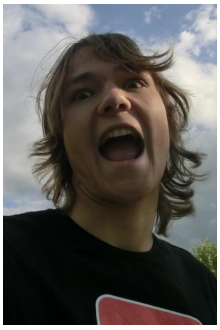
Contextproject Programming Life
Group 2 - GEVATT
Final Report

TU Delft



Ruben Bes
rbes
4227492

Mathijs Hoogland
mhhoogland
4237676

Jasper Denkers
jdenkers
4212584

Robbert van Staveren
rhvanstaveren
1527118

Willem Jan Glerum
wglerum
4141040

June 19, 2014

Abstract

This is the final report for the Programming Life Contextproject, a second year course from Computer Science at TU Delft. The Contextproject course is about applying all learned skills in a particular context at developing a piece of software. In this case the context was bioinformatics and we worked 10 weeks with a team of five people.

This document contains the main information about development, implementation and validation of the product. Main features of the product will be presented and it will be discussed how they satisfy the user needs. Furthermore, this document will contain an HCI module about the interaction of users with the product. Finally, an outlook will be given to show what possible improvements could be implemented if this project will continue in the future.

Besides this document several other documents are made covering other parts of the project. This is the final document consisting of the most information about the project.

Contents

1 Introduction

The application we developed is called GEVATT, which stands for GEnetic Variations Analyzer Through Triodata, which is an application to be used by doctors to browse genetic information of patients. Such applications are also called genome browsers. It is a secured web application based on the Play Framework. Doctors need to login from any device with a browser in order to use the application.

The application makes it possible for a doctor to upload a VCF file and let the application analyse it. A VCF (Variant Call Format) file is a file used in bioinformatics for storing gene sequence variations. This storing of genetic information of patients is based on variations between the patient and a reference genome. This is done because storing all information would be taking too much space.

After uploading the file, the user of the application (a doctor) waits until the file is processed. In the meanwhile the doctor could browse other patients he uploaded information about earlier. After uploading, the first important part of the application is executed: analysing the data. The outcome of the analysation consists of mutations found in the genome of the patient and the relations between these mutations.

Secondly, the main focus of the application is about visualising the found mutations. This is firstly done by giving a main overview of the whole patient. An overview of all chromosomes is given, and per chromosome is indicated if and how many mutations it contains. Besides a visual overview of the chromosomes there's a tabular overview with all mutations and some extra information per mutation. This makes it easier to estimate which mutations are more harmful than others.

Most information is shown on the overview pages per mutation. The pages with these visualisations have two visualisations which both give another insight in the mutation. The first part shows the position of the mutation relative to a gene. The second visualisation is a graph-based interactive visualisation showing proteins related to the mutation and the connections between these proteins. Here we also have a tabular overview per protein with about which diseases they could cause and to which other mutations of the patient they are related.

2 Overview of the product

The developed product, a secured web application, is built on de Play Framework. It contains several parts that work together to deliver a user friendly environment to explore genetic information.

2.1 Authentication

Starting with the basis, it's a secured web application where a user needs to log in. After logging in some secured pages become accessible. Some other pages are already accessible before logging in, e.g. the 'documentation' and 'about' pages. We distinguish the following parts:

- A login page

- Secure pages that aren't publicly accessible
- Redirect if secured page is requested while unauthenticated
- Prevent doctors from accessing data of patients of other doctors

2.2 Context specific

The web framework and it's built-in secure module isn't developed by ourselves so we could focus on developing the context specific parts. This was mainly separable in three parts: analysing data, retrieval of relevant data from databases and visualising data.

2.2.1 Data analysation

Analysing the data is done by processing the VCF file and detecting mutations. The outcome of the processing is used in the visualisation. At both of these parts there's information used given by some databases. The information is used to get more information about mutations and find relations between several mutations. The application does the following:

- Read the VCF file and find mutations of two types: de novo's and recessive homozygous
- Save metadata about the uploaded file
- Find relations between found mutations

2.2.2 Data retrieval

Information from multiple databases is used for both visualisation and finding relations between mutations. This includes:

- Manage connections to multiple databases
- Querying databases to get relevant information from mutations and proteins

2.2.3 Visualisation

The most important part of the application: visualisation. Multiple views of the same found mutations are made to give as much insight as possible. This includes the following visualisations:

- An overview of all mutations in a patients VCF file with a view per chromosome
- A distinct page per mutation, showing:
  - The position of the mutation relative to nearby genes
  - A graph with a protein related to the mutation, and proteins related to that protein

3 Description of the functionality

This chapter covers a detailed description of the developed functionality in the product. This description is based on the overview of all functionality given in the previous chapter and therefore has the same structure.

3.1 Webapplication

The application is a web-application based on the Play Framework. This means the application is accessible via devices with an internet connection and a browser. The layout is optimised for devices with a screen with a minimum width of 1024 pixels. The page structure is as follows:

- A login page (see authentication below)
- A dashboard page with an introduction to the application and links to the following context specific parts:
  - The patient overview with a sortable list of all patients with some meta data. By clicking on a patient the user goes the patient specific page
  - On the patient page an overview of the found mutations is given. It contains a visual overview of the chromosomes and the mutations found per chromosome
  - A separate page with visualisations and information (see visualisation below) per mutation
  - There's a separate page for adding new patients to the database

3.2 Authentication

The authentication part takes care of securing the pages and data of the application to prevent it from being accessible to everyone. It does the following:

- If a user isn't authenticated and tries to open a secured page, he is redirected to the login page
- At the login page, a user needs to supply his username and password to sign in. When these credentials aren't recognised, the user receives an error message
- When logged in, each page contains some user specific information, like his name. Furthermore, links to application specific pages become visible and a logout button appears

3.3 Context specific

3.3.1 Data analysation

The analysing of data is done after a VCF file is uploaded.

- When a user has added a patient by filling in the associated information and uploading a VCF file, the user gets redirected to the patient overview. In the background the file gets processed, so the user can continue browsing. After processing, the patient overview gets updated automatically and the patient view becomes available.
- The mutations that are found are stored in a databases dedicated to the application
- The application searches for relations between the found mutations and stores them in the database

3.3.2 Data retrieval

We use the following databases to retrieve relevant information:

- CADD for retrieving scores representing the dangerousness of mutations
- dbSNP for retrieving all kind of information related to SNPs
- STRING for retrieving everything related to proteins

3.3.3 Visualisation

Several visualisations are used to bring show retrieved information in a clear way to the user.

- The first visualisation is about giving an overview of all found mutations, showing them per chromosome. This overview is given on the patient overview page. It contains a graphical representation of all chromosome pairs and each pair is colored red or black if it respectively contains a mutation or not. While hovering over a chromosome pair, a list appears with links to the mutations found in that pair.
- On the pages for individual mutations there are two extra visualisations
-
  - The first mutation-specific visualisation is the top one found on the mutation page. It shows the part of the chromosome that the mutation is on, and displays the position of the base pair and the genes relatively close to this pair.
  - There's also an overview in the form of a graph containing proteins related to the mutation. The protein directly related to the mutation is marked and proteins related to the marked one are shown in the graph. This graph is interactive so proteins can be dragged around to get a clearer view. Connections between protein are shown darker and thicker if the connectivity between two proteins is high relative to the other connections in the graph.

4 Human Computer Interaction

As described before, we developed an application for doctors to browse through genetic information. A big challenge in this assignment was to make our product intuitive for the end users: people that have a completely different background then we, the developers, have. We couldn't just deliver a product, and hope it was good, we had to have some feedback from a person that was not involved in the project (but still had biological background).

4.1 Our action plan

After meeting with Willem-Paul Brinkman, we decided that we were going to organise an empirical evaluation. This evaluation should be of the type experiment, the experiment would make use of the **think aloud protocol**. We had a fellow student, Max, test our application. He was the most suitable person we could find, as he studies nanobiology at the TU Delft. We were going to give him a computer with our application on it, and give the following assignment:

> *You want to browse through the genome of patient John Doe, he is sick. He has the symptoms of Marfan's syndrome, can you look if you can find any traces of this sickness?*

We deliberately added the patient to the application, because adding a new patient and uploading his SNPs aren't the main concern in our project.

4.2 The realisation of our experiment

As expected, our test person didn't have any trouble to find his way through our dashboard (See Figure 1), this dashboard has a clear "Getting started" instruction that easily helps the user to navigate to the Patients page. Also when at the Patients page (See Figure 2), Max was able to directly navigate to the patient John Doe.
When clicking on the patient John Doe, the patient overview page appeared (See Figure 3). Here are a couple of quotes that our testing person made:

1. *"I see browse mutations, and a lot of vertical red lines. I suppose these lines represent the mutations that are found in the patient".*

2. After hovering over these mutations, as Max called them (they represent chromosomes), a new block with the title Mutations in chromosome X appeared, it became clear that Max was wrong. *"Ah, I see, these vertical lines represent chromosomes. Not that obvious since the header says: Browse mutations".*

Initially, our test person was a bit confused by all the information that appeared. What is a rsID, what is a frequency and what is the CADD score? After some explanation (we basically told that the CADD score is an indication about how harmful a mutation could be), Max immediately sorted the column with mutations on the CADD score, and clicked on the mutation with the highest value. Our tester saw the mutation view (See Figure 4 and Figure 5). Here are a couple of useful quotes that the tester made:

3. *"I see a big vertical red line with a id on it, and a couple of horizontal lines next to it. I can imagine that the red line symbolizes the mutation, and that the black lines symbolize the proteins around it".*

4. *"So I guess that this central green bulb symbolizes the protein that has a mutation, but I'm not really sure how I can get more information to confirm that this is the mutation that causes the Marfan syndrome".*

Our tester was clearly not going to come further, we explained that the green bulb was clickable and useful information appeared. We stopped the experiment and started with the debriefing.

4.3 Debriefing

The first thing we asked Max was: "So what did you think of our application?" He was quite positive, and found most parts were quite intuitive. The main problem in our experiment was that Max wasn't exactly the end user. A couple of terms (like CADD score) were terms that he had never heard of. We asked if there were any observations that Max had made, and if he wanted to share them with us. Max answered that he didn't have a lot, he did think that the navigation in a mutation view could be more intuitive.

4.4 Conclusion

The numbered quotes that Max made and are included in this report are all quotes that we used in making our product better. Quote number one for example made us realise that the heading "Browse Mutations" is inappropriate when it is placed above 23 pairs of chromosomes, this was confusing an thus we renamed it.
Quote number three was a confirmation that our mutation overview is quite intuitive, although quote number four still gave us some room for improvement: we should make it clear that the round Proteins are clickable. We improved this by implementing a tooltip that pops up when hovering over the protein.

5 Evaluation of the functional models

This section describes the evaluation of the functional modules and how they are tested to guarantee that each module performs as required. Furthermore, the failure analysis is discussed to show where the product does not perform as needed.

5.1 Evaluation of functional models

All of our functionality that is implemented, is tested using the test suite of the play framework. Each model is simply tested with JUnit, where every bit of functionality is checked for functioning correctly.

The controllers and views are tested by running a fake play application, which starts an application in the background so we can call the controllers and view. We check if each request to the controllers results in the expected action, such as a "Ok", "Page not found", "Redirect" or "Bad request". Next we check

the content of the rendered page, this should match the information we sent to this page. For the views only the content is checked.

5.2 Failure analysis

We rely a lot on data coming from other systems. First of all we have to read VCF-files to load the data of a patient. This is done by a doctor which could load a corrupted file to our server. This causes the system to stop processing the current file and the error will be written to the log. At the moment there is no notification to warn the user of this error, however this could be simply added in a future release.

Second we need access to external databases to retrieve relevant data. As these databases are too large to host them on the same webserver as the application, we cannot expect to always have a connection to the remote server. However we also store basic information in a small database on the webserver which stores the basic information such as authentication, user data and patient data. This helps us to always have a basic running system. If for some reason the connection to the database is dropped the application will continue to function, but no new information is loaded. In the future, useful notifications could be shown to the user to indicate such problems.

Thirdly, a user could direct his web browser to an invalid URL, in this case a "Not found" page will show, to prevent retrieving private information from the web-application. Also when something goes wrong in the application, the user will be sent to an "Error" page, which will only show a basic error and not the full error trace, which could expose our application logic.

6 Outlook

GEVATT is developed in quite a short time. The course Contextproject lasts about ten weeks and in this ten weeks the team had to become skilled in the context, make accompanying documents and present the final product. So only a limited part of the time is actually used to develop the application (week 4 to 9). This means the team focusses on the main parts of the application and there is more functionality that could be added later if development were to continue. This chapter is about what functionality could be added.

One thing is exporting; this is currently not possible in the application. The found mutations and information from the VCF files are only visible in the application itself and require logging in. For a doctor, it might be handy to export some data and visualisations to make it easier to present them. E.g. it's easier for a doctor to show a print of visualisations to his patients than showing a screenshot of the application. It also might be handy to have the data exported to a spreadsheet format.

The application is web-based and therefor principally platform independent: each device with a modern browser and an internet connection could open the application. Something that is limited to the use of the application now is the minimum screen width. When a device with a resolution of lower than 1024px is used, the user need to scroll to see all information. The application could be modified so that the layout adopts to smaller screens and become thereby

on lower resolution tablets and phones. A nice feature, but it is questionable whether a doctor would use the application on such devices.

Currently the application is mainly focused on individual mutations found in the uploaded data, and less on the relations between these mutations. If development of this application were to continue, it will probably make sense to focus on implementing/extending visualisations to show these relations.

Testing is only done on a low scale. If GEVATT would be used in production, there should be some testing be done based on heavy use. Furthermore, if the application would be used in production, some managing features should be added. For example, creating accounts en revoking access to specific doctors. Due to time considerations this is left out in this project.

7 Apendix A: Screenshots

Here are a couple of screenshots of our application.

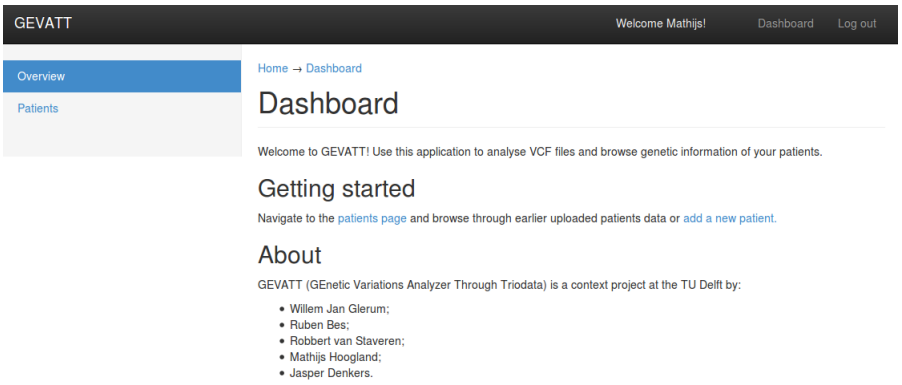


Figure 1: A screenshot of the dashboard

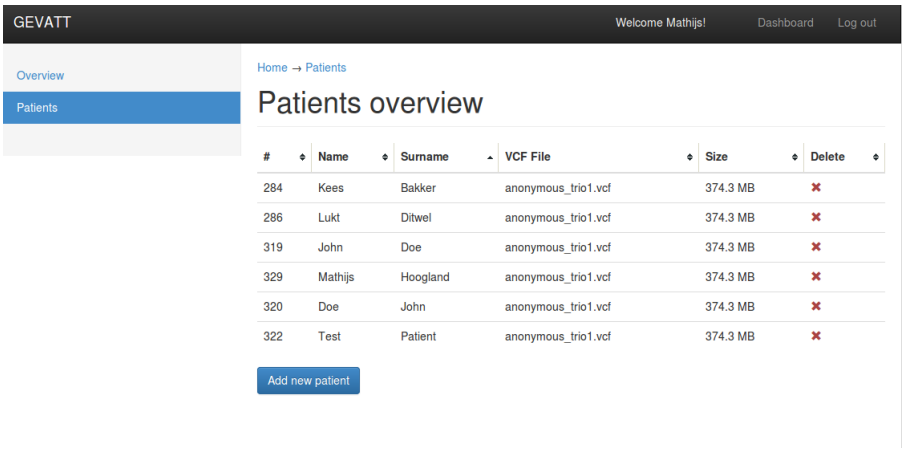


Figure 2: A screenshot of the patients view

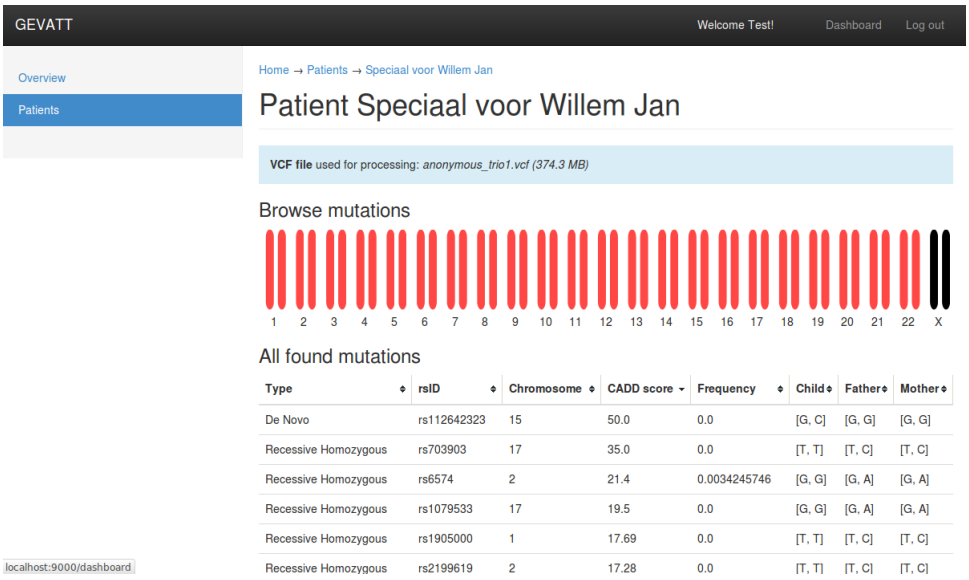


Figure 3: A screenshot of the patient view

Figure 4: A screenshot of the mutation view

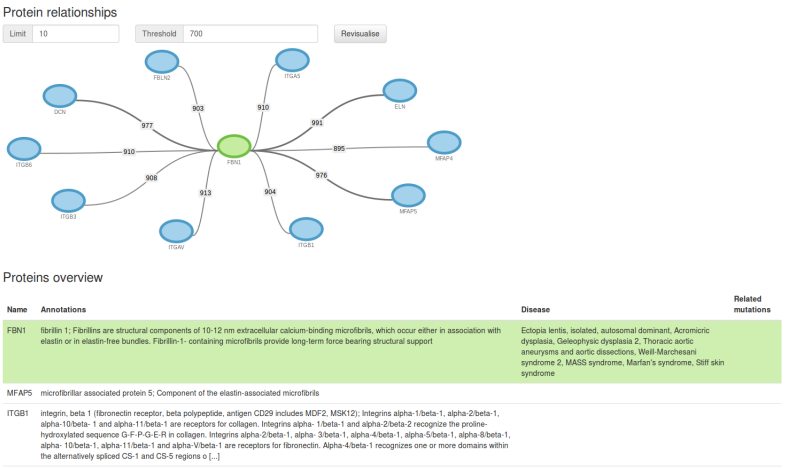


Figure 5: A second screenshot of the mutation view