootstrap aangeboden component om op een gemakkelijke manier aan de eindgebruiker een beknopt stuk informatie aan te bieden. In de EventTracker applicatie worden deze alerts gebruikt om het succes of falen van een gebruikersactie aan te tonen. Een concreet voorbeeld is het inloggen. Als de gebruiker een correct email adres en paswoord aanlevert, zal een succes alert verschijnen. Indien het paswoord verkeerd zou zijn, wordt de gebruiker hier ook op de hoogte van worden gebracht via een ander type alert.

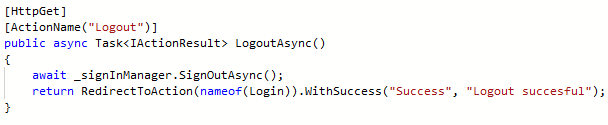
EventTracker maakt gebruik van een techniek die gebruik maakt van extensions methods om alerts makkelijk te integreren binnen ASP.NET Core. Deze techniek wordt dieper in detail uitgeschreven op de website trycatchfail. (<https://www.trycatchfail.com/2018/01/22/easily-add-bootstrap-alerts-to-your-viewresults-with-asp-net-core/>)

Eerst en vooral voorziet men de nodige HTML markup voor het weergeven van Bootstrap alerts. Gezien we deze alerts doorheen de gehele applicatie willen gebruiken, plaatsen we deze markup in de verschillende layout views:

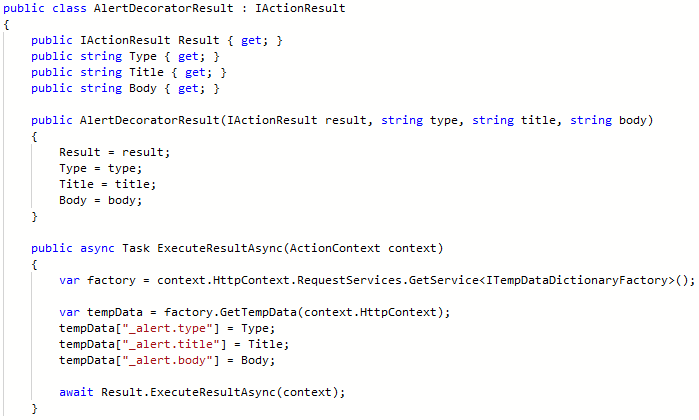


Door gebruik te maken van onderstaande extension methods kunnen we alerts makkelijk en overzichtelijk binnen de controller integreren. Het enige dat we immers hoeven te doen, is een titel & boodschap meegeven samen met het returnobject van een method.

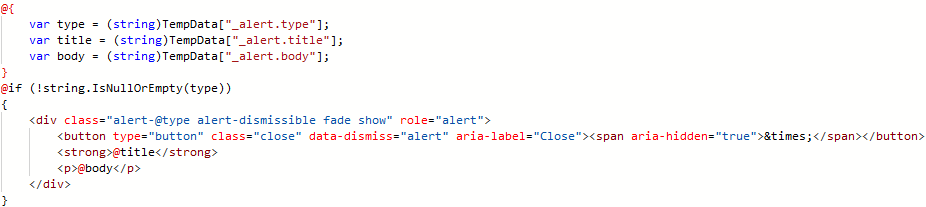




Via een helperklasse kunnen we duidelijk te maken welk soort alert we wensen:



Het resultaat van de ‘ExecuteResultAsync’ method, vangen we tenslotte op in onze HTML markup via deze kleine toevoeging:



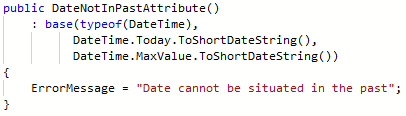
Kort samengevat zijn dit de verschillende stappen die de applicatie doorloopt om alerts te laten verschijnen:

1. Een gebruiker voltooit een actie en activeert daarbij een controller action. Die action eindigt met een returnobject waarop een extension method is toegepast. Deze extension method heeft naast het return object ook een titel en boodschap.
2. Het resultaat van de extension methods is een ‘AlertDecoratorResult’ object dat verantwoordelijk is om het type, de titel en de boodschap van de alert in een tempData object te steken.
3. Dit tempData object wordt door de View gebruikt om de HTML markup aan te vullen.
4. Het Bootstrap framework doet de rest & de alert verschijnt.

##### Validators

Binnen de EventTracker applicatie worden 2 zelfgemaakte validators gebruikt om een ‘Timeframe’ object te valideren. Een ‘Timeframe’ object bestaat uit een ‘Id’, ‘Event Date’, ‘Start’ en ‘End’ property.

* Het DateNotInPastAttribute zorgt ervoor dat wanneer een superuser/admin een evenement aanmaakt of wijzigt, de datum van dat evenement nooit in het verleden kan liggen.



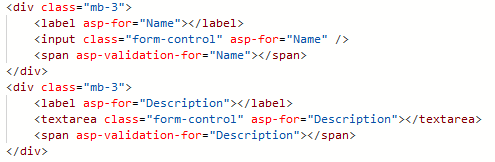
* Het GreaterThanAttribute zorgt ervoor dat de property waarop het wordt toegepast groter moet zijn dan een meegegeven property. In het geval van de EventTracker applicatie wordt deze Attribute toegepast op de ‘End’ property om zo te verzekeren dat het einduur op een bepaalde datum niet voor het startuur kan liggen.



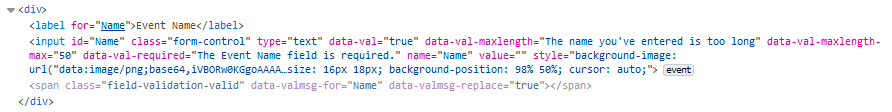
## Vraagstukken Tijdens Development

##### Timeframes toevoegen

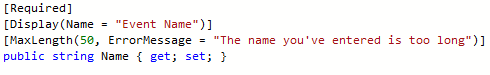
Een bijzonder complexe regio van de EventTracker applicatie is het toevoegen van nieuwe evenementen. Een evenement heeft enkele properties zoals een naam of een omschrijving. Bij het implementeren van een ‘Add’ of ‘Edit’ functionaliteit, zorgen de MVC TagHelpers, met dank aan .Net Core, hier voor een doodeenvoudige implementering:



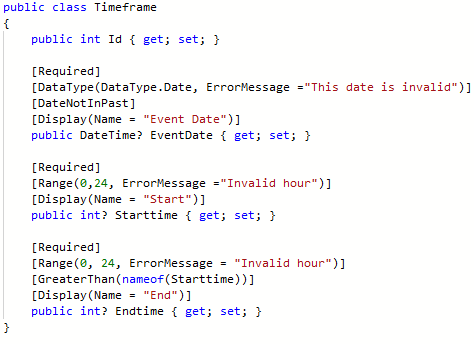
Deze TagHelpers genereren tijdens het compileren de nodige HTML voor een succesvolle modelbinding:



Hier wordt niet alleen het “Name” attribuut gegenereerd, hetgeen nodig is voor model binding, ook de “Data Annotations” waarmee de properties van het model zijn gedecoreerd, zijn omgezet naar de nodige attributen voor validatie.



Maar niet alle properties vallen zo makkelijk te implementeren. Zo zijn er de ‘Timeframes’ van een evenement. Deze property is een “List” van het complexe type “Timeframe”. Één enkele timeframe bestaat uit een datum, start- en einduur. Elke property van een “Timeframe” is ook voorzien van enkele “Data Annotations”, waarvan zelfs 1 custom-made, namelijk [DateNotInPast]:



Niet alleen de complexere aard van deze property zorgt voor enkele hersenkronkels. De achterliggende domeinproblematiek bemoeilijkt de zaak. Zo moet een gebruiker bij het aanmaken van een nieuw evenement minstens één ‘Timeframe’ toevoegen. Hij mag er meerdere toevoegen. Hoeveel ‘Timeframes’ een gebruiker zal toevoegen is ongekend. Ook het wijzigen van een bestaand evenement botst hier op extra domeinproblematiek. We willem immers vermijden dat een gebruiker alle timeframes van een bestaand timeframe verwijderd. Een evenement heeft minstens 1 timeframe nodig.

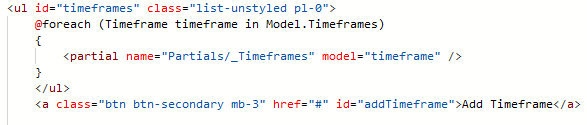
Hoe zorgen we ervoor dat deze logica gerespecteerd blijft? Hoe kunnen we taghelpers op een dynamische wijze gebruiken zodat elke toegevoegde ‘Timeframe’ correct zal modelbinden?

De oplossing bestaat uit meerdere stappen. Eerst maken we een Partial View met taghelpers voor de properties van een ‘Timeframe’:

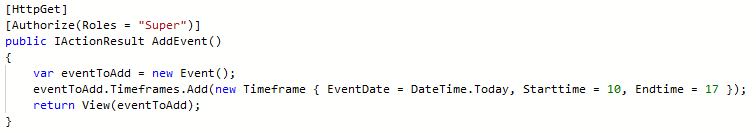


Merk op dat we hier gebruik maken van een ‘Html.BeginCollectionItem’ helper. Dit is een HtmlHelper die ervoor zorgt dat ook collection properties correct binden. Meer details over deze helper vind je in het hoofdstuk ‘’[Technologieën – Back End](#_Back_end)”.

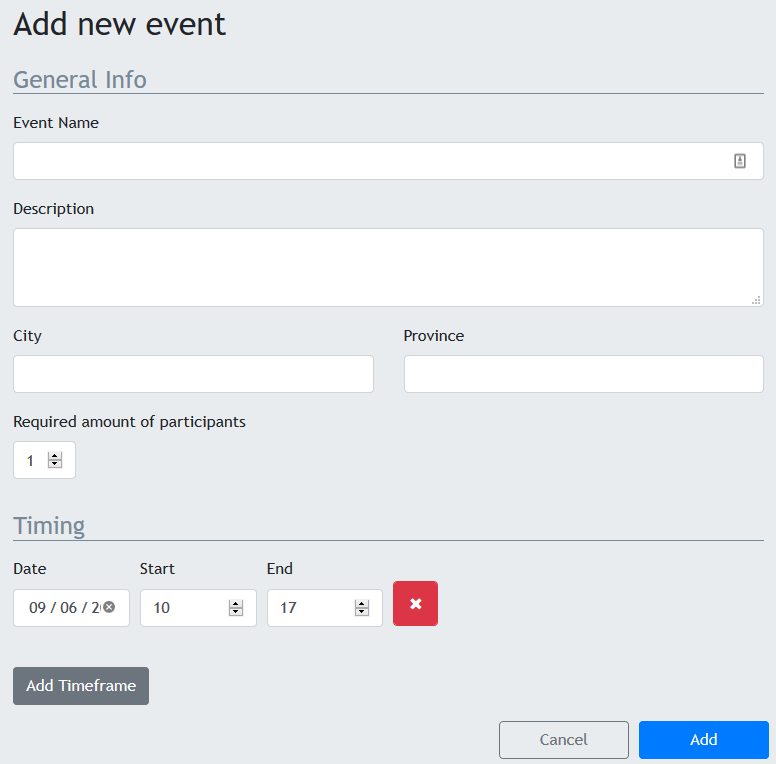
Deze Partial View integreren we in de ‘Add’ of ‘Edit’ View.



De applicatie loopt hier door de ‘Timeframe’ collectie van een nieuw evenement. Je zou denken dat daar geen enkel element in zit maar als er geen enkele ‘Timeframe’ in de collectie zou zitten, zou er ook geen enkel inputveld verschijnen. Pas na het klikken op de knop ‘Add Timeframe’ zou dergelijke UI verschijnen. Jammerlijk klikwerk omdat we willen dat een gebruiker sowieso 1 ‘Timeframe’ toevoegd. Een perfect moment om dit af te dwingen en onmiddellijk enkele default waardes mee te geven. We doen dit door in de controller een nieuw ‘Event‘ model aan te maken en onmiddellijk een ‘Timeframe’ toe te voegen aan de collectie:

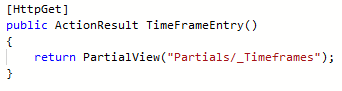


Dit alles resulteert in onderstaande user interface:

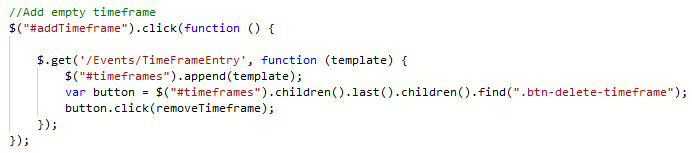


Als we op de ‘Add Timeframe’ knop duwen, zal er echter niet veel gebeuren. We moeten immers nog configureren dat de applicatie steeds een Partial View van een ‘Timeframe’ genereerd telkens we erop klikken. En daarbovenop moeten we verhinderen dat een gebruiker alle ‘Timeframes’ verwijderd en dus een evenement aanmaakt zonder ‘Timeframes’.

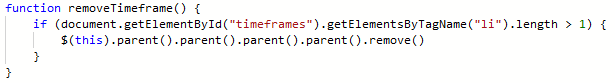
Om dit probleem om te lossen, voorzien we eerst en vooral een action in de controller die verantwoordelijk is voor het leveren van zo’n Partial View:



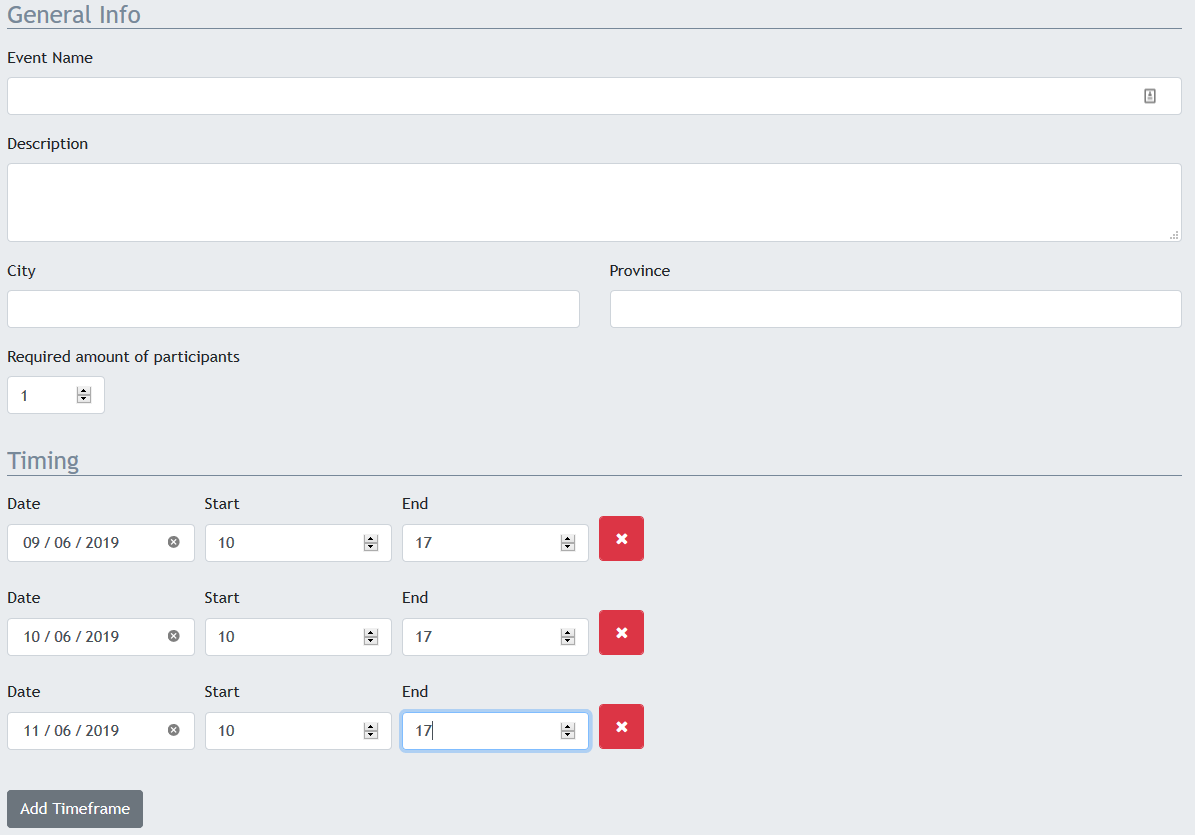
Met behulp van ajax vragen we naar het resultaat van deze action & plakken we hem bij de reeds bestaande HTML:



De laatste 2 lijnen in bovenstaande Jquery, zorgen ervoor dat bij elke ‘Delete’ button die gegenereerd wordt, de nodige eventhandler wordt toegevoegd. Deze zorgt ervoor dat er telkens minstens 1 ‘Timeframe’ op de user interface blijft staan.



Uiteindelijk eindigen we met onderstaande user interface:

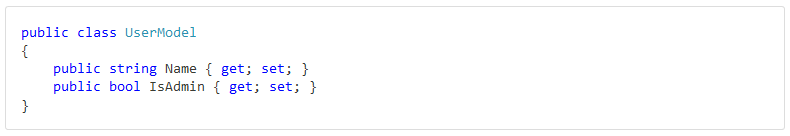


##### Overposting

In het hoofdstuk ‘[De Models Layer](#_De_Models_Layer)’ werd reeds vermeld dat de entity ‘Event’ niet over een ViewModel beschikt terwijl de entity ‘UserProfile’ wel degelijk over meerdere ViewModel alternatieven beschikt. Doorgaans wordt er aangeraden om wel met ViewModels te werken. De reden hiervoor: overposting! Om te kunnen verduidelijken waarom er bij de entiteit ‘Event’ niet voor is gekozen om met ViewModels moet eerst worden uitgelegd wat overposten is en waarom het belangrijk is om een applicatie hiertegen te beschermen.

Overposting is een kwetsbaarheid binnen een webapplicatie waarbij er een onvoldoende afgeschermd datamodel bij een POST operatie teruggestuurd wordt naar de server waardoor een gebruiker in staat is om meer data dan gewenst/toegestaan te manipuleren. Een concreet voorbeeld brengt meer duidelijkheid:

Stel, we hebben onderstaand UserModel:

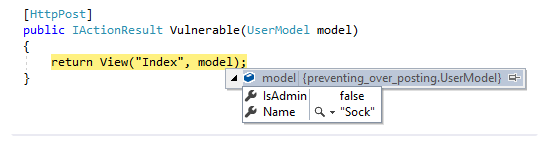


We wensen dat een gebruiker enkel de ‘Name’ property kan wijzigen. Het zou immers niet wenselijk zijn dat de gebruiker zomaar kan beslissen wie er een admin is en wie niet. We implementeren onderstaande controller met bijbehorend model.

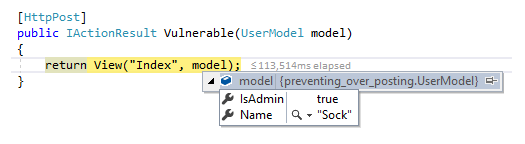




Hier gebruiken we hetzelfde UserModel en voorzien we de nodige html waardoor een gebruiker op een eenvoudige en gebruiksvriendelijke manier de ‘Name’ property kan wijzigen. Als we de applicatie zouden debuggen, krijgen we volgende waardes te zijn bij het wijzigen van de ‘Name’ property via de aangeboden user interface:



Een kwaadaardaardige gebruiker kan echter met behulp van tools zoals Postman of Fiddler de ‘IsAdmin’ property toch manipuleren:



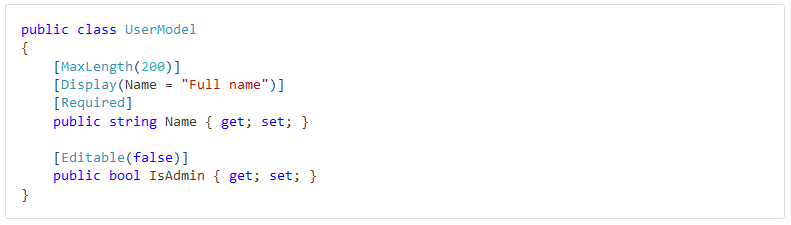
Dit komt omdat de ‘IsAdmin’ property nu eenmaal deel uitmaakt van het model. Tevens is de property ‘public’ dus vrij toegankelijk voor iedereen. Op geen enkele manier wordt in de code ervoor gezorgd dat de property niet mag gewijzigd worden door een gebruiker.

Er zijn meedere manieren om een applicatie te wapenen tegen overposting. Hieronder enkele mogelijkheden:

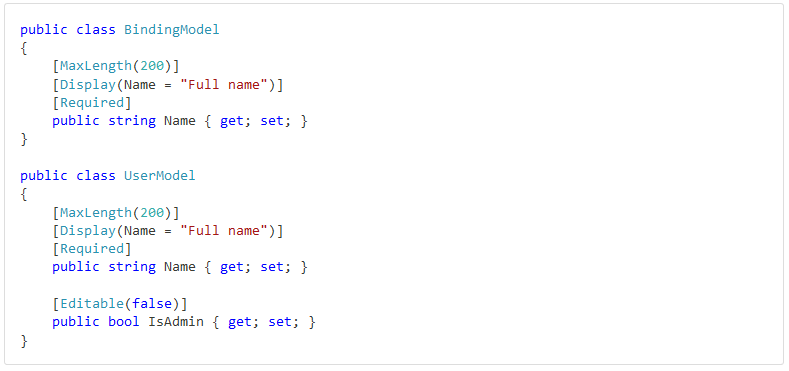
* Gebruik maken van het [BindAttribute] zorgt ervoor dat men in de controller kan specifiëren welke properties er gewijzigd kunnen worden:



* Een soortgelijk resultaat kan bereikt worden met de properties van het model te bekleden met het [Editable] of [BindNever] attribuut.



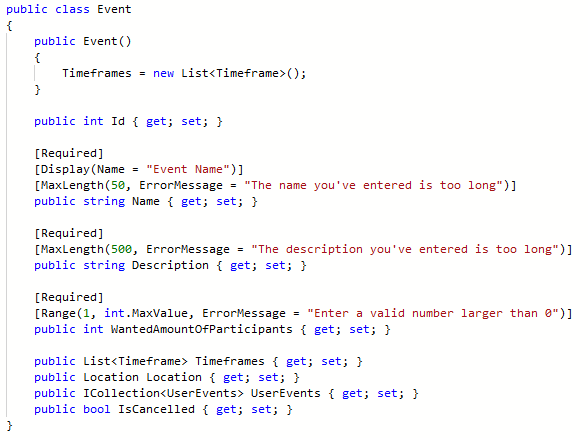
* Veruit de meest gebruikte methode is het gebruik van ViewModels. Dit kan beschouwd worden als gesimplifieerde versies van het oorspronkelijke model (vanaf nu entity model genoemd). Deze ViewModels worden gebruikt om data van de client naar de server te sturen en bevatten enkel de properties die mogen geworden gemanipuleerd. Zelfs als een kwaadaardige gebruiker de data in het post request zou manipuleren, wordt er met deze data geen rekening gehouden op de server.





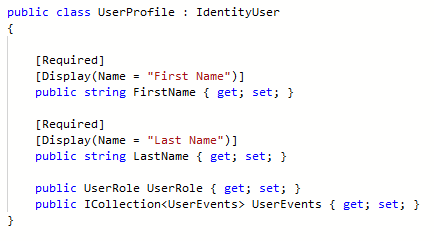
Er zijn nog mogelijkheden om overposting te voorkomen. Deze liggen echter buiten de scope van dit project. Indien gewenst kan men hier meer over te weten komen op deze [blogpost](https://andrewlock.net/preventing-mass-assignment-or-over-posting-in-asp-net-core/) (<https://andrewlock.net/preventing-mass-assignment-or-over-posting-in-asp-net-core/>). Het is tevens deze post waaruit bovenstaande voorbeelden zijn onttrokken.

Binnen de EventTracker applicatie worden geen van bovenstaande technieken noch een andere techniek gehanteerd om overposting op het ‘Event’ model te voorkomen. Dit voor de simpele reden dat er geen gevaar tot overposten is. Laat ons het ‘Event’ model even in detail bekijken:



Iedere property van dit model mag gemanipuleerd worden. DIt is zelfs in grote mate de boedeling van de applicatie voor zowel de superusers als de admins. Niet alleen moeten zij nieuwe evenementen aanmaken, zij zijn ook verantwoordelijk voor deze up to date te houden. Stel dat de datum van een evenement verandert, dan is het de bedoeling dat een superuser/admin dit aanpast via de applicatie. Door dit te doen zou hij in dat geval de timeframes collectie benaderen en aanpassen. Ook de ‘Name’, ‘Description’, ‘WantedAmountOfParticipants’, ‘Location’, ‘UserEvents’ en ‘IsCancelled’ property moeten aangepast kunnen worden. De enige property die niet gewijzigd mag worden, is de ‘Id’ property en gezien deze property door Entity Framework gebruikt wordt als primary key in de onderliggende databank, kan deze niet gewijzigd worden.

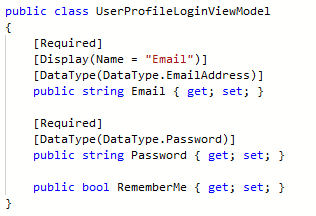
Dit alles gaat er wel van uit dat evenementen enkel en alleen aangemaakt en gewijzigd kunnen worden door gebruikers met de daarvoor voorziene autorisatie. Zoals hierboven reeds vermeld, zijn dit dus de superusers en de admins van de applicatie. Deze autorisatie wordt toegewezen aan een ‘UserProfile’ model via de ‘UserRole’ property:



Gezien een ‘UserProfile’ overerft van ‘IdentityUser’, wordt dit model gebruikt bij alle functionaliteiten omtrent authenticatie, zoals bijvoorbeeld bij het inloggen:



In bovenstaande View is echter gebruik gemaakt van een ‘UserProfileLoginViewModel’ en niet van het oorspronkelijke ‘UserProfile’ model. Dit om te vermijden dat een user bij het posten van data tijdens het inloggen ook zijn UserRole zou kunnen aanpassen (bijvoorbeeld naar een hoger niveau). Het ‘UserProfileLoginViewModel’ beschikt namelijk niet over deze property:



Door gebruik te maken van ViewModels wordt het ‘UserProfile’ model beschermd tegen overposting. Dit in tegenstelling tot het ‘Event’ model waar dit dus niet nodig is.