
CRAMER: A lightweight, highly customisable web-based genome browser supporting multiple visualisation instances

M. Anastasiadi¹, E. Bragin², P. Biojoux¹, A. Ahamed¹, J. Burgin¹, K. de Castro Cogle¹, S. Llana-Lago¹, R. Muvunyi¹, M. Scislak¹, I. Aktan¹, C. Molitor¹, T. Kurowski¹, F. Mohareb^{1*}

¹ School of Water, Energy and Environment, Cranfield University, College Road, Cranfield, UK, MK43 0AL

² Next Gen Diagnostics, Wellcome Trust Genome Campus, Hinxton, UK, CB10 1SA

*To whom correspondence should be addressed.

Abstract

Summary: In recent years the ability to generate genomic data has increased dramatically along with the demand for easily personalised and customisable genome browsers for effective visualisation of diverse types of data. Despite the large number of web-based genome browsers available nowadays, none of the existing tools provide means for creating multiple visualisation instances without manual set up on the deployment server side. The Cranfield Genome Browser (CRAMER) is an open-source, lightweight and highly customisable web application for interactive visualisation of genomic data. Once deployed, CRAMER supports seamless creation of multiple visualisation instances in parallel while allowing users to control and customise multiple tracks. The application is deployed on a Node.js server and is supported by a MongoDB database which stored all customisations made by the users allowing quick navigation between instances. Currently, the browser supports visualising a large number of file formats for genome annotation, variant calling, reads coverage and gene expression. Additionally, the browser supports direct Javascript coding for personalised tracks, providing a whole new level of customisation both functionally and visually. Tracks can be added via direct file upload or processed in real-time via links to files stored remotely on an FTP repository. Furthermore, additional tracks can be added by users via simple drag and drop to an existing visualisation instance.

Availability and Implementation: CRAMER is implemented in JavaScript and is publicly available on GitHub on <https://github.com/FadyMohareb/cramer>. The application is released under an MIT licence and can be deployed on any server running Linux or Mac OS.

Contact: f.mohareb@cranfield.ac.uk

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Genome browsers have proven popular within the life-science community mainly because they provide interactive visualisation of genomic data integrated with aligned annotations, enhancing the ability to intuitively extract and recognise biologically interesting patterns in large and complex datasets. Genome browsers were first introduced with the aim to visualise the large amount of genomic data generated by the Human Genome Project as well as other model organisms. Examples of web-based genome browsers include the UCSC Genome Browser, Ensembl Genome Browser and NCBI Map Viewer (Coordinators 2017; Kent et al. 2002; Zerbino et al. 2018), which are among the most well-known multiple-species genome browsers, as well as some species-specific genome browsers such as TAIR, and Wormbase (Berardini et al. 2015; Lee et al. 2018). Among the

most desirable features of genome browsers is the extensive list of input file formats and generated tracks with annotation data which enables researchers to visualise and browse entire genomes with a high level of detail including gene model, expression profiles, regulation and genetic variation. Despite the popularity of web-based genome browsers there is an ever-increasing need for customised visualisation of diverse types of data which is not easily met using centralised web-based browsers. Standalone genome browsers such as the Integrative Genome Viewer (IGV) (Robinson et al. 2011) and NCBI Genome Workbench (www.ncbi.nlm.nih.gov/tools/gbench/) can provide a higher level of customisability and are more dynamic compared to web-based genome browsers. Alternatively, there are publicly available genome browsers such as JBrowse (Buels et al. 2016) developed for viewing genomes within a web browser. The advantage of these applications is their inherent

capability to share genome and genome annotations with a group of people, but they require expert knowledge to set up and run.

Herein, we present the Cranfield Genome Browser (CRAMER), a highly customisable standalone web-application for genomic data visualisation. The browser is capable of displaying visualisation Tracks added via direct file upload or processed in real-time via links to files stored remotely on an FTP repository or via the Ensemble API. Additional tracks can be added by users via simple drag and drop to an existing visualisation instance. Furthermore, CRAMER allows users to set up multiple visualisation instances simultaneously, while customising the appearance and behaviour of individual tracks.

2 Cranfield Genome Browser - CRAMER

CRAMER is a standalone web-application deployed on a Node.js server which allows the application to respond quickly and efficiently to multiple queries on the client-side, in real time. The application is easily customisable by automatically creating a new instance each time the system administrator wishes to display another set of data without the need to manually change the source code (Fig 1). For this purpose, a MongoDB was set up to store users' details and instances generated which the users can access and modify at any time. Additionally, tracks can be added using ftp file links as data input and are processed for visualisation in real-time; therefore preventing the need of transferring large files over for visualisation. To achieve this, embedded SAMtools and tabix (H. Li et al. 2009) commands are executed on the server side upon visualisation requests for a given chromosome and genomic region. CRAMER's visualisation engine is largely based upon the publicly available HTML5 genome browser Genoverse (<http://genoverse.org>). A list with the implemented tools is presented in Table 1 in Supplementary Materials. Because installation of these dependencies can be an arduous process for non-expert users, the application and all its dependencies have been packaged as a docker image.

3 Visualisation features

Currently, CRAMER supports the following types of data files: FASTA, BED, BIGBED, BAM, WIG, BIGWIG, GFF, GFF3, GTF, VCF and RSEM genes.results (B. Li and Dewey 2011); generating a total of 16 track types. Three new tracks have also been added in this release; namely a "Read Coverage" track, a "SNP density" track and a "Gene Expression" track. Furthermore a "Search" functionality is available which allows the user to move to a different position in the chromosome, or search for a specific gene or group of genes.

Both the plugins and tracks can be configured on the instance webpage which consists of a combination of a selected reference genome, a choice of the plugins to implement, and a selection of the tracks to visualise. In order to limit the allocated storage space for visualisation instances on CRAMER, which is primarily intended for single user or small to medium size research groups, only the administrator can add users to the application minimising the risk of unintended modification or deletion of existing instances by unauthorised users. Registered users can sign into their personal account to create new instances or view and modify already existing ones.

Conclusions

CRAMER is an interactive, fast and user-friendly web-based genome browser able to run multiple visualisation instances in a highly customised manner. The genome browser is light-weight, and offers a smooth navigation combined with a high degree of interactivity.

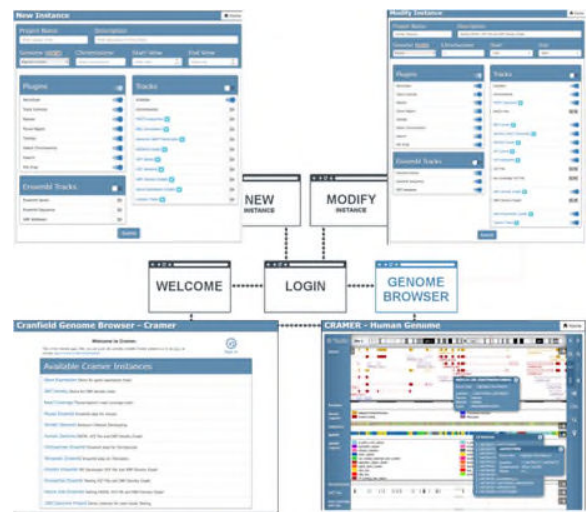


Fig. 1. The CRAMER Workflow: The application is capable of hosting multiple visualization instances simultaneously without the need to manually change the source code.

Acknowledgement

We would like to thank Simon Brent and the rest of Genoverse's team for their support during the process of developing CRAMER.

References

- Berardini, T. Z., et al. (2015), 'The Arabidopsis information resource: Making and mining the "gold standard" annotated reference plant genome', *Genesis*, 53 (8), 474-85.
- Buels, R., et al. (2016), 'JBrowse: a dynamic web platform for genome visualization and analysis', *Genome Biol*, 17, 66.
- Coordinators, Nebi Resource (2017), 'Database Resources of the National Center for Biotechnology Information', *Nucleic Acids Res*, 45 (D1), D12-D17.
- Kent, W. J., et al. (2002), 'The human genome browser at UCSC', *Genome Res*, 12 (6), 996-1006.
- Lee, R. Y. N., et al. (2018), 'WormBase 2018: molting into a new stage', *Nucleic Acids Res*, 46 (D1), D869-D74.
- Li, B. and Dewey, C. N. (2011), 'RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome', *BMC Bioinformatics*, 12, 323.
- Li, H., et al. (2009), 'The Sequence Alignment/Map format and SAMtools', *Bioinformatics*, 25 (16), 2078-9.
- Robinson, J. T., et al. (2011), 'Integrative genomics viewer', *Nat Biotechnol*, 29 (1), 24-6.
- Zerbino, D. R., et al. (2018), 'Ensembl 2018', *Nucleic Acids Res*, 46 (D1), D754-D61.

Cramer: A lightweight, highly customisable web-based genome browser supporting multiple visualisation instances.

M. Anastasiadi¹, E. Bragin², P. Biojoux¹, A. Ahamed¹, J. Burgin¹, K. de Castro Cogle¹, S. Llaneza-Lago¹, R. Muvunyi¹, M. Scislak¹, I. Aktan¹, C. Molitor¹, T. Kurowski¹, F. Mohareb^{1*} —

¹ The Bioinformatics Group, School of Water, Energy and Environment, Cranfield University, College Road, Bedford, MK43 0AL, UK.

² Next Gen Diagnostics, The BioData Innovation Centre, Wellcome Trust Genome Campus, Hinxton, UK, CB10 1SA

—

* To whom correspondence should be addressed.

Contact: f.mohareb@cranfield.ac.uk.

—

Supplementary Material S1: User Manual

The Cranfield Genome Browser - CRAMER

Introduction

CRAMER is a customisable, JavaScript and Jade based genome browser for interactive exploration of genomic data. Data is visualized in the browser, meaning CRAMER can be installed on any website and show data from a wide range of online, ftp links or local sources. CRAMER works with a variety of formats, such as XML, JSON, BED, VCF, GFF, GFF3, BAM or delimited text files, and can be customised to parse and display any data source as required.

The application and source code is freely available to download from the CRAMER Github repository: <https://github.com/FadyMohareb/cramer>.

Table of Contents

How to install CRAMER

Dependencies

Install on Linux using Docker

Install on MacOS

Manual Installation

CRAMER 2.0 workflow

How to create a new user account

Tracks

How to navigate CRAMER 2

Welcome page

Genome browser page

Hands-on Example

Common errors

Requirements for the correct display of imported files

List of Ensemble Genomes not currently available to display

How to install CRAMER

This section provides detailed instructions on how to install and execute the program, so it can be accessed from any browser. This can be done using: 1. Pre-built Docker image download 2. Built a new Docker image using the latest version available on Git 3. Manual installation

Dependencies

CRAMER requires running on Linux or MacOS. The programme requires NodeJS and three commonly used bioinformatics programs.

- NodeJS LTS (The application was also tested on the latest current version 12.12.0)
- Samtools
- Bwtool
- kentUtils from UCSC

Make sure that they are available on the \$PATH. (Typically installed under /usr/local/bin). To check whether this is the case, type

```
echo $PATH
```

This should include (among other paths) `usr/local/bin`

How to install CRAMER on Linux:

Option 1: Docker-based installation

a. Run CRAMER using pre-built docker image

For this, you need to have Docker service available on your machine. To check if that is the case, you can run:

```
sudo service docker start
```

Verify Docker:

```
sudo docker run hello-world
```

This docker image assumes there is a MongoDB service running on localhost:27017

1. Make sure mongo is up and running on Docker:

```
sudo docker run --name mongo -p 27017:27017 -d mongo:latest
```

Alternatively, if you have already created the *mongo* container, run:

```
sudo docker start mongo
```

2. Run CRAMER via the docker image:

```
sudo docker run --rm --name=cramer --pid=host --network=host digimeow/genoverse2
```

You may now visit <http://localhost:4000> in your web browser.

b. Build a new docker image based on the latest git version

The easiest way to satisfy all dependencies and run CRAMER is to build and run the Docker image defined in Dockerfile within this repository. For this, you need to have Docker service available on your machine. To check if that is the case, you can run:

```
sudo service docker start
```

Verify Docker:

```
sudo docker run hello-world
```

This docker image assumes there is a MongoDB service running on localhost:27017

1. Make sure mongo is up and running on Docker:

```
sudo docker run --name mongo -p 27017:27017 -d mongo:latest
```

Alternatively, if you have already created the *mongo* container, run:

```
sudo docker start mongo
```

2. Clone this repository locally:

```
git clone https://github.com/FadyMohareb/cramer.git && cd cramer
```

3. Build the application:

```
sudo docker build . -t cramer
```

4. Run CRAMER:

```
sudo docker run --rm --pid=host --network=host cramer
```

You may now visit <http://localhost:4000> in your web browser.

**In order to start adding instances to your running application, you need to add a new admin user to the application back-end as follows:

1. Stop the application using Ctrl+C in the terminal window running the CRAMER image.
2. Install the npm dependencies by running:

```
npm install
```

3. Add a new admin user using the following command:

```
npm run newUser <user@example.com> <password> <John>
```

For example: `npm run newUser admin@admin.com adminadmin admin`

4. Build the application:

```
sudo docker build . -t cramer
```

5. Run CRAMER:

```
sudo docker run --rm --pid=host --network=host cramer
```

Docker-based installation (MacOS)

Although it is possible to get the docker image to work on Mac, the process can be tricky and not very stable. This is mainly due to the fact that the host networking driver currently only works on Linux hosts (See more information here: <https://docs.docker.com/network/network-tutorial-host/>). It's possible to use a different port for the application, however access to the "helper" tools such as `tabix` and `samtools` can be tricky. Therefore, we do recommend the manual installation on Mac, and this is quite straight forward as shown below:

Option 2: Manual installation

First of all, make sure you have all the Dependencies installed. Make sure that they are available on the `$PATH`. (Typically installed under `/usr/local/bin`). To check whether this is the case, type

```
echo $PATH
```

This should include (among other paths) `usr/local/bin`.

1. Clone the Github repository with the following command:

```
git clone https://github.com/FadyMohareb/cramer
```

2. To install, run the following commands:

```
cd \<cramer/Directory\>
```

```
npm install
```

```
node bin/www
```

3. The output should be:

```
Running at port: \<port\>
```

```
MongoDB connection open
```

4. Then, it could be accessible from any browser on any operating systems.

Open your browser and write this in the URL address. Make sure the port value is the same than in the output displayed in the prompt command:

```
localhost:\<port\>
```

OR

```
IPAddress:\<port\>
```

To begin adding visualisation instances, you need to add a new admin user using the following command:

```
npm run newUser <user@example.com> <password> <John>
```

For example: `npm run newUser admin@admin.com adminadmin admin`

Once installed, the multiple files that form the program can be altered to change the functionalities of the program as the developer team wants, and then saved to apply these changes. However, modified files which are contained in *list-js.js* would require building the program before running it to apply the changes. This can be done with the following command:

```
$ npm run-script build
```

CRAMER workflow

CRAMER consists of multiple web pages backed by a node server, and a mongo database. It provides a number of web pages to assist with setting up and visualising genomic tracks (**Fig. 1**), with the actual genome browser page as the main gateway for the program. It is made up of a set of tracks – horizontal sections of the browser which display features, such as genes or variants, – which share the same start and end position. These features can either be fetched from Ensembl or extracted from a local or FTP file. Currently, the browser supports the following types of data files: FASTA, BED, BIGBED, BAM, WIG, BIGWIG, GFF, GFF3, GTF, VCF and RSEM genes.results. In addition, CRAMER has a series of plugins which add extra functionalities.

The plugins and tracks can be configured on the instance webpage. An instance is a combination of a chosen genome, a choice of the plugins to implement, and

a selection of the tracks to visualize; that will be displayed in the browser. The genome data could be obtained from a local file or through Ensembl database.

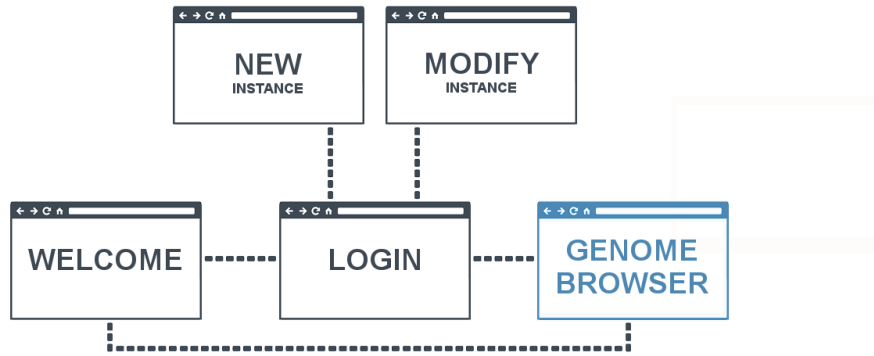


Figure 1: C:\Users\Sergio\Desktop\forsergio.png

Fig.1. Diagram of the relation between the multiple webpages that form CRAMER

The welcome webpage allows registered users to create new instances or view and modify already existing instances. Non-registered users are still able to view and explore existing instances however they do not have the right to create or modify instances. Our intention is not to provide the system users with means to register and add their own instances; instead, this functionality is reserved to the genome browser administrator (or group of administrators) to manage, on behalf of the research group. The reasoning behind this is to control the size of the database and storage space on the deployment server, by preventing the users uploading duplicating genome files and/or large bam files which may result in running out of disk space quite quickly with genomic data. Since all the webpages are connected to an external server and a database, the data that is introduced from any device is stored in the database and is readily accessible from any other device.

File requisites

The numerous file types that the genome browser can support must adhere to certain formatting rules. Any of the changes that are required to convert the files into compatible files are to be carried out by the user prior to loading them into the web application. The requirements for compatibility for each file type are the following:

- **GFF, VCF, and BED:** These files must be gzipped and indexed using tabix with the index file in the same directory [1]. The program will be able to automatically distinguish the characteristic chromosome IDs from the full chromosome names of the file, as long as the IDs are at the end of

the name (for example SL2.50ch01).

- **FASTA:** These files must be indexed using `faidx` with the index file in the same directory [2], and the chromosome names have to be formatted as `chr[num]` (for example `chr1`).
- **RSEM genes.results:** Matching gene IDs should be included in the GFF file that is loaded with this file format as the genomic coordinates of the genes will be fetched from it.
- **Genomic BAM:** These files must be gzipped and indexed with SAMtools with the index file in the same directory [2]. Like with tabix indexed files, the program will be able to automatically distinguish the characteristic chromosome IDs from the full chromosome names of the file, as long as the IDs are at the end of the name (for example SL2.50ch01). This can be pre-processed using the “`processingBAM.sh`” script.
- **read coverage BIGWIG:** These files can be made compatible with CRAMER by using the “`processingBAM.sh`” script that is included with the program. This script automates the steps to configure the BAM file and generate a BIGWIG file for FTP loading, which can be loaded into the program.

The script can be executed by typing `./processingBAM.sh input.bam` in a Linux terminal, and following the instructions that will be prompted on the screen.

Beware that for using the script and loading the file via FTP correctly the chromosome IDs must be at the end of the chromosome name (for example SL2.50ch01) as indicated for previous files. The script requires the following tools to be fully executed: `samtools`, `bam2wig`, and `ucsc-wigtobigwig` (installed through `bioconda`) [3]. Any BIGWIG that follows this format can be used.

In addition to this, local files can be dragged and dropped in the genome browser window. The drag-and-drop action is available for the following types of files, with the prerequisite that their indexed file is also stored in the same location: **GFF, GFF3, GTF, VCF, BED, BIGBED, WIG, BIGWIG** and **BAM** files can all be dragged and dropped into the user interface but their format is restricted to chromosome numbers formatted as `chr[num]` or `[num]` (for example `chr01` or `1`).

How to create a new user account

The administrator can create new user accounts via the command line by typing the following:

```
npm run newUser <user@example.com> <password> <John>
```

The required inputs are the user’s email address, the password and a user name.

Sign in

Registered users can sign in by clicking on the sign in link on the home page and entering their login credentials on the login page (**Fig. 2**).

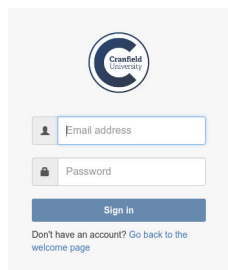


Fig.2. Login page for registered users.

Tracks

Currently, CRAMER has sixteen different tracks available that can be divided into three groups: **Standard** tracks, **Ensembl** tracks, and **File** tracks.

There are two **Standard** tracks which can work without the need to fetch data from Ensembl or upload a file:

- Scalebar: Displays a bar which provides information regarding the current position in the chromosome.
- Chromosome: Displays the karyotype of the selected chromosome.

The **Ensembl** tracks are only available when the user chooses a genome from the Ensembl database. There are three Ensembl tracks:

- Ensembl Genes: Fetches data from Ensembl and displays the genes that are present in the chosen chromosome positions. It also displays a legend for each gene.
- Ensembl Sequence: Fetches data from Ensembl and displays the nucleotide sequence of the chromosome. Each base has a different colour assigned. Adenine (A) is green, Thymine (T) is red, Cytosine (C) is blue, and Guanine (G) is yellow.
- SNP Database: Fetches data from Ensembl and displays the single nucleotide polymorphisms (SNPs) present for the selected positions of the chromosome. It also displays a legend.

The **File** tracks work in a different way to the others. A user can generate multiple tracks of each type with a name, a short description, and an URL or file path to the file that the user wants to display on the browser. Each track type has different configurable properties, such as the threshold for example, due to

the nature of the file. Each track type can be individually activated, deactivated and any tracks created can be removed. There are eight **File** tracks available:

- FASTA sequence: Displays the chromosome sequence from a FASTA file.
- BED Annotation: Displays the positions and details of annotation from a BED file.
- Genomic BAM Transcripts: Displays the sequences of reads from a genomic BAM.
- BIGWIG Graph: Displays the contents of a BIGWIG file as a graph.
- GFF Genes: Displays a GFF file with all the components forming a gene with exons and introns represented.
- VCF Variants: Displays the position of each variation and whether they are higher or lower than the quality threshold set. Requires a VCF file.
- SNP Density Graphs: Displays the density of SNPs over a range of positions. Requires a VCF file.
- Gene Expression Graphs: Displays the predicted counts at the position of each gene.

Finally, there is an additional track available within the **File** tracks, called **Custom** track. A **Custom** track must be written as a JavaScript object, which will be implemented into the program. There is no limit to the number of **Custom** tracks a user can create.

Plugins

Table 1. provides a summary and short description of the plugins available in CRAMER.

Table 1. Plugins included in CRAMER.

Plugin

Implements:

Karyotype

An ideogram of the selected chromosome on the top of the browser.

A marker on the ideogram that reports the current position in the chromosome. This marker can be dragged and resized, which will update the positions of the tracks to match the positions of the markers.

Track Controls

An additional menu in each track which provides the option to display info about the track, or to delete it.

Resizer

A handler at the bottom of each track that can be vertically dragged to resize the track.

Focus Region

A button on the control panel that resets the start and end positions currently displayed to the default ones of the instance.

Tooltips

A button that displays a brief description of each element in the genome browser.

Select Chromosome

A button that displays a list of all the chromosomes for the visualised genome. A chromosome can be displayed by clicking on the chromosome number on the list.

Search

A button that displays a pop-up window which provides the option to search by gene name, gene position, gene name AND position, or ENSEMBL id in the currently displayed chromosome.

File Drop

Allows users to drag and drop a file -in a format supported by CRAMER - directly into the browser and display it.

How to navigate CRAMER

Welcome page

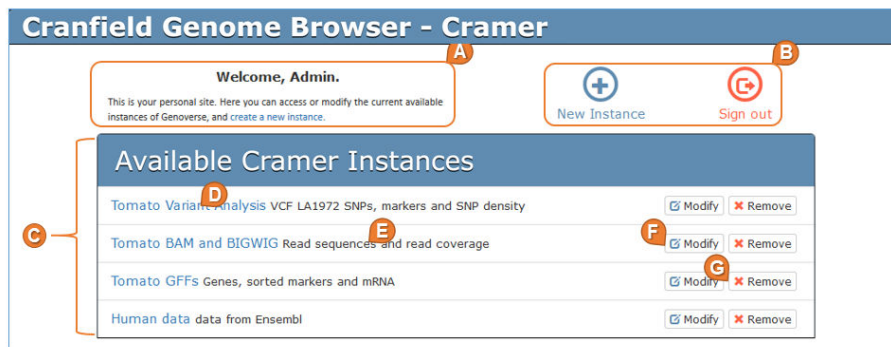


Figure 2: C:\UNIVERSIDAD\Applied Bioinformatics\Group Project\STARTING Bioinformatics\Group Project\STARTING DEF\User Guide\Screenshots\Screenshot_Welcome_page.png

Fig.3. Welcome page of the program for a logged-in user.

The welcome page is where users can access existing instances. Registered users can also log-in which provides them with the option to create, modify or remove instances (**Fig. 3**).

On the top of the page, there is a brief welcome message and description of what the user can do (**A**). Next to it, on the right side, depending on whether the user is registered or not there can be different buttons (**B**):

- Sign in: Registered users can sign in to unlock additional actions; this will direct them to the login webpage.
- Create a new instance and Sign out: Logged-in users can create new instances, which will redirect them to the instances webpage, or sign out of their accounts.

A list of all the available instances is displayed in the centre of the welcome (**C**). The names of the instances are highlighted in blue (**D**), followed by a brief description of each instance in black (**E**). The name of the instances (**D**) are also links to the instance deployment on the genome browser itself. If the user is logged-in there will be two additional buttons:

- Modify: A registered user can modify instances that are already created (**F**). This button will redirect the user to the instances webpage.
- Remove: A registered user can delete an instance that is already created (**G**). Clicking on the button will make a pop-out window asking confirmation appear, if the user clicks on “OK”, the instance will be deleted.
- **Instance page**

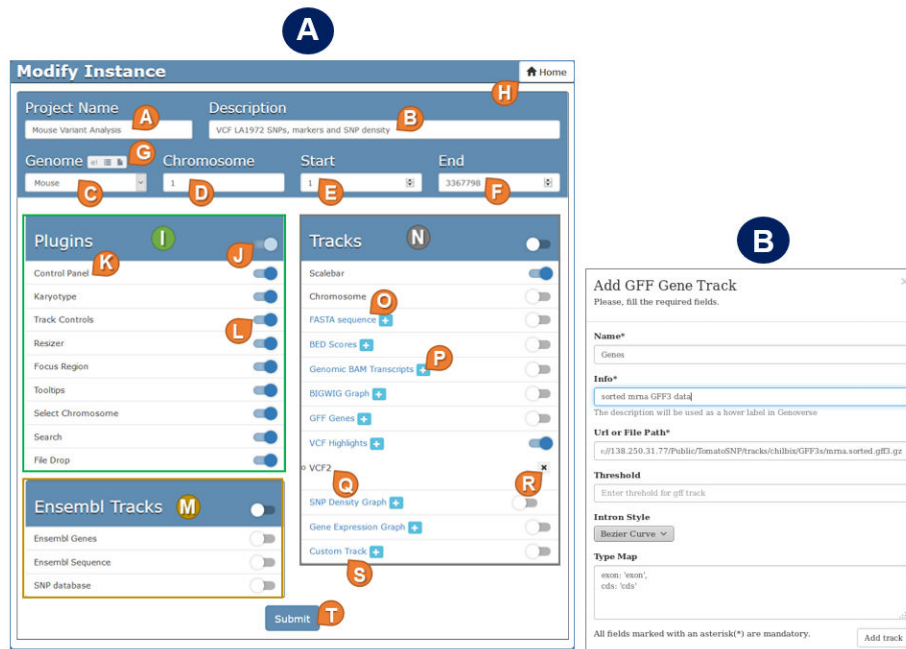


Fig.4. (A) Instance page with examples of how to fill each field. (B) Pop-up window to create a new sub-track in the GFF Gene parent track.

The instance page is accessed through the “Create new instance” or “Modify” button on the welcome page (**Fig. 4A**). This page is only available for registered users and allows creating or modifying an instance. When a new instance is being created, all the fields will be empty. In contrast, if an instance is being modified all the previous configurations will be provided in each field.

In the fields marked “Project Name” (A) and “Description” (B) the user can insert the instance name and a short description which will be displayed in the welcome page. Other fields required to be filled include: “Genome” (C), for the species genome, “Chromosome” (D) for the chromosome number in which the instance will be initialized (D), “Start” (E) and “End” (F) which need to be filled with the start and end positions of the chromosome to be displayed. The species genome karyotype (C) can be fetched from the Ensembl database, uploaded from a local file or selected from a list of already available genomes. These options can be switched by a button next to the genomes (G), which is formed by three sub-buttons, one per each option. The uploaded file needs to follow the format described in the Appendix.

On the right top corner of the screen, there is also a “Home” button (H) which redirects the user to the welcome page.

Other features of the instance page include a list with the available plugins for the current instance (I), and a checkbox to activate/deactivate individual plugins

(K, L).

In the case where an Ensembl species genome has been selected, an additional panel will appear underneath the “Plugins” entitled “Ensembl tracks” (M). The functionality of this type of tracks has been described in the Tracks section above.

The other two types of tracks **Standard** and **File**, are grouped together in the “Tracks” box (N) adjacent to the “Plugins” box. The file tracks are highlighted in blue (O) and next to them there is a blue plus-shape button (P). Clicking on this button will display a pop-up window which allows creating a track by filling all the required forms (Fig. 4B). Once a track is created, clicking on the track type (O) will reveal all the tracks of this type (Q). A track can be deleted by clicking on the remove button (R). In the case where the user wishes to display data from a local file on a track, a track with an FTP file link needs to be created. It is the responsibility of the user to write the FTP link to the correctly formatted file in order to get a correct track display. In addition, a custom track can be created by selecting the “Custom Track” option at the bottom of this list (S).

Finally, an instance can be saved in the database by clicking on the “Submit” button (T). If any parameters are incomplete or wrongly filled, a red alert will inform the user about the cause of the error.

Genome browser page

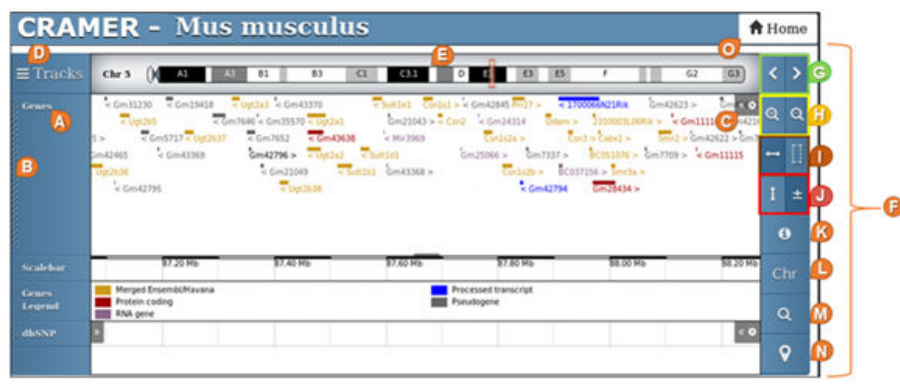


Fig.5. Genome browser page. In the browser is displayed the karyotype of the chromosome 5 of *Mus musculus* and the genes found in Ensembl for the current view position.

The genome browser page (Fig. 5) is where all the data of the instances is visually represented. The interface consists of a set of tracks, each representing a different type of data. On the left side of the browser window there is a vertical panel with labels for each track (A) and a small handler which allows the user to vertically drag the track and relocate it (B). On the right side of the browser

window, there is a small menu tab (**C**) which allows the user to get more info about the current track, adjust the height or remove the track altogether.

On the top corner of the left panel, there is a “Tracks” button (**D**) which displays a pop-out window where all the current implemented tracks are listed. From there, any implemented track can be removed (clicking on the “x” shape button) or added (clicking on the “+” shape button), even if it has already been added.

The karyotype of the chromosome is displayed on a ribbon located at the top of the browser window (**E**). Users can navigate through the chromosome by clicking and dragging on the red bookmark located on the karyotype. On the right side of the browser window there is a vertical panel with an arrangement of multiple buttons (**F**):

- Scrolling buttons (**G**): Changes the current view position of the chromosome to the left (<) or right (>).
- Zoom in/out buttons (**H**): Changes the view position to get more or less detail of the viewed area.
- Mouse drag action (**I**): Switches the outcome of the user dragging with the mouse. By default, when the user drags the mouse, the view position is scrolled through the chromosome. However, if the other option is activated, the drag mouse action will highlight the dragged area and zoom in.
- Mouse wheel action (**J**): Switches the outcome of the user spinning the mouse wheel. By default, this will make the user scroll over the web page, but with the other option, the user will make zoom in/out on the area where the mouse cursor is placed.
- Tooltips (**K**): Clicking on this button will display little text boxes with a brief description of each element on the browser interface.
- Select chromosome (**L**): This button will display a list of all the available chromosomes. Clicking on one of them will show it on the browser. By default, the button name is “Chr” but, once that is used, it will change to the name of the selected chromosome.
- Search (**M**): This button will trigger a pop-out window which allows searching genes by position, name, or ID. If the positions forms are not filled, the tool will search in the whole chromosome, while if a range of positions is specified, it will only look for the genes in that range. Through the three tick boxes, the search can be broadened to the GFF genes track (if it is loaded) or to Ensembl IDs and gene names, in case that the genome file is being fetched from Ensembl. Once the search is completed, a new pop-up window with all the matches will appear and, by doing click on one of the matches, the viewpoint will be moved to the position of that match.
- Reset focus (**N**): This button will reset the viewpoint to the default one of the instances.

To get back to the welcome page, users can click on the “Home” button on the top right corner of the browser window (O).

It is important to note that although the browser page allows users to drag and drop correctly formatted files directly into the browser to visualise them, these files will not be saved in the database.

Hands-on Example

Objective: to create a visualization instance for vcf files containing human variant calls from the International Genome Sample Resource (IGSR) and the 1000 Genomes Project repository.

Download vcf files from the IGSR repository

VCF files from IGSR can be downloaded from the FTP site hosted at the EBI `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/.` Both the VCF files and their corresponding indexed files (.tbi) are available to download eliminating the need to index the VCF files for use on CRAMER. To download the following files: “CHBJPT.low_coverage.2010_09.xchr.sites.vcf.gz” and “CHBJPT.low_coverage.2010_09.xchr.sites.vcf.gz.tbi” type:

```
$ wget ftp://ftp.1000genomes.ebi.ac.uk/ vol1/ftp/pilot\_data/paper\_data\_sets/a\_map\_of\_h\_coverage/snps/CHBJPT.low\_coverage.2010\_09.xchr.sites.vcf.gz*
```

—

As already mentioned in the “File requisites” section, VCF files must be compressed and properly indexed with tabix in order to be visualised on CRAMER.

N.B. In the case where a tabix file is not available, it can be generated by running the following command on a compressed vcf file:

```
$ tabix vcf\_file.vcf.gz
```

The generated tabix file must be uploaded together with its corresponding vcf file on CRAMER.

2. Create a new instance on CRAMER from Ensembl

Application administrator can log into CRAMER and create new instances or modify existing ones. To create a new instance, go to the welcome page and click the “Sign in” button. This will prompt you to add your credentials. Type your email address and password and press “sign in”. After signing in click on the “Create New Instance” button to open the instance page. Fill out the fields on the form (**Fig. 6**) to set a name and description for the new instance, select the chromosome to display and set the default starting and end points of view when the instance is displayed on the browser. All these details can be modified later. To fetch the human genome from Ensembl, go to the “Genome” region on

the top of the page and hover with the mouse over the three available tabs to see the description of what each does. Click on the first button (*e!*), to activate it. This option enables you to fetch genomes from Ensembl. Then go to the scroll down menu underneath and select “Human”.

Please note that not all the species genomes on this list can be fetched from Ensembl. The reason for this is that they are either not available in the database, or that their content is not correctly formatted. A list of Ensembl genomes not currently displayed on CRAMER is available in the Appendix.

Next you need to activate the “Ensembl Tracks” to display more information about the human genome. The available Ensembl tracks are “Ensembl Genes”, “Ensembl Sequence” and “SNP database” and their function has been described in the Tracks section. You can either activate all the Ensembl tracks at once or select individual tracks to be displayed.

The screenshot shows the 'New Instance' configuration page. At the top, there are input fields for 'Project Name' and 'Description'. Below these are fields for 'Genome' (a dropdown menu currently showing 'Algerian mouse'), 'Chromosome', 'Start View', and 'End View'. The main part of the form is divided into two columns of toggle switches. The left column, titled 'Plugins', includes: Karyotype, Track Controls, Resizer, Focus Region, Tooltips, Select Chromosome, Search, and File Drop. The right column, titled 'Tracks', includes: Scalebar, Chromosome, FASTA sequence, BED Annotation, Genomic BAM Transcripts, BIGWIG Graph, GFF Genes, VCF Variants, SNP Density Graph, Gene Expression Graph, and Custom Track. A 'Submit' button is located at the bottom center of the form.

Fig. 6 Form for new visualization instance on CRAMER

Add local files

Next, we need to visualize the files we have just downloaded in the previous step. In this example we will upload the “CHBJPT.low_coverage.2010_09.xchr.sites.vcf.gz” downloaded from IGSR. To create new VCF tracks follow the steps below:

1. Go to the “Track” box and activate the “VCF Highlights” tab.
2. Click on the plus button next to “VCF Highlights”.
3. Start populating the fields in the pop-up window. The ones marked with an asterisk (*) are mandatory and consist of:
 1. Name: Name of the track that is going to be created.
 2. Info: Brief description of the track that will be used as a hover label on the browser.
 3. URL or Filepath: The url or the filepath of the VCF file.

When the fields have been completed, press “Add Track”.

You can also create a SNP density track by activating the “Add SNP Density Track” and uploading the same VCF file.

Finally press the “Submit” button. This action will take you back to the instance list page. Scroll down and select the name of the instance you just created from the instance list. This will open the genome browser window (**Fig. 7**) and will display the data for the chromosome and specific region you selected when creating the instance. This information is also contained within the URL link for this page, eg. <http://localhost:4000/index?name=Human%20Genome%20Testing&r=X:35938492-35987711> The “r=X denotes chromosome X and **35938492-35987711** is the region currently displayed on the genome browser. The link is updated each time the user performs an action, such as zoom in/out or change chromosome (Please note that the vcf file we downloaded contains only variants on Chromosome X, so if you pick a different chromosome, the visualization instance would work fine, however, the vcf track will contain no SNPs). You can replicate the same view of the genome by copying and pasting the URL link in a new tab on the web browser.

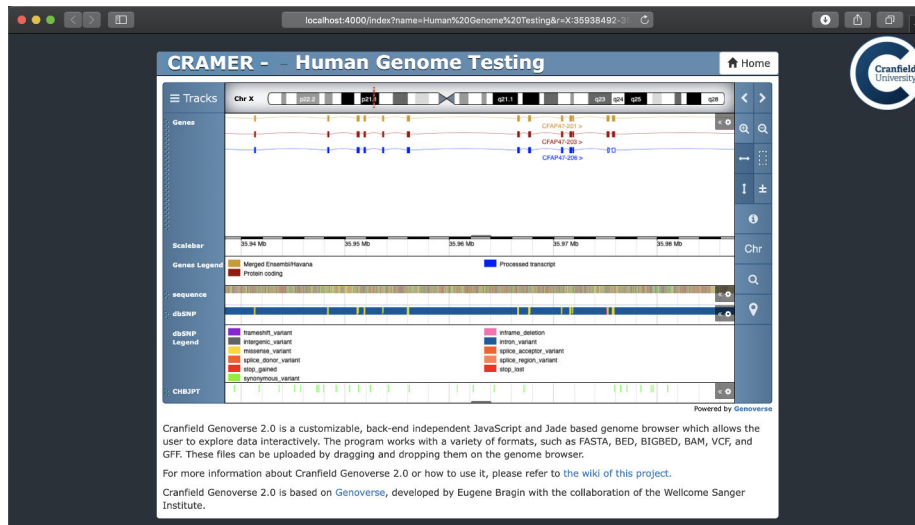


Fig. 7 Screenshot of the Human Genome visualisation instance showing tracks for: Chromosome X and 35938492-35987711

Drag and drop

An alternative way of uploading files on CRAMER is by using the “drag and drop” functionality which permits users to display data on the genome browser page for as long as the instance is displayed. The drag and drop functionality is a convenient way of displaying data without permanently adding them to the instance. Just select files with the proper format and drag them to the browser which will automatically generate the tracks displaying the data for that specific region. To replicate the example above, select both the “CHB-JPT.low_coverage.2010_09.xchr.sites.vcf.gz” file and its tabix indexed file “CHB-JPT.low_coverage.2010_09.xchr.sites.vcf.gz.tbi” and drag and drop them in the genome browser page. The name of the new track will be the same as the file name. In case you only selected the vcf file, the following error message will appear: “ERROR: GZipped VCF files must be accompanied by a .tbi index file”. Drag and drop can be used for other file types as well.

Explore Data

Once you have uploaded the data, there are a number of actions you can do on the genome browser page to explore the data as described in the “Genome browser page” section. For instance, you can search for a specific gene on a chromosome, e.g. PLAC1 on chromosome X, by first selecting the correct chromosome from the drop down menu and then clicking on the “Search” button and typing “PLAC1” in the name field (Fig. 8).



Fig. 8 CRAMER gene search form.

Then tick the fields “Ensembl gene IDs” and “Ensembl gene names” since the genome is fetched from Ensembl. After clicking on the arrow next to “Search” a list will appear containing information including the gene name and ID and the start and end positions. By clicking on the gene name on ID the browser will resize to display the area containing the gene of interest. For more information on the gene you can click anywhere on the gene schematic on the annotation track and a pop-up window will appear with additional information (Fig. 9). If you click on the gene name on the pop-up window, you will be redirected to the Ensembl webpage for this gene, which will open in a new tab.

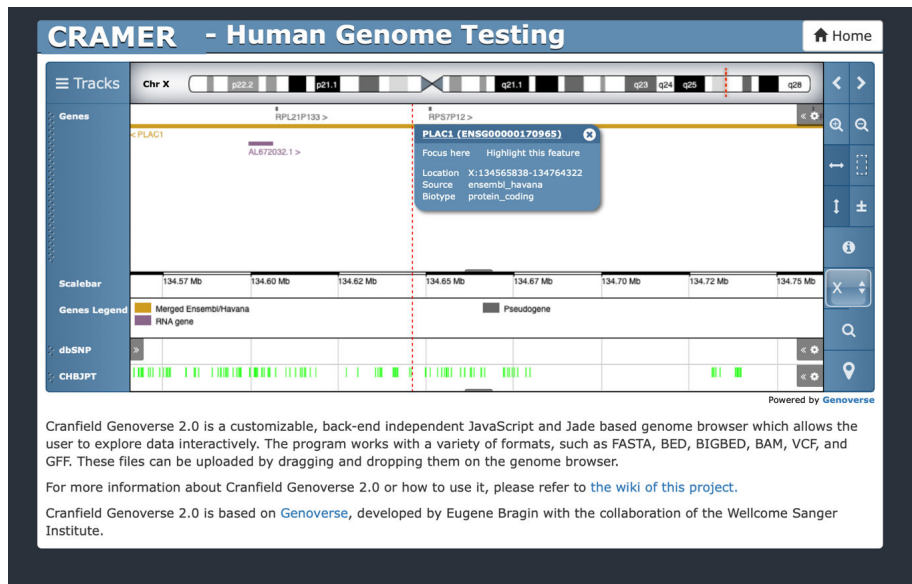


Fig. 9 CRAMER gene search result for PLAC1

Common errors

When trying to visualise an instance the following error messages may appear:

“Wrong Content in the Genome File” or “Genome File Does Not Exist.” and/or “the instance is blank”.

This error can be caused by:

A complex chromosome name such as “genescaffold:gadMor1:GeneScaffold_3590:1:3911206:1 REF”

- The chromosome positions have not been correctly established on the instance details.
- An incorrect URL/Filepath

When uploading a VCF file, the following error may appear: “*ERROR: Gzipped VCF files must be accompanied by a .tbi index file*”

This error means that the VCF file is not properly compressed.

In the genome browser page a warning may appear after you perform the drag and drop operation: “*Data for this track is not displayed in regions greater than 100.00 kb*”.

To remove this warning press the “zoom in” button repeatedly.

Appendix

Requirements for the correct display of imported files

The genome file needs to be in JSON Object format. The file needs to respect some requirements:

- start with ‘**Genoverse.Genomes.**’ and then the name of the genome;
- contain the name of the chromosome such as “**1**” or “**MT**”;
- in the chromosome element, the **size** and the **bands** are **required**. The size is the length of the chromosome and the bands can be filled with a list to display the karyotype.

Below, there is an example of the genome file for the tomato:

```
Genoverse.Genomes.tomato = {  
  "1": {  
    "size": 98543444,  
    "bands": []  
  },  
  "2": {  
    "size": 55340444,  

```

```
“bands”: []  
},  
“3”: {  
“size”: 70787664,  
“bands”: []  
}  
};
```

List of Ensemble Genomes not currently available to display (As of 20.10.2019):

Genome files not available

Alpaca, Algerian Mouse, Armadillo, Black snup-nosed monkey, Chinese Hamster CriGri, Chinese softshell turtle, Cod, Coelacanth, Coquerel’s sifaka, Damara Mole Rat, Dolphin, Flycatcher, Hyrax, Lesser hedgehog tenrec, Marmoset, Shrew, Sloth, Sooty Mangabey, Tarsier, Tilapia, Tree Shrew, Upper Galilee, Wallaby.

Genome file exists with wrong content

Amazon Molly, Angola Colobus, Bolivian Squirrel Monkey, Brazilian Guinea Pig, Bush baby, C. Savignyi, Capuchin, Cave Fish, Chinese Hamster CHOK1GS, Cow, Degu, Drill, Elephant, Ferret, Fugu, Golden Hamster, Golden snub nosed monkey, Guinea Pig, Hedgehog, Horse, Kangaroo rat, Lamprey, Lesser Egyptian Jebroa, Long-tailed chinchilla, Ma’s night monkey, Medaka, Megabat, Microbat, Naked mole rat female, Naked mole rat male, Northern American Deer Mouse, Panda, Pig tailed macaque, Pika, Platyfish, Squirrel, Stickleback, Tasmanian Devil, Xenopus.

References

1. Genome Research Limited. tabix manual page. 2018. <http://www.htslib.org/doc/tabix.html>. Accessed 29 Apr 2018.
2. Genome Research Limited. samtools manual page. 2018. <http://www.htslib.org/doc/samtools.html>. Accessed 29 Apr 2018.
3. The Bioconda Team. Using Bioconda — Bioconda documentation. 2016. <https://bioconda.github.io/>. Accessed 28 Apr 2018.

CRAMER: A lightweight, highly customisable web-based genome browser supporting multiple visualisation instances

M. Anastasiadi¹, E. Bragin², P. Biojoux¹, A. Ahamed¹, J. Burgin¹, K. de Castro Cogle¹, S. Llaneza-Lago¹, R. Muvunyi¹, M. Scislak¹, I. Aktan¹, C. Molitor¹, T. Kurowski¹, F. Mohareb^{1*}

¹ School of Water, Energy and Environment, Cranfield University, College Road, Cranfield, UK, MK43 0AL

² Next Gen Diagnostics, The BioData Innovation Centre, Wellcome Trust Genome Campus, Hinxton, UK, CB10 1SA

*To whom correspondence should be addressed.

**Supplementary Materials S2:
CRAMER Technical Documentation**

Table of Contents

| | |
|--|----|
| 1. Summary of the Document | 1 |
| 2. Application process | 1 |
| 2.1. Description | 1 |
| 2.2 Structure | 1 |
| 2.3. Workflow | 2 |
| 3. Client | 3 |
| 3.1. Architecture | 3 |
| 3.2. Structure of a CRAMER Instance | 4 |
| 3.3. Information and Warnings | 5 |
| 4. Server | 6 |
| 4.1. Architecture | 6 |
| 4.2. Node.js | 6 |
| 4.3. GET and POST | 6 |
| 4.4. File Parser | 7 |
| 4.4.1. Drag and drop | 7 |
| 4.4.2. Command line | 8 |
| 4.5. Database | 8 |
| 5. Sequence diagrams | 11 |
| 5.1. Index | 11 |
| 5.2. Delete | 12 |
| 5.3. Modify | 12 |
| Appendix | 15 |
| Files Structure | 15 |
| Files Summary | 16 |
| Main Folders | 16 |
| Within <i>public</i> | 16 |
| Within JavaScript | 16 |
| Within <i>Track</i> | 16 |
| Within <i>Model</i> | 17 |
| Within <i>View</i> | 17 |
| Within <i>library</i> | 17 |
| References | 17 |

1. Summary of the Document

This document is the official technical documentation of the application Cranfield Genome Browser (CRAMER). It is divided into six parts:

- The technical documentation of the operation;
- The technical documentation of the client;
- The technical documentation of the server;
- The sequence diagrams;
- The appendix;
- The references.

2. Application process

2.1. Description

The application tool is coded in JavaScript and executed using asynchronous scripting of Node.js server. The Node.js server has libraries called modules whereby each are implemented into a JSON package to perform different tasks during launching. The application launch location is on local available ports and connects through API addresses. The application uses MongoDB database and the server to communicate by making HTTP, URL and AJAX requests. The application architecture built supports a DOM format whereby the application process extends into a creation of webpage series. They are subdivided into five pages that include six different layouts. Each page works independently as a stand-in tool and works in synergy with other pages to increase browser performance. The following are the six different layouts:

- Welcome: Display the list of the instances available from the Database and allow the system administrator to access the login page and the CRAMER main page;
- Welcome (admin mode): The same page as the welcome page but with the possibility to modify or create new instance;
- Login: Allows the system administrator (referred to as the “user” throughout the rest of this document) to connect by using an approved username and password. It should be noted that the application does not include a user registration page. Our intention is not to provide the system users with means to register and add their own instances; instead, this functionality is reserved to the genome browser administrator (or group of administrators) to manage, on the behalf of research group. The reasoning behind this is to control the size of the database and storage space on the deployment server, by preventing the users of uploading duplicating genome files and/or large bam files which may result in running out of disk space quite quickly with genomic data.
- New Instance: Allows the user to create an instance with all the parameters (tracks, plugins, genome and information) and saves it into the database;
- Modify: Allows the user to modify an instance with all the parameters and saves it into the database;
- CRAMER (index): Displays the genome browser.

The input of the application consists of a web form. The output of the application is a customised instance of CRAMER.

2.2 Structure

The project is structured with three common components for a web application: a database, a server and a client. The server chosen is Node.js and given that, it accommodates the database is MongoDB.

Fig. 1. represents a basic illustration of how the three elements are working together.

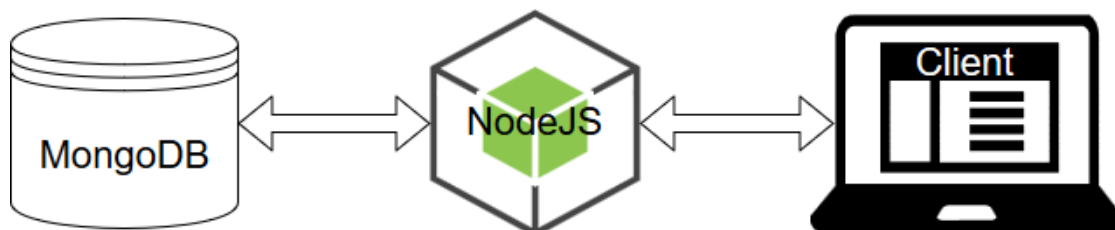


Fig.1. Application Structure

Fig. 2. (below) represents how Node.js works with the client. Contrary to a common server, it executes JavaScript on the client side to generate HTML pages which avoids using Object Oriented language like PHP, Java EE.

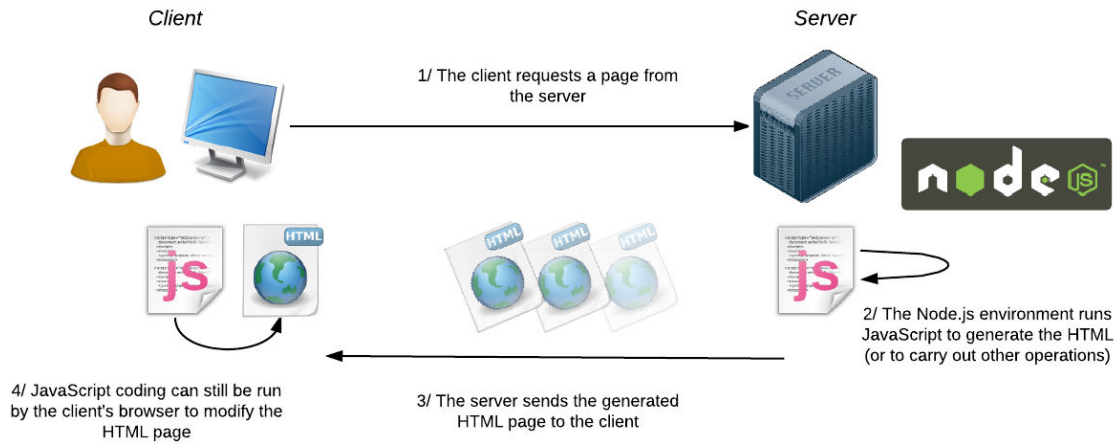


Fig.2. NodeJS schema. Source [1]

2.3. Workflow

Fig. 3. (below) explains how the application interacts with the different pages, the database and the user's decisions. The start-point is the welcome page. The yellow squares represent the different web pages and the red diamonds represents decisions. Then, the blue arrows represent all the interactions with the database. The grey arrows illustrate the results of the decision and the black arrows with the label represent what the user selects from the page.

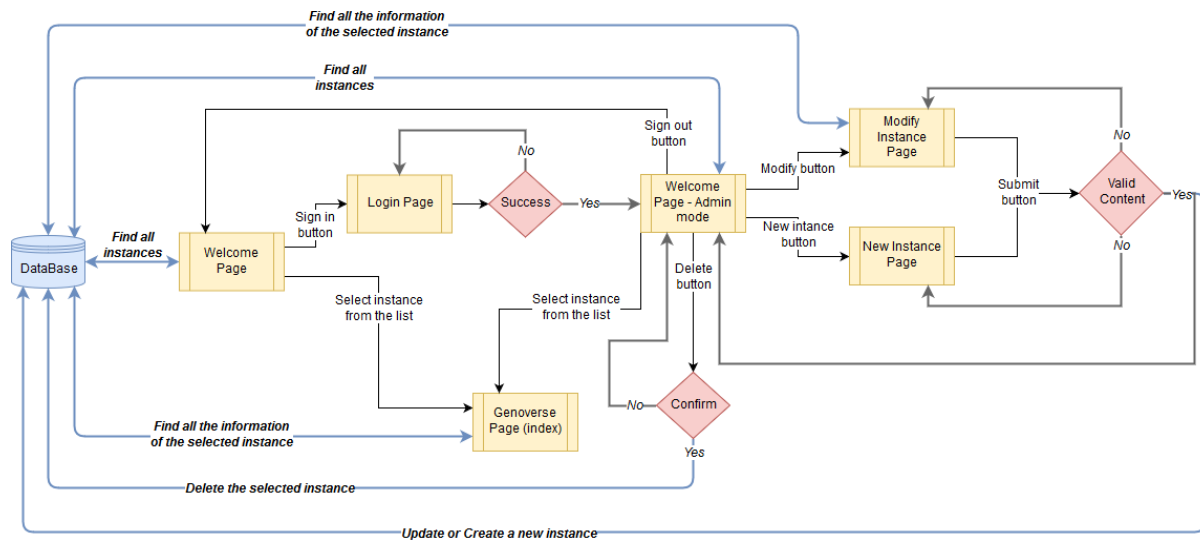


Fig.3. Application workflow

3. Client

3.1. Architecture



Fig.4. Stylesheets and JavaScript files included in Jade

bootstrap.min.css – link to set layout arrangements of the page elements;

css/style.css – link to set style of font sizes, line heights, font colours, background and foreground colours etc;

loginpage.css – link to set layout properties of login page;

bootstrap.min.js – contains JavaScript code for Bootstrap components;

jquery.js – refers script to jQuery libraries to facilitate browser manipulation, event handling, and making AJAX requests to the server;

configGenoverse.js – refers to a file script with all the configuration properties to launch the application for every new instance;

getDataForConfig.js – refers to a file script with all the functions to load tracks, remove tracks, add tracks, check all inputs for all pop-up track menus and the instance menu, controls the upload event of genome files;

genoverse.min.js – compressed version of all the JavaScript files for the Genoverse visualisation library used by CRAMER where unnecessary characters are removed in order to preserve bandwidth. Deleting characters such as a whitespace, new lines and comments allows the file becomes more compact. They are not required for source code to execute, thus both versions possess the same functionality. However, optimizing the file size increases the speed of the application.

genoverse.min.js is created with two files: **list-js.js** and **webpack.config.js**. The first one is used as an input file to manage all the files which need to be integrated into genoverse.min.js by containing their paths. The second one uses a node module called 'webpack' to write the content of genoverse.min.js according to the list of all JavaScript files in list-js.js.

The process is launched with the command line: `'npm run-script build'` and overwrites `genoverse.min.js` file.

3.2. Structure of a CRAMER Instance

Contrary to the Node.js server, this application uses an object-oriented programming architecture, using the **Base.js** library. This means classes extend from **Base.js** in typical object-oriented hierarchy fashion; inheriting all its functions and properties. These classes can then be extended from to expand upon their functionality and create more specific examples.

As a genome browser, this application has 2 main objects: the main panel and the tracks contained. The main panel object, CRAMER, is constructed in **Genoverse.js** and an instance of CRAMER is created each time the user wants to view an instance from the welcome page using **configGenoverse.js**. The Track object is initially defined in **Track.js** but is extended upon in the library folder where tracks are extended and configured for different datatypes and displays. Each time the user creates or modifies an instance of CRAMER, there is the option to create different track objects and customise them as required.

In the construction of each track, there are 3 more objects required: the model to condense the dataset into 'features' to be displayed, the view to display the 'features', and the controller to set how the user interacts with the track. These are first defined in **Model.js**, **View.js** and **Controller.js**. These classes are expanded on within the model, controller and view files, inheriting the abstract classes' properties but allowing them to be tailored to more specific examples (Fig 5 below).

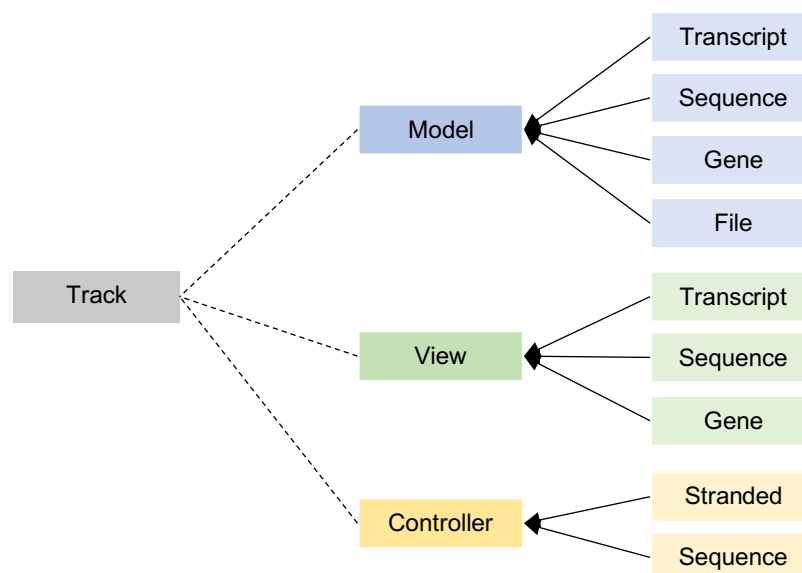


Fig. 5. Track component object inheritance. Black arrows show direction of inheritance, dotted lines show the components.

Within an instance of CRAMER, there are plugins set up for the user to interact with the view. Fig. 6. (below) shows the hierarchy of these plugins and how they rely on each other when constructed within CRAMER.

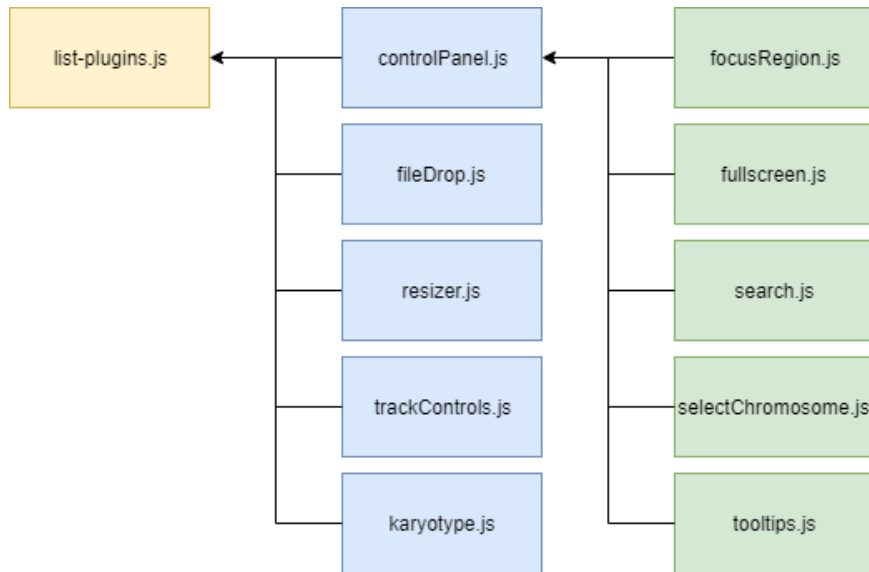


Fig. 6. Hierarchy of plugins. Green boxes show classes that rely on construction of the controlPanel.js plugin. Blue plugins are stand-alone. All plugins are stored within list-plugins.js to be called when a user edits or creates an instance.

3.3. Information and Warnings

To increase the proficiency and efficiency of the application, precautionary actions were taken by including alerts and error messages for user's discretion. The instance management process is only available to logged-in users, therefore the program has been improved with security features. The restrictions imposed verify whether the user is in the database. This is possible by use of the node package module called 'Passport'. Adding new users to the database is only possible by the website administrator, which increases browser security. The user database stores information about the e-mail address and the encrypted password for each one. With the correct input of data on the login page, the user is redirected to the logged-in version of the welcome page. In the case of one of them is incorrect, an alert with an appropriate message is displayed.

When creating or modifying instances and creating new tracks, the user is asked to enter the required information about them. Incorrect completion of one field causes the display of alerts. For this purpose, **alert.jade** file is used, which consists of a system that returns alert messages when the user fills in information incorrectly.

In cases of system errors, not dependent on the user's actions, a redirection is followed to an error page. This error page uses **error.jade** file to display details about the error. The user has the ability to redirect to the home page.

Additionally, information for the user is available by the plugin 'tooltips' which displays information on each tool element in the application. The user can also hover over track names to display the description set.

4. Server

4.1. Architecture

The server is built with Node.js and more particularly with a node module called 'Express', which is explained in more detail in the next section. The structure of the project was made with the express-generator module with the command line "\$ express generate". With this generator, the application creates this following:

```
|— app.js
|— bin
    └─ www
|— package.json
|— public
|— routes
└─ views
```

Table 1. Description Server Structure

| File/Folder | Description |
|--------------|--|
| app.js | Express application where the parameters of the applications are set such the routes, the view engine, the body parser, the cookies, ... |
| bin/www | Server file where the port is set, the listener to handle all the requests and errors |
| package.json | Contains list modules |
| public | Contains the stylesheets, the images and the JavaScript files for the client side |
| routes | Contains all the routes |
| views | Contains all the views in JADE language |

For more details of the overall structure of the project refer to the [appendix](#).

4.2. Node.js

Well known to many developers, Node.js is a platform, based on the Chrome JavaScript runtime, which is able to quickly develop highly scalable applications. The use and installation of the various modules is greatly simplified using the command "\$ npm install" (see the [instructions](#)). This enriches its JavaScript code.

This application uses modules such as:

- Express: Framework for building web applications faster and easier;
- Mongoose: Gateway between Node.js server and MongoDB server;
- Async: Functions to work with asynchronous JavaScript;
- Passport: Manage authentication from account creation to login. It is preconfigured to work with MongoDB;
- Jade: Template engine for NodeJS;
- Webpack: Module to minify the code.

The list of all the modules used are in the file **package.json**. This file is used to list which modules to install. It also contains the information for the version, the homepage, the authors and the build and start.

4.3. GET and POST

In order to communicate, the server and the client send POST and GET methods. These methods allow data requests for GET and data submissions for POST through a URL. Table 2 shows all the GET and POST requests in the application with the description.

Table. 2. GET and POST requests within CRAMER

| URL | Methods | Description |
|--|---------|---|
| / | GET | Send the welcome page generated |
| /login | GET | Send the login page generated |
| /login | POST | Get the credentials |
| /logout | GET | Send the welcome page generated after logout |
| /delete/:name | GET | Send the updated welcome page generated after deleting the instance |
| /instance | GET | Send the instance page generated |
| /instance | POST | Get the instance information (JSON Object) to save it in the DB |
| /modify | GET | Send the instance page generated with the instance data from the database |
| /modify | POST | Get the instance information (JSON Object) to save it in the DB |
| /index?name=[instance-name] | GET | Send the index page generated with the instance data from the database |
| /index/request?chr=[num]&start=[start]&end=[end]&type=[type]&file=[file] | GET | Send the data (URL parameters) from the ftp |

Fig. 7. (below) is a detailed example of a GET method to request data in order to display a FASTA sequence track:

<http://138.250.31.99/index/request?chr=1&start=1&end=100000&type=faidx&file=ftp://138.250.31.77/Public/Sequence.fa>

[This sets the parameters below:](#)

| Parameters | Signification |
|--|---------------------|
| chr=1 | The chromosome |
| start=1 | The start position |
| end=100000 | The end position |
| type=faidx | The tool to execute |
| file=ftp://0.0.0.1/Public/Sequence.fa | The FTP address |

Then on the server side, with all this information the application knows what to execute and the command line looks like this:

```
$ /usr/bin/samtools faidx ftp://0.0.0.1/Public/Sequence.fa chr1:1-100000 | tail -n+2
```

The output is returned like this:

```
child.stdout.pipe(res);
```

And if there is no error in the file, CRAMER parses the returned sequence.

Fig. 7. A detailed example of a GET method to display a FASTA track

4.4. File Parser

Two solutions are available to display tracks:

- Drag and drop a local file into the browser when the CRAMER page is opened;
- Add a FTP file when an instance is created or modified.

4.4.1. Drag and drop

'fileDrop' is a plugin within CRAMER that allows the user to drag and drop local files within the track display to visualise them.

Default tracks for each file type are saved within the 'File' folder in the track library. These are generated depending on the suffix of the file. For example, if the user drops a GFF file, subsequently the file object generated from the drag and drop event is read by the GFF track model and a GFF track is added to the display. If there is an index file also present, it is recognised by the defined suffix within the track. For instance, a VCF track has its index suffix listed as '.tbi' and in this case it would be read with a library (dalliance).

4.4.2. Command line

To avoid downloading the whole file with each client request (The file can either be stored locally on the application server, or stored elsewhere and made available to the application via an FTP link, the application uses bioinformatics tools (Samtools, Bwtool, ...) which extract only the region needed. This makes the application faster and prevents the application from reading the whole file as many genomic files are very large.

To use these tools, the application executes command lines in the router **index.js** with a node module called 'child_process'. To extract a specific region, an index file must be provided in the current directory and it will be downloaded into the indexes folder in the application, except for BigWig where the library does not require an index.

The typical command line is '\$ **[tool] [option] [file] [chromosomeHeader]:[start]-[end]**'.

Table 3. Command line tools used to read ftp file inputs

| Parameters | Description |
|------------------|--|
| tool | The server path for the tool |
| option | The option for the tool |
| file | FTP address |
| chromosomeHeader | The header with the chromosome number |
| start | The beginning of the sequence to extract |
| end | The end of the sequence to extract |

However, these tools have some requirements in order to be executed correctly:

- The index file must have the same name as the input file with the correct index extension such as this:
 - File input: *ftp://IP.address/name.of.the.file.gff*
 - Index file: *ftp://IP.address/name.of.the.file.gff.tbi*
- The headers in the file need to be sorted and the chromosome number must be at the end. The function "extractHead()" in **utils.js** calls for the headers in the file and selects the correct one according to the chromosome number/name at the end. Here are some examples of headers in compatible files:
 - SL2.50chr01, SL2.50chr02, ..., SL2.50chr12;
 - chr1, chr2, ..., chrY, chrMT;
 - 1, 2, ..., MT.

Note that this function does not work for a FASTA input, here, the header needs to be formatted as **chr[num]**. If all these requirements are respected, the tools will successfully extract the data.

Table 4. (below) illustrates the requirements per tool.

Table 4. The requirements for each file input to run correctly with the command line tool set

| Tools | Files | Requirements |
|--------------------|---------------|-----------------------------------|
| Samtools fadix | FASTA | Index file and header like 'chr1' |
| Samtools view | BAM | Index file |
| Tabix (Samtools) | BED, VCF, GFF | Index file |
| Bwtool, bigWigInfo | BIGWIG | Index file |
| wget | RSEM | Extension file .results |

4.5. Database

The application is supported by a MongoDB database backend, which contains two models which are **User** and **GenoveseInstance**. The first one is a simple collection used solely for admin login and authentication (Table 5). To manage the MongoDB, Node.js uses a module called 'Mongoose'.

Table 5. User table in MongoDB database

| User |
|------|
|------|

| |
|--|
| name: {type: String, required: true} |
| mail: {type: String, required: true, index: {unique: true}} |
| password: {type: String, required: true} |

The second one is more complex and includes Subdocuments which are “genomeSchema”, “pluginSchema”, and “trackSchema” (Table 6, 7, and 8 respectively).

Table 6. GenomeSchema table in MongoDB database

| GenomeSchema | |
|---|--|
| name: {type: String, required: true} | The name of the genome |
| type: {type: String, required: true} | Indicate if the genome is uploaded from a file or from Ensembl |

Table 7. PluginSchema table in MongoDB database

| PluginSchema | |
|---|--|
| "name": {type: String, required: true} | The name of the plugin |
| "checked": Boolean | Indicate if the instance has to display this plugin |
| "id": {type: String, required: true} | ID of the plugin |
| "info": {type: String, required: true} | Description for the hover label in modify and instance pages |

Table 8. TrackSchema table in MongoDB database

| TrackSchema | |
|--|--|
| group: {type: String, required: true} | The name of the group (BED, Fasta sequence, ...) |
| checked: Boolean | Indicate if the track has to display this plugin |
| trackChildren: [track] | List of the tracks in this group |

“trackSchema” also includes a subdocument “track” because it is possible to have more than one instance of each track type with different file inputs in a CRAMER instance. This way, all the different tracks of the same type are saved in one group and if “checked” is true the application receives the list of trackChildren to display (Table 9).

Table 9. Track table in MongoDB database

| Track | |
|------------------------------------|--|
| name: {type: String} | The name to display in CRAMER |
| description: {type: String} | The description for the hover label |
| data: {type: String} | Contain the configuration of the track |

Finally, the result of GenoveseInstance is contained in the “GenoveseSchema” (Table 10. and Fig. 8.).

Table 10. GenoveseSchema table in MongoDB database

| GenoveseSchema | |
|--|---|
| name: {type: String} | The name to display on the welcome page |
| description: {type: String, required: true} | The description on the welcome page |
| genome: genomeSchema | Contain the genome information |
| chr: {type: String, required: true} | The chromosome to display |
| start: {type: String, required: true} | The start position to display |
| end: {type: String, required: true} | The end position to display |
| plugins: [pluginSchema] | List of plugins |
| tracks: [trackSchema] | List of tracks |

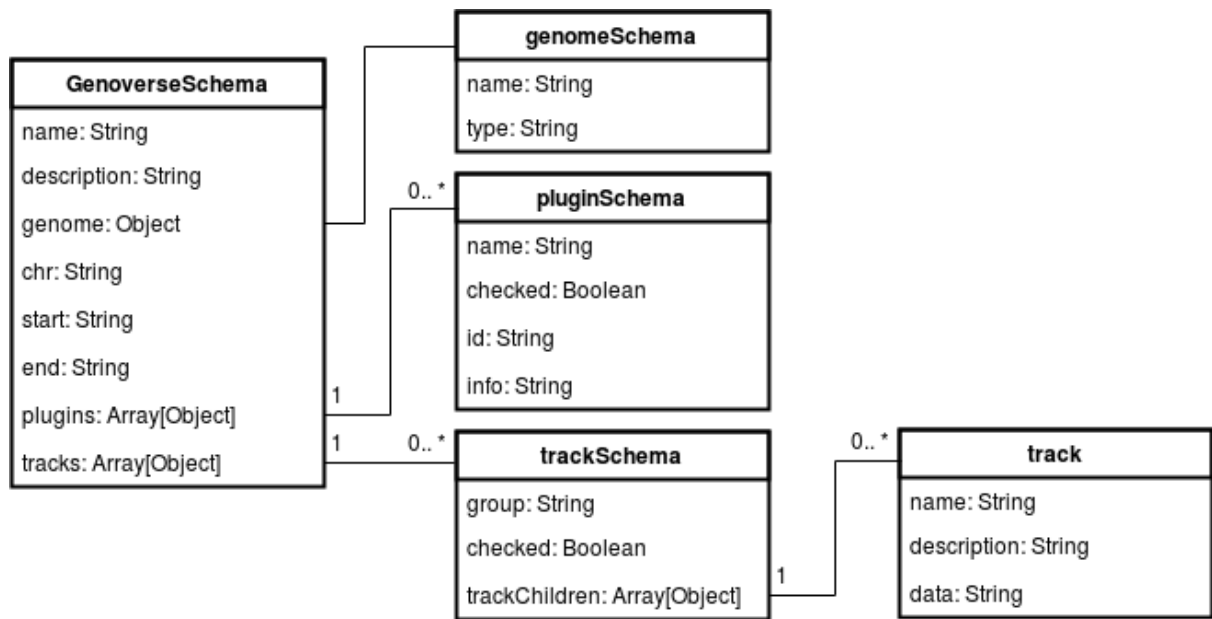


Fig. 8. UML diagram of CRAMER instance schema in the database

5. Sequence diagrams

5.1. Index

Fig. 9. (below) is the sequence diagram with a GET method to extract the data for a specific region using a GFF file as the example. This GET method executes a command line and this sequence diagram illustrates all the processes involved.

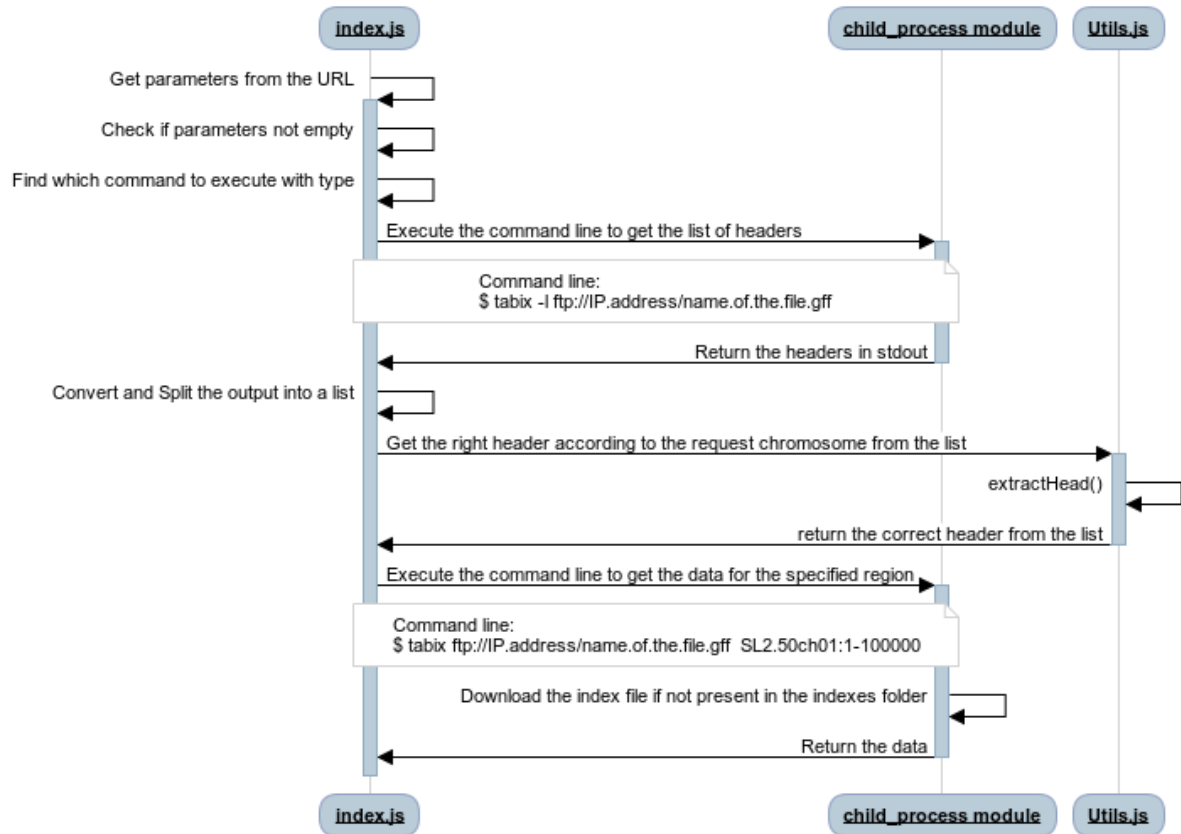


Fig. 9. Sequence diagram of index.js

5.2. Delete

Fig. 10. is the sequence diagram when the user clicks on the delete button. This action executes two GET methods:

- To delete the instance;
- To redirect to the welcome page after the instance deleted.

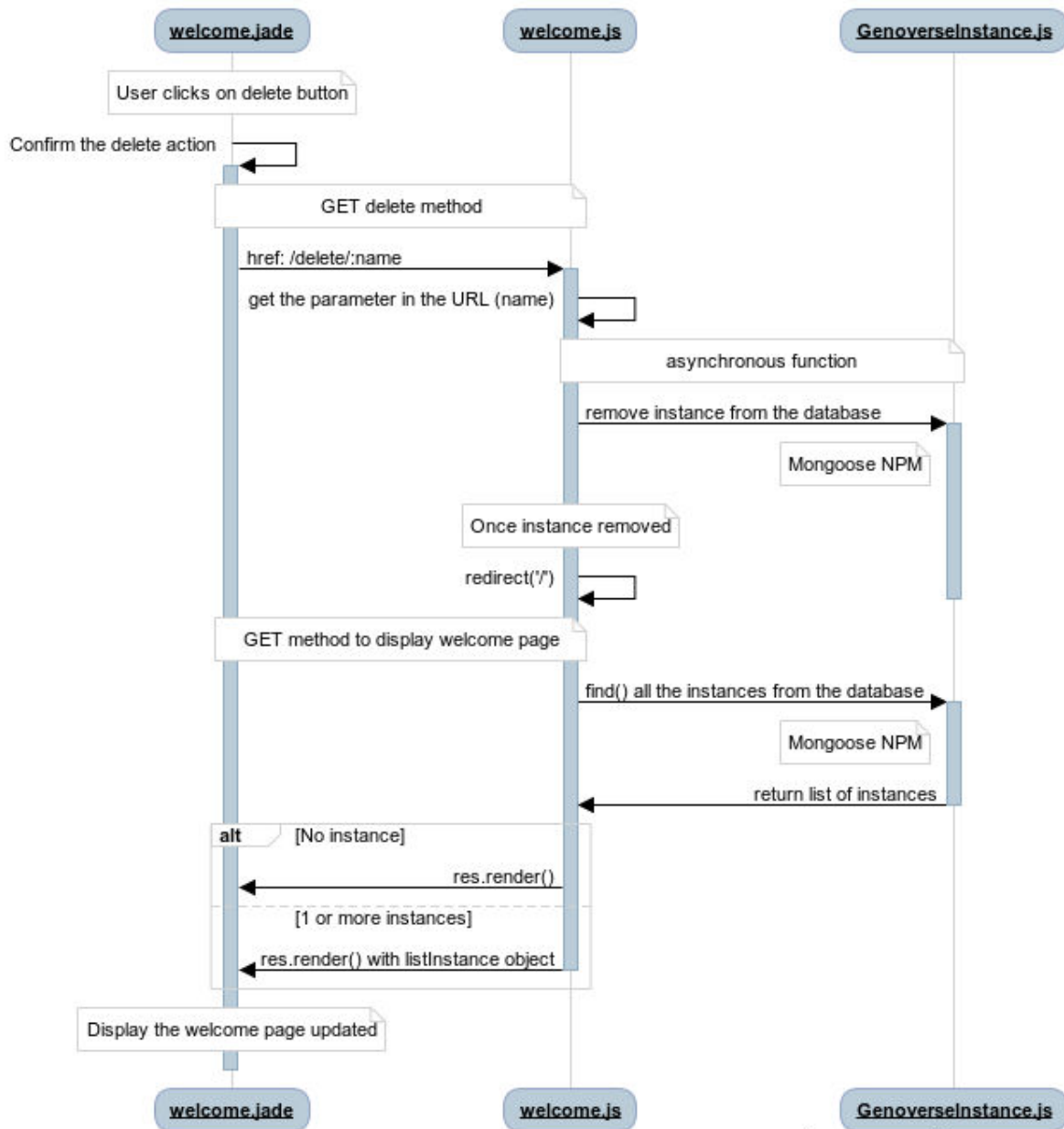


Fig. 10. Sequence diagram of delete button

5.3. Modify

These are sequence diagrams of the modify route. These illustrate the connection between the client and the server. The instance route works in the same way except there is no need to fetch the instance information from the database when the page is opened.

Fig. 11. shows the GET method which is called when the user clicks on the button “modify” on the welcome page. The result of this request is the display of the modify page.

Modify Post Method

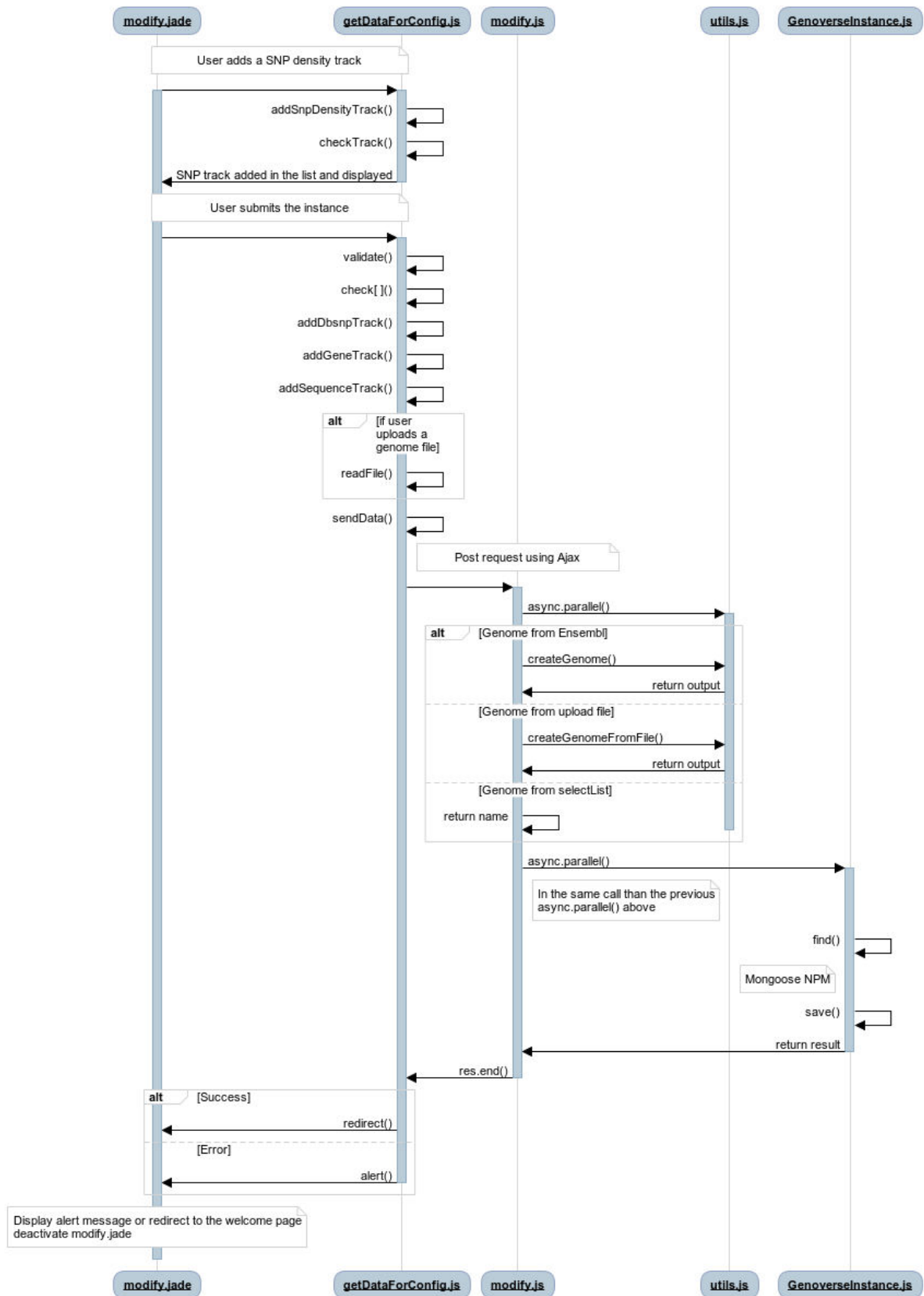
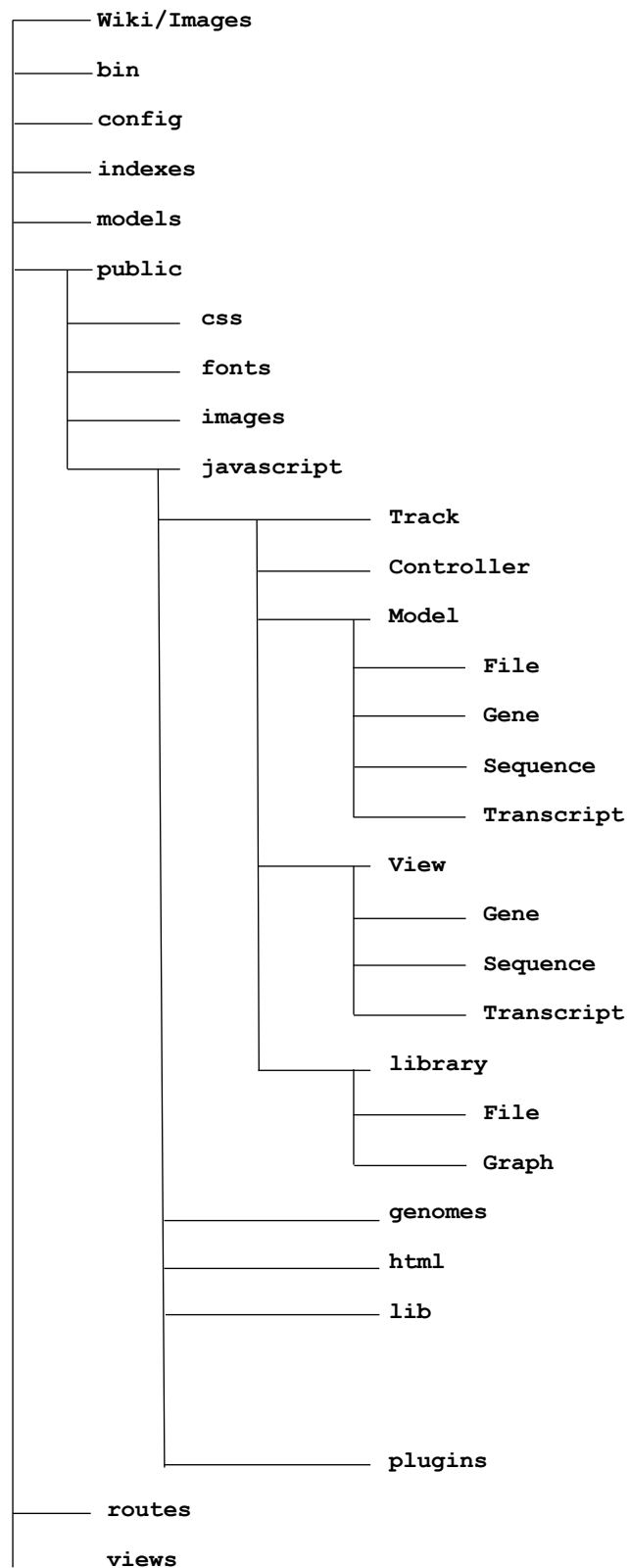


Fig. 12. Sequence diagram of post method for modify.js

Appendix

Files Structure



Files Summary

Main Folders

Wiki/Images - contains the images used in the wiki.

bin - contains the script to deploy the application onto the Node.js server.

config - contains the scripts for interacting with the database and admin identification.

indexes - contains all the index files downloaded from the ftp files when the user sets up an instance. It ensures the ftp files are only read in the region that is viewed using command line tools.

models – contains the database models.

public - contains all scripts from the publicly available CRAMER project with additional scripts made by the CRAMER development team to improve customisability with new models and tracks. This folder contains any JavaScript used to define a CRAMER instance or the tracks within it.

routes – contains all the server-side scripts for the Jade webpages.

views – contains all the Jade webpages.

Within *public*

css – contains all the css style files.

fonts – contains all the program specific fonts.

images – contains any images used in the program.

javascript – contains all the JavaScript to create an instance of CRAMER.

Within JavaScript

Track – contains all the JavaScript for the three objects required in creating a track– the model, view and controller as well as the library of ready set up tracks where all 3 are ready defined.

genomes – contains all genome files either uploaded by the user or generated from a species chosen from Ensembl.

html – contains the JavaScript that works with the **instance.jade** and **modify.jade** in creating the instance configuration.

lib – contains all the external libraries integrated in the project such as jQuery and dallianceLib.

plugins – contains all the plugins used for user interaction with a CRAMER instance including the main control panel, karyotype and the new search and selectChromosome plugins.

Within *Track*

Controller – contains all JavaScript that extends from the basic controller class and allows the user to interact with the tracks.

Model – contains all JavaScript that extends from the basic model class and allows specific examples with multiple datatypes for different track configurations.

View – contains all the JavaScript that extends from the basic view class and allows specific examples of display types for different track configurations.

library – contains a library of all complete tracks, combining suitable models, views and controllers. These track classes are what are later extended upon for customisation by the user in **instance.jade** or **modify.jade**.

Within Model

File – contains all the JavaScript models that reads data from dragged and dropped files or ftp file links.

Gene – contains all JavaScript that extends from the abstract gene model, this is mainly used for Ensembl tracks.

Sequence – contains all JavaScript that extend from the sequence model, this is used for Ensembl based examples as well as reading FASTA files.

Transcript – contains all the JavaScript that extends from the abstract transcript model, this is mainly used for Ensembl tracks.

Within View

Gene – contains all JavaScript that extends from the abstract gene view script. It is mainly used for Ensembl based tracks but can be expanded on for custom tracks.

Sequence – contains all JavaScript view scripts that extends from the sequence view script. For example, sequence variation which has not been expanded on further in this application but is available for custom tracks.

Transcript – contains all JavaScript that extends from the abstract transcript view script. It is mainly used for Ensembl based tracks but can be expanded on for custom tracks.

Within library

File – contains a library of all track classes that are based on drag and drop file inputs or ftp file links. Their models and views are defined within these classes.

Graph – contains all JavaScript that extends from the graph track such as bar or line options. Within these, the controller, model and view are defined in detail. Other tracks use these classes as a basic for their view and model if the output is wished to be displayed as a graph, such as the new SNP density track and Gene Expression track.

References

1. Nebra M. Node.js: what is it for exactly? <https://openclassrooms.com/courses/ultra-fast-applications-using-node-js/node-js-what-is-it-for-exactly>. Accessed 6 Mar 2018.