Parkour Documentation

MPI

Sep 17, 2020

Contents

CHAPTER 1

Contents

1.1 User Manual

1.1.1 Introduction

Parkour LIMS supports sample processing laboratories, dealing with thousands of samples per year, with laboratory management, sample documentation, tracking, and evaluation. The LIMS has a web-based interface to be accessed by different user groups, that'll profit from the software's functionality. Any *active user* (see 2.2 permissions) can access the software, create new requests for Next Generation Sequencing (NGS), follow in real-time the status of the request over the different stages of the workflow, store any request related data for documentation, upload request related metadata to public sequencing data archives. User with *staff* permission and higher (see 2.2 permissions) can access any part of the software to edit and process requests and samples through the different stages of the workflow including *Request overview - Incoming Libraries and Samples - Index Generator - Preparation - Pooling - Load Flowcells - Invoicing - Usage - Statistics*. Access to the administrators area allows dynamic content adjustment as well as user management without any software programming knowledge.

The LIMS's general principles are a simple yet functional web interface, ease of use, user input validation, and fast data processing. Interestingly for software developers, Parkour's dynamic structure allows for any program adjustments or implementation of new features into the existing workflow. Furthermore the program can be extended to support other than NGS workflows. Basically any samples processing laboratory can profits from the usage of Parkour as central laboratory management platform.

Table 1: Sections of Parkour LIMS available for users with permissions "Active" and "Staff"

	Active	Staff
Requests	X	X
Libraries & Samples	X	X
Incoming Libraries/Samples		X
Index Generator		X
Preparation		X
Pooling		X
Load Flowcells		X
Invoicing		X
Usage		X
Statistics (Runs/Sequences)		X

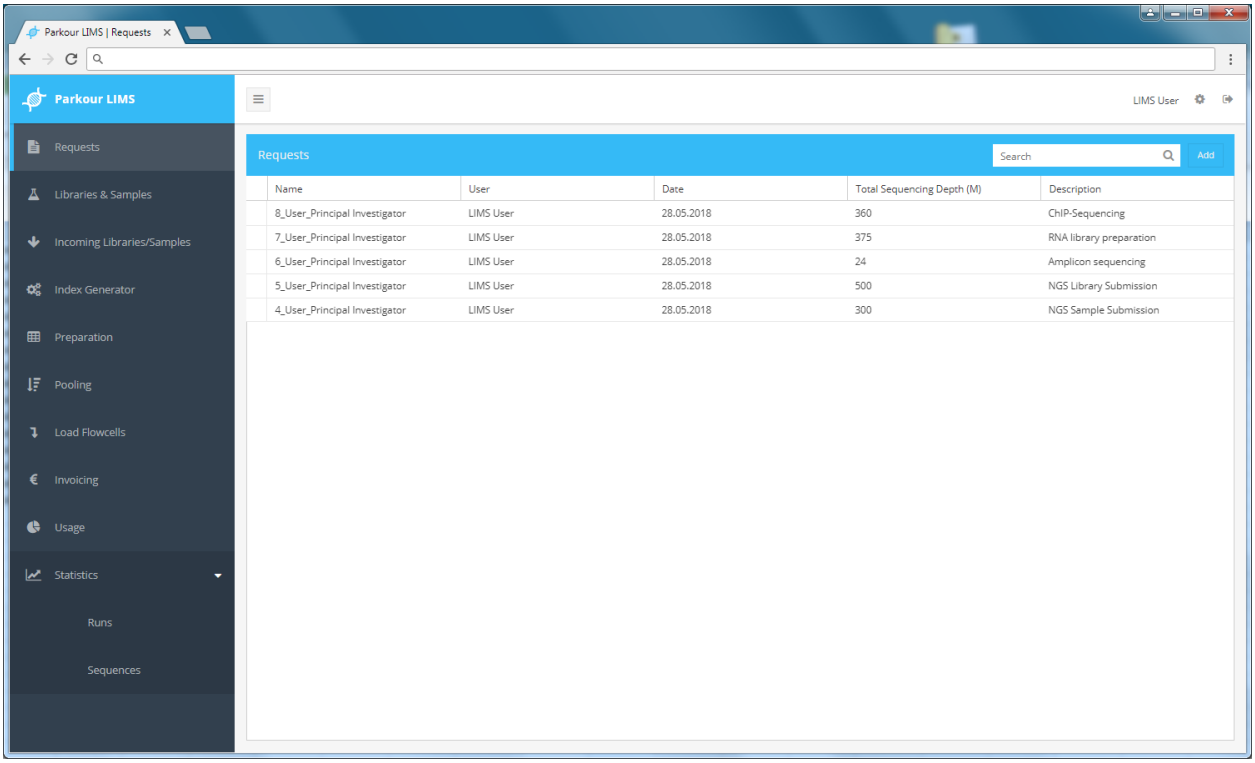

Fig. 1: Screenshot of Parkour after login for users with the permission *staff*

Note: Users with the permission *active* will see only **Requests** and **Libraries & Samples**.

1.1.2 Getting Started

Managing accounts

Parkour has a central user management to register new users, edit or delete existing users.

Only users with staff status can enter the administration site to add or edit new users and allocate permissions.

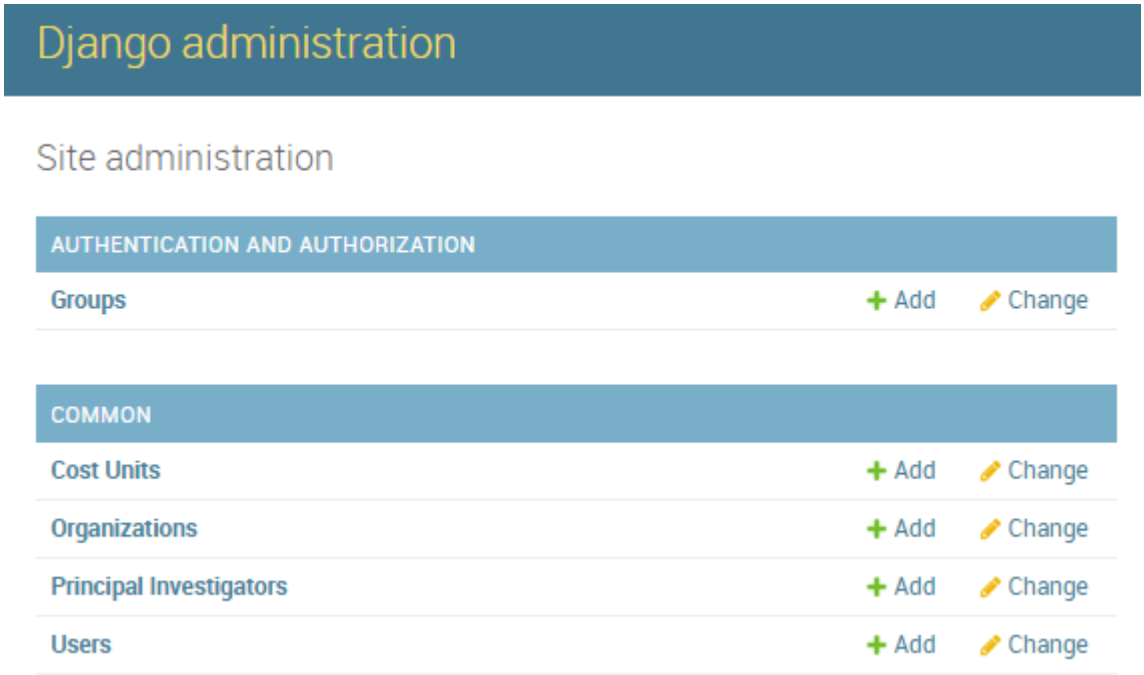

Fig. 2: Parkour site administration. Changing users.

To add a new user, open *Parkour* and go to site administration (/admin) and select "add" new user. Enter first name, last name and email address. A random password will be generated. Click "Save and continue editing". Specify phone number, organization, principal investigator and cost unit. Multiple Cost units can be assigned to a single user. Depending on the role of the user, set permissions for the user; choose between: Active, Staff, and Superuser. Click "Save" to save the changes to the database.

Permissions

A Parkour user can get assigned the following roles:

- **Active**. Users can create requests and view the status of their own requests. The user can attach files to a request. An active user will only see the request and the libraries and samples windows populated with data from own requests. All additional functions of Parkour are hidden for an active user.
- **Staff**. Users with the staff permission are typically any laboratory personal that is involved in request processing. Such users will see the complete Parkour software and all submitted requests and can edit requests at any time. Staff users can access administrator area to edit shared tables. It is possible to control/restrict access to selected parts of the administrators area. Note allocate "staff status" to a user select both, "activate" and "staff" in the administration site/permissions.
- **Superuser**. Users with the superuser permission have all the rights of staff users, editing rights for all shared tables in the system and the rights to add new users. No further restrictions can be set. Note to allocate "superuser status" to a user select all, "activate", staff" and "superuser" in the administration site/permissions.

Changing your password

Upon registration in Parkour, a link is sent to your email address and you are asked to change your password. If you forgot your password, click on "Forgot password" on the main login page and change your password accordingly.

Fig. 3: Parkour site administration. Add user.

Adjusting Parkour Content

To customize Parkour to individual needs a staff user and higher can access the site administration and edit shared tables. For instance, to add a new library preparation protocol or edit/delete an existing protocol, go to shared tables and choose Library Protocols. Choose "Add Library Protocol" to add a new protocol. Enter information into the requested fields and save the changes. To edit any other parameter i.e. invoicing, sequencers, index types etc. follow the same strategy.

Fig. 4: Parkour site administration. Shared tables.

1.1.3 Requests

This is the main page where you can see all of your requests and create new ones. The search bar allows you to quickly filter for requests by name, creation date and description. The search retrieves only exact matches to your query, so be

attentive to what you type. The search is not case sensitive. The result of this search will affect all data viewable (ex: if you search for a request named '1_User', the only requests that will be displayed in the requests view screen will be those that contain '1_User').

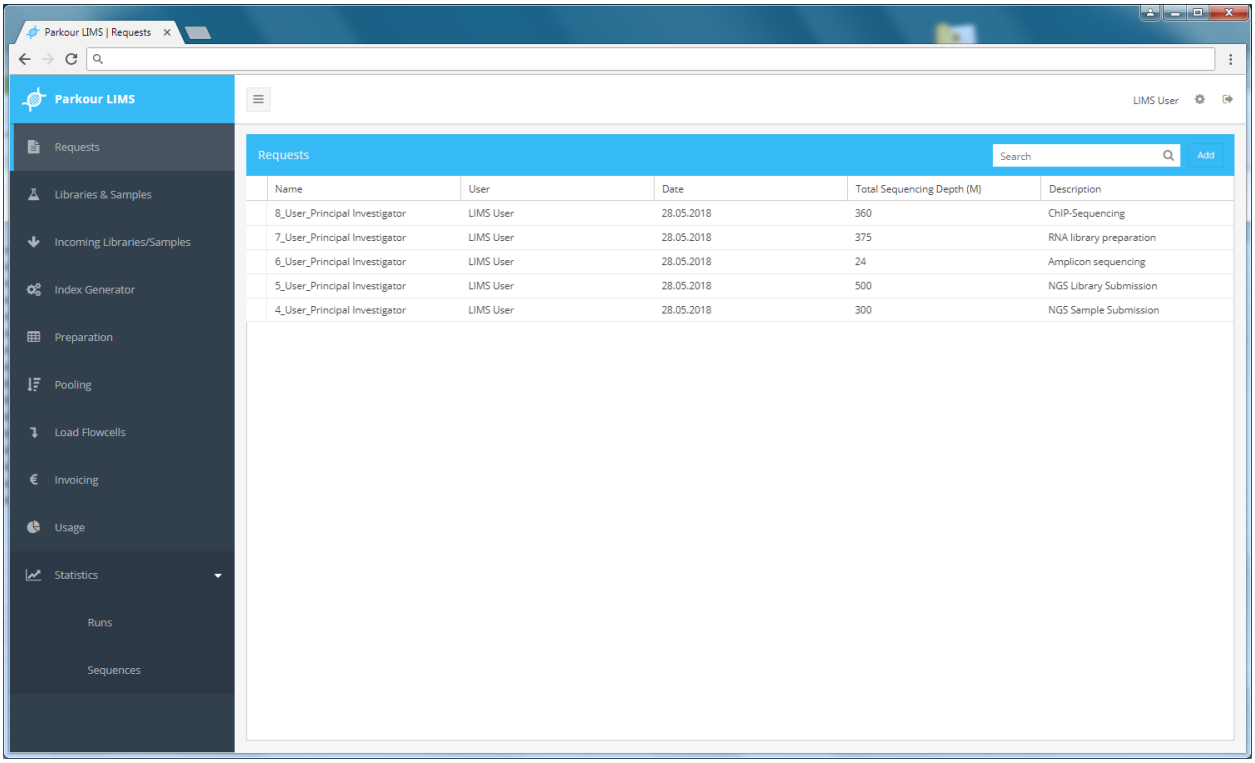

Fig. 5: Requests module.

Request creation

Login to Parkour, choose requests and select "Add" (top right corner in Requests tab) a dialogue "new request" will appear. Use this dialogue to enter a request description, add libraries or samples to the request, and attach any files if needed.

To start adding samples to a request, click the Add button in the bottom right corner of the "New Request" dialogue. A request can contain either libraries or samples. Choose "library" if samples for sequencing are already prepared by the user. Choose "sample" if libraries will be prepared by the sequencing facility.

Depending on the selection (library or sample) different parameters need to be filled into the webform.

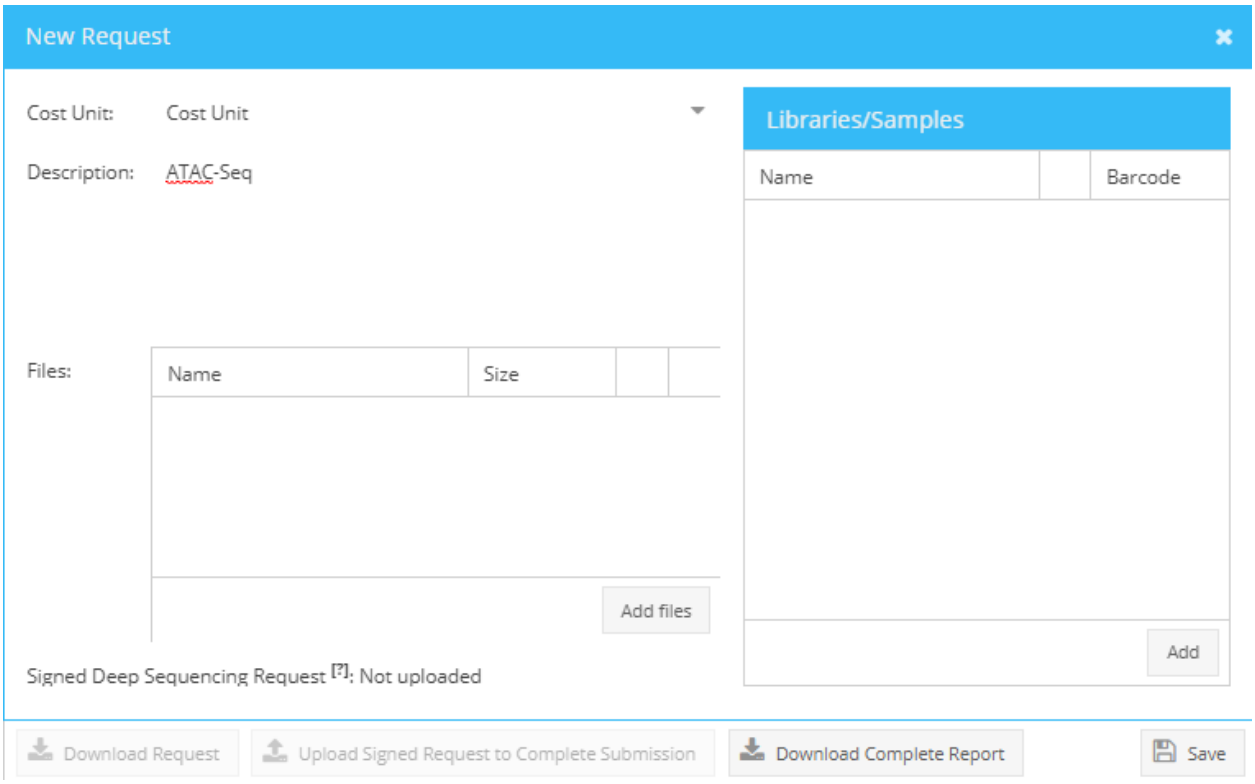

Fig. 6: Add new request.

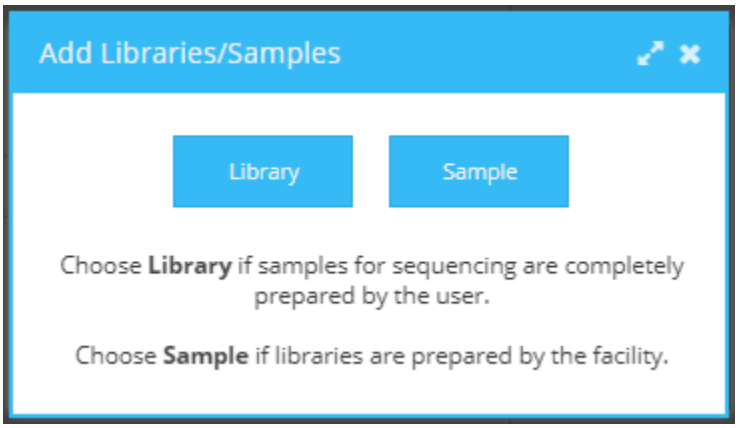

Fig. 7: Add library or sample.

Table 2: Request generation, editable parameters, marked fields are mandatory

Parameter	Explanation	Field type, Restrictions
Name*	Meaningful label for your sample	String, only A-Za-z0-9 as well as _ and- are allowed; no duplicate entries
Nucleic Acid Type*	Type of nucleic acid submitted for library preparation i.e. total RNA, genomic DNA.	Select from dropdown menu
Protocol*	Library preparation protocol used to prepare the sequencing libraries	Select from dropdown menu
Library Type*	Type of sequencing library generated i.e. RNA-Seq, ChIP-Seq.	Select from dropdown menu
Concentration (ng/µl)*	Measured concentration	Float
RNA quality (RQN)*	RQN or RIN (RNA Quality Number or RNA integrity number), determined by software of capillary electrophoresis device	Float in the range 1-10
Size (bp)*	Size distribution of submitted samples/libraries in bp (base pairs)	Integer
Index Type*	Predefined compilations of indices, differ per library preparation protocol	Select from dropdown menu
Index Reads*	Total number of index reads to be conducted by the sequencing instrument	0,1 or 2
Index I7*	Sequence ID and sequence of index I7 used to construct library	Select from dropdown menu
Index I5*	Sequence ID and sequence of index I5 used to construct library	Select from dropdown menu
Read Length*	Number of bases to be sequenced by the sequencing instrument	Select from dropdown menu
Sequencing Depth*	Number of reads that align to known reference bases	Integer
Amplification cycles	Number of PCR amplification cycles	Integer
Equal representation of nucleotides	Technical requirement for sequencing on Illumina sequencing instruments	Check = yes, no check = no
qPCR result	qPCR quantification result	Integer
Concentration determined by*	Specify concentration measurement concept i.e. fluorometry, spectrometry	Select from dropdown menu
Organism*	Origin of samples or libraries	Select from dropdown menu
Comment	Additional information on libraries or samples	String

The request creation table can be edited in multiple ways.

- **Copy and Paste**. For convenient batch editing (per column editing) select a cell, press Esc (cell turns yellow) and paste your data (ctrl + v).
- **Drop down lists, apply to all function**. Multiple cells provide drop down lists, indicated by little arrows. Choose the desired parameter from the list, press enter or hit the "update" button to fix the parameter into the cell. To assign the chosen parameter to all samples of a request mouse over the selected parameter, right click and choose "apply to all". To delete a parameter from a column, choose the respective cell, mouse over, right click and press "delete".

Note: Invalid cells are marked in **dark red**. If you hover on them, you will see a tooltip with a help message.

Most of the requested fields are mandatory. If all required fields are filled the colour of the request form turns from red to white. Click "Save" to save the metadata to the database. You will return to the "new request" dialogue.

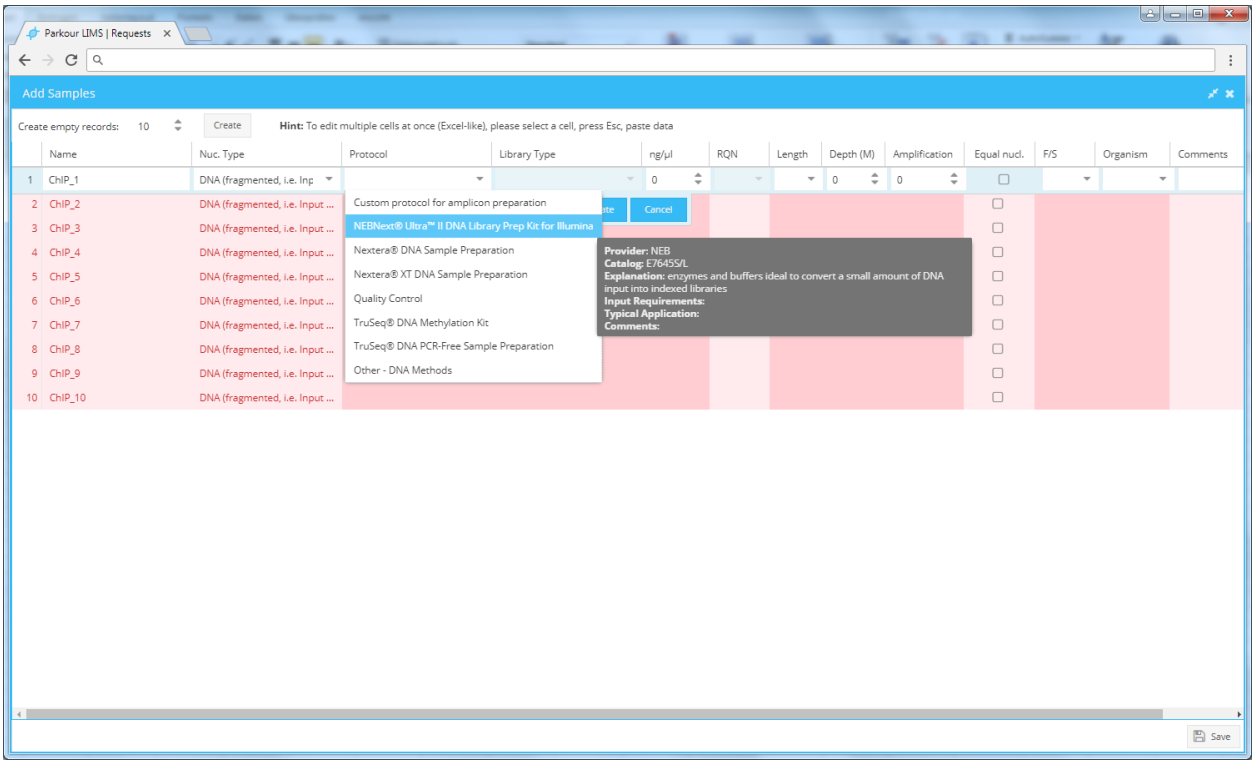

Fig. 8: Add new samples window.

Complete the request submission by adding the description of your experiment and choose a cost unit, press "Save". A request ID is allocated and request appears in the request window. The request ID is a a running number followed by the users last name and the last name of the principle investigator. All samples or libraries are assigned a barcode which is a combination of the year, the letter "S" or "L" to indicate either sample or library and a running number with 6 placeholders. If needed the counter can be set to "0" by the beginning of the year.

Attaching files to a request

Files can be attached to a request at any time. To attach a file, choose the tab Requests and right click on the request of choice to open the context menu, select "View". Use the "Add Files" button to browse the desired files. Click "Save" to attach the files to your request. In the Requests table a little paper clip icon indicates files attached to a request. Click on the paper clip icon to view and download attachments. Both, users as well as laboratory staff should use this central position to store any additional files i.e. raw data from sample quality assessment.

Request approval

A request, including all attached samples or libraries, will change its status from "Pending submission" to "Submission completed" only when the PI in charge has approved the request.

Once approved, a request will appear in the stage "Incoming libraries and samples" and further processing can start.

To approve a request, choose the tab “Requests” and select the pending request. Right click on the request to open the context menu and select “View”. Click the button “Download Request” (1) to generate the deep sequencing request form. Then, print the downloaded PDF file and ask your PI to sign it.

To upload the signed form, choose the pending request in the Requests tab, open context menu by right-clicking on the request, select “View” and click the “Upload signed request” button (2) to attach the approval to your request. The request status changes from “Pending submission” to “Submission completed” and request processing can start.

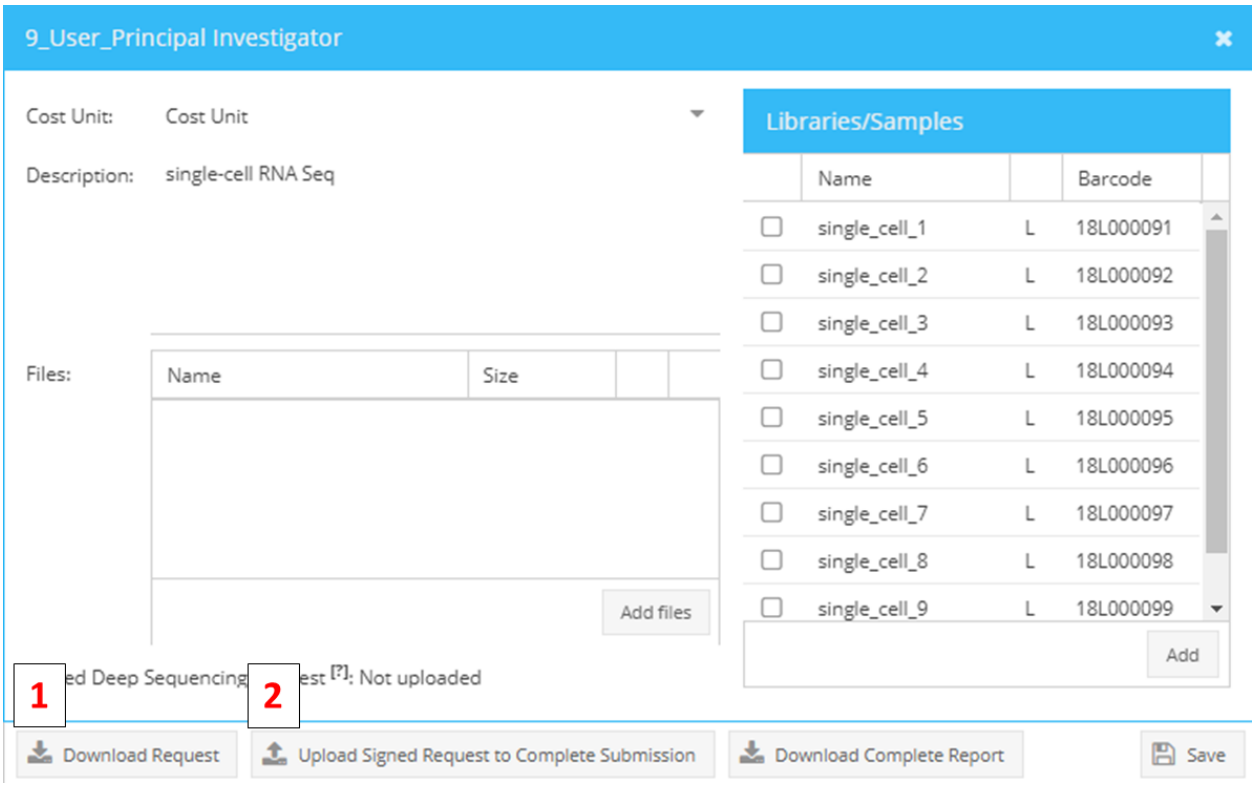

Fig. 9: Downloading (1) and uploading (2) of sequencing requests to approve samples for sequencing.

Request editing

Until request approval by a PI (submission completed) both, active and staff users, can edit samples and libraries in a request. When a request reaches status “submission completed” editing is reserved for staff users only.

To edit a request, select from the Requests tab the pending request (right click, view) Select from the top left corner of the appearing Libraries/Sample table (right side of the appearing dialogue) the command “select all libraries or samples” and start editing by clicking “edit all items”. A table including all requested libraries/samples opens and you can start editing. Press save to return to the main request window.

Request status

Once a stage in Parkour is completed, libraries and samples change status, indicated by changing colour. To view the status of individual samples and libraries, go to the “Libraries & Samples” tab and expand the request. In the column “status” a coloured dot will appear. Mouse over for further explanation. Note that a request can be composed of samples and libraries and samples/libraries can be at different stages in the workflow. Therefore, only a status per sample/library is shown and not an overall request status.

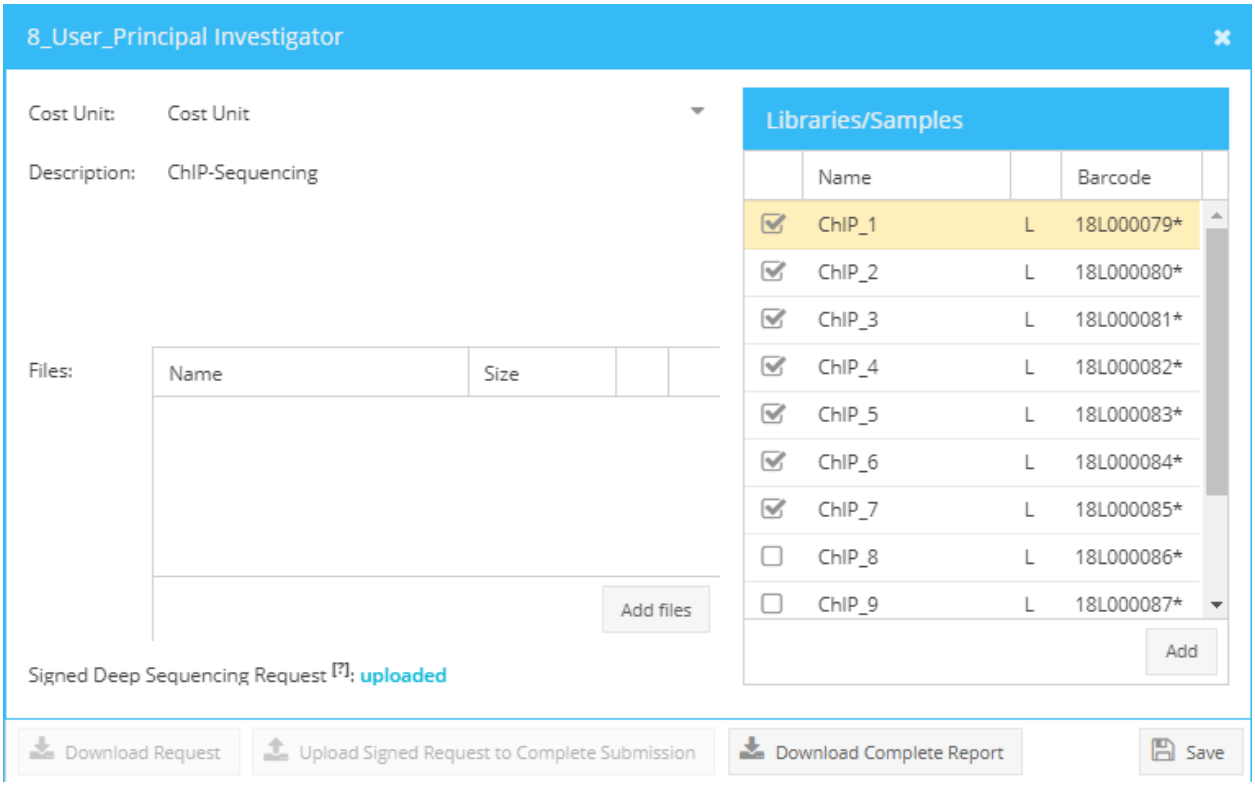

Fig. 10: Editing of samples/libraries.

1.1.4 Incoming Libraries and Samples

Once a request is approved by the respective PI, status of all samples or libraries changes to "submission completed" and samples/libraries are appearing in the "Incoming Libraries and Samples" window. At this stage only users with permission set to "staff" will see the requests and can start conducting incoming sample/library quality control.

During "incoming quality control" each request is evaluated by users with "staff" permission, typically staff from the sample processing laboratory. Each request must pass predefined quality criteria. Even though a user has specified all parameters (i.e. sample concentration and sample integrity), samples will be requalified by members of the core facility. Any further steps in the workflow are based on quality parameters entered by the core facility. In case of quality issues, an email notification can be sent to the user.

Editing Incoming Libraries and Samples

Any user with staff permission can see and edit requests awaiting incoming quality control and quality approval. To start quality control, choose the "Incoming samples/libraries" window. All requests containing samples/libraries with status "submission completed" are displayed and can be assessed. Click on the plus/minus icon to expand/collapse the requests and list all samples or libraries belonging to a request.

In the Incoming Libraries and Samples window quality information for each submitted sample/library is displayed. Left side of the window/red: quality criteria entered by request holder; right side of the window/green, editable table to document quality control.

	Status	Criteria
	Submission pending	Request created by user
	Submission completed	Request approved by principal investigator
	Quality check approved	Incoming quality control passed, ready for preparation
	Libraries prepared	Library quality check passed
	Pooled	Pooling completed, ready for sequencing
	Sequencing	Sequencing ongoing
	Failed	If samples fail during initial quality check or preparation
	Compromised	quality compromised, samples still move to the next stage

Fig. 11: Statuses of samples/libraries.

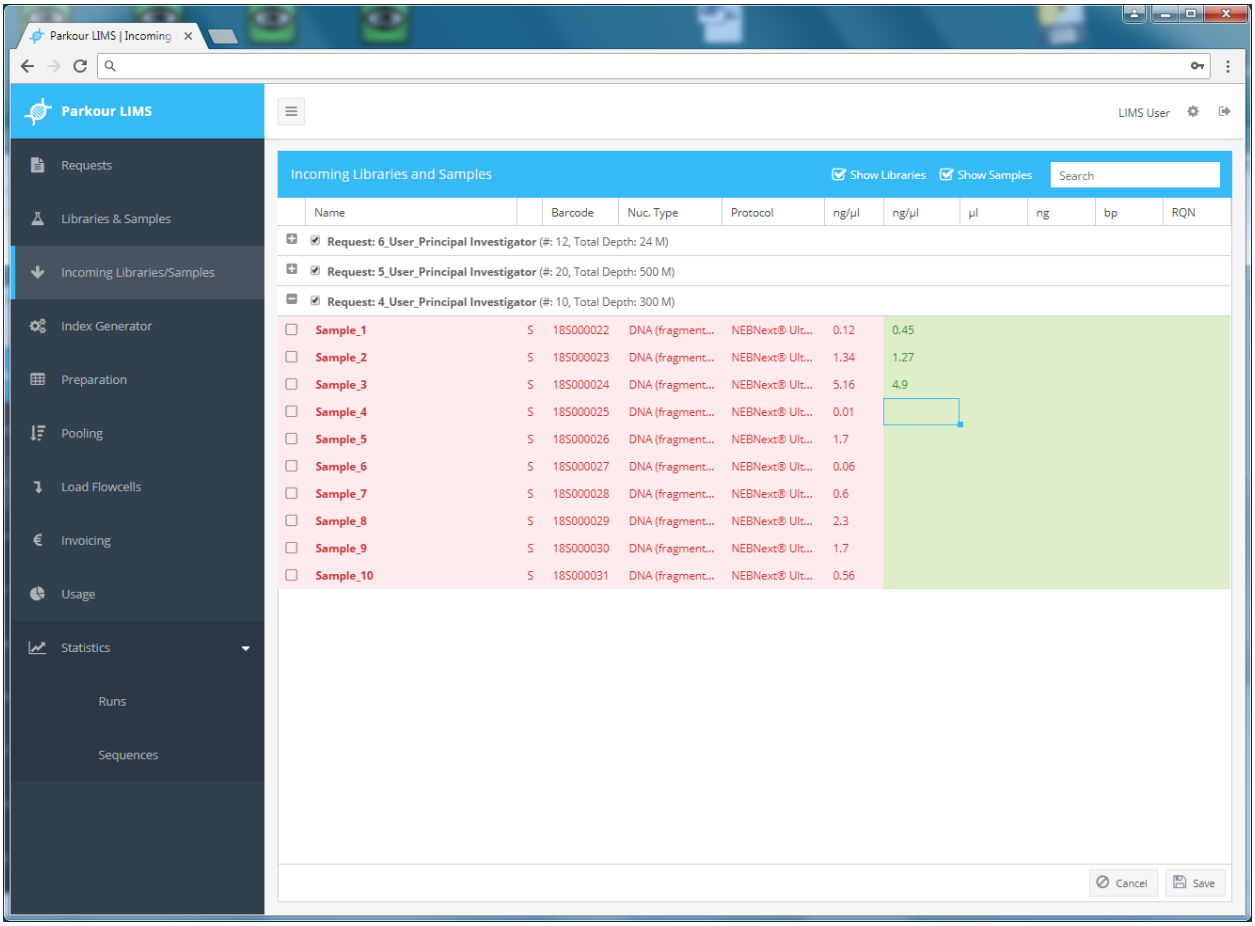

Fig. 12: Incoming Quality Control module.

Table 3: Editable Quality Control parameters

Parameter	Explanation	Field type
DF	Dilution factor, if values is >1 measurement value corresponds to a diluted and not the stock sample	Integer
Concentration (ng/µl)	measured concentration	Float
µl	volume of submitted sample or library	FLoat
ng	Amount of nucleic acid in stock sample (DF * ng/µl * µl)	Automatically calculated field, Float
F/S	Measurement technology: Fluorometry or Spectrophotometry	Select from drop-down menu
qPCR result	qPCR quantification result	Float
bp	Mean size distribution of sample/library	Integer
RNA quality (RQN)	RQN or RIN (RNA Quality Number or RNA integrity number), determined by software of capillary electrophoresis device	Float in the range 1-10
Comment	Additional information on libraries or sample quality	String

To enter parameters from i.e. concentration measurements choose a request and select a respective cell. Paste or type individual values into cells and press enter or save to fix values into the database. To paste a series of values into multiple cells, mark a cell, press Esc, start pasting data (ctrl + v).

To attach measurements reports to a request, choose from the request window your request, right click and choose view to start uploading files from the quality control step.

Quality Evaluation

Once all measurements are conducted and documented in Parkour, staff of the sample processing laboratory can start quality evaluation. To this, mark individual samples or libraries using the checkbox attached to each sample or library. Right click and select either passed, compromised or failed. To evaluate all samples or libraries of a request at once, right click on the request header, choose "select all" and one of the three depicted quality options.

Once evaluated, all samples or libraries that passed the quality control will change status to "quality check approved", clear from Incoming Libraries and Sample window and at the same time appear in the Index Generator window. All samples, evaluated as failed, will be rejected and will not appear in any of the subsequent steps. Such samples are flagged as failed and can be viewed in the window Libraries & Samples, Samples or libraries, evaluated as compromised, will move with the approved samples.

1.1.5 Index Generator

Sample Selection

Index Generator is one of the central components in Parkour LIMS. The tool groups samples by compatible index types and run conditions and assigns generated indices to them. The Index Generator chooses indices from predefined lists to ensure proper image registration on sequencing devices.

All samples or libraries with status quality approved will appear in the Index Generator for either index assignment or index validations. On the left side of the window, grouped by request, you can select libraries and samples for grouping and index validation or assignment.

Before starting working with the samples, you need to make sure that a Pool Size is specified (by selecting an option in the corresponding drop down menu). Ideally, you shouldn't exceed the chosen Pool Size, but you will still be able to save the samples as a pool.

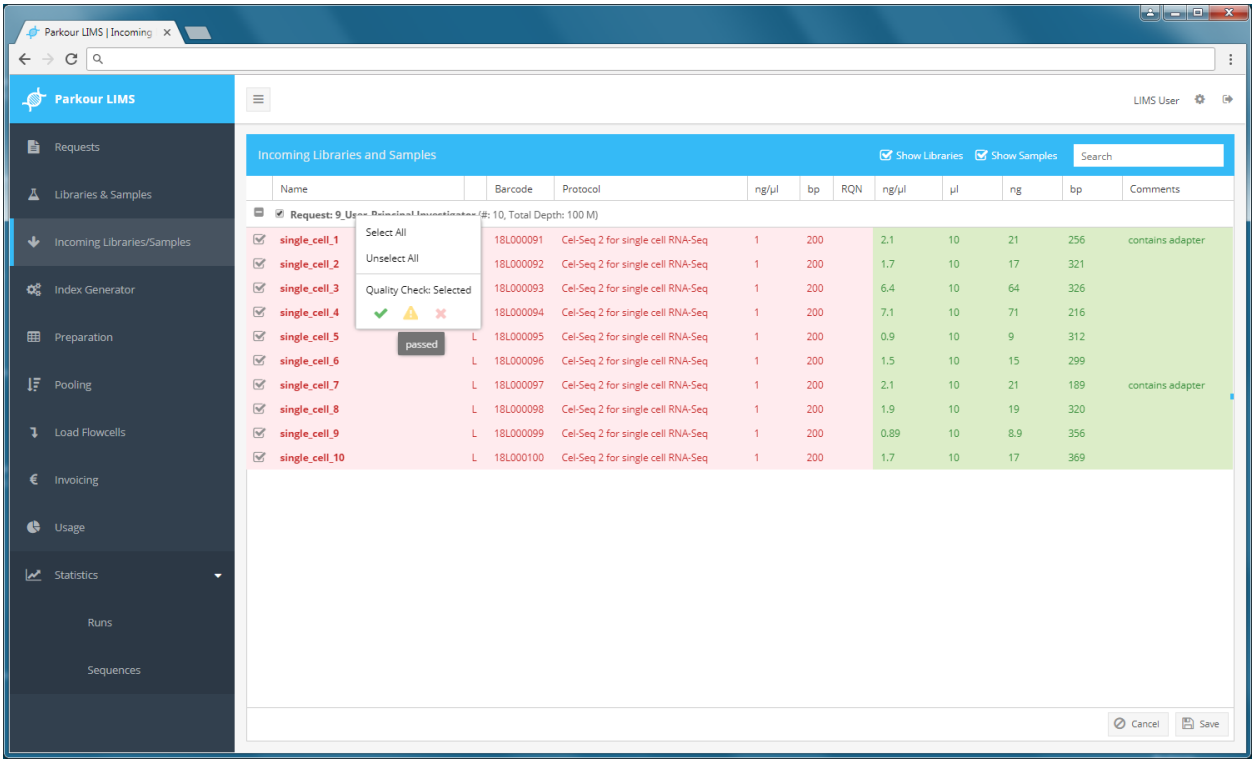

Fig. 13: Sample/library quality evaluation.

You can group only those libraries and samples if they have:

- the same read length
- compatible index type
  - The same index length (6 or 8 nucleotides)
  - Either Index I7 (single index) or Index I7 + Index I5 (dual index)

Index Type is required to be set for all samples to proceed.

The Index Generator can handle different index types and formats such as single- and dual-indexing in individual tubes or standard 96-well plates.

Index Generation

After you selected libraries and samples, you should see them on the right side of the window.

Now you can click the Generate Indices button. If all the requirements to the Pool Size and Index Types are met, you should be able to see the newly generated indices. Otherwise, a corresponding message(s) will appear.

Index Generator shows the color diversity of the generated indices as a ratio between the green (G/T) and red (A/C) nucleotides per column. Balanced signal in both the channels (red and green) is a prerequisite for proper image registration and base calling on Illuminas' sequencing instruments.

When indices have been generated, you can save the selected libraries and samples as a pool. For best organization and simple tracking a running number is assigned to each pool. The generated pool number is completely independent of the pooled/grouped requests. After index generation and pool saving, samples will disappear from the right side of the index generator window and appear in the next stage of the workflow.

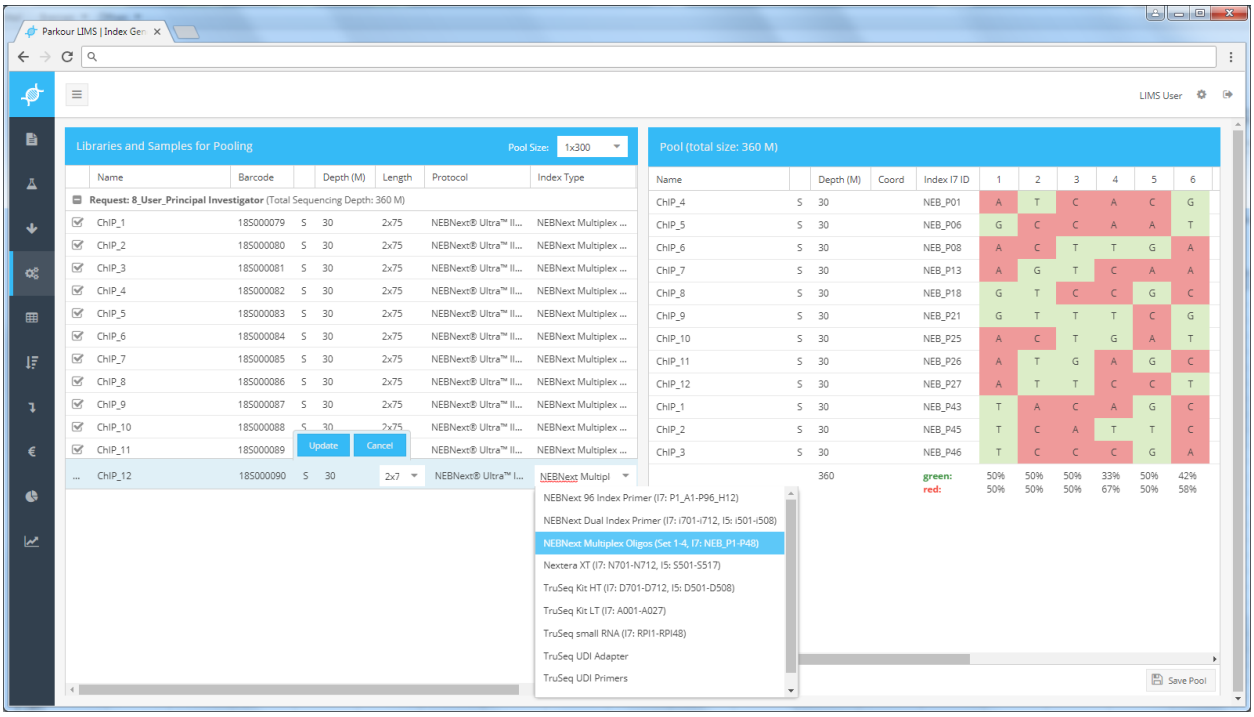

Fig. 14: Index Generator.

To maximize sample throughput, facilitate shallow sequencing and sequencing of small sample batches (do not fill complete lanes) the index generator provides the following grouping options.

A pool can consist of libraries only. To this the index generator displays index diversity of the already assigned indices and allows index saving only if none of the indices is duplicated. For dual indexed libraries the index generator allows duplicates in i5 or i7 index sequence, but rejects duplicated i5 or i7 index pairs.

A pool can consist of samples only. To this the index generator generates and assigns indices as described above.

A pool can contain samples and libraries. To do this, the index generator displays index sequence of libraries and assigns on top of those libraries indices to samples from predefined lists.

There are two principal approaches to the index generation.

Randomized

If the number of selected samples is less than a certain threshold, Index Generator will randomly pick an index for the first sample and then generate indices for the remaining samples, maximizing the color diversity score by reaching to the 50%/50% ratio as close as possible. Indices will be assigned to samples in an ascending order, based on the numbering of each index. This ensures simple and immaculate handling of index tubes during sample preparation in a laboratory.

Subsequent Indices

If the samples are set to have a 96-well plate index type, Index Generator will take subsequent index pairs (predefined Index I7 + Index I5 pairs), starting from a certain position and direction. By default the start position is A1 and the direction is right.

Assume we have a 9-well plate:

Table 4: 9-well plate.

	1	2	3
A	I7_1 - I5_1	I7_2 - I5_1	I7_3 - I5_1
B	I7_1 - I5_2	I7_2 - I5_2	I7_3 - I5_2
C	I7_1 - I5_3	I7_2 - I5_3	I7_3 - I5_3

Depending on the selected direction, Index Generator will consider index pairs in the following order:

- **Right**: A1, A2, A3, B1, . . . , C3
- **Down**: A1, B1, C1, A2, . . . , C3
- **Diagonal**: A1, B2, C3, A2, . . . , C2

The start position controls the first position to be considered. For example, given the start position B3 and the direction down, the index pairs will be taken in the following order: B3, C3, A1, B1, . . . , A3.

1.1.6 Preparation

All samples that undergo library preparation in the core facility will be listed in the preparation window of the Parkour LIMS. For simple overview and grouping of samples from multiple requests into a single library preparation, samples are grouped by protocol. Note: Any submitted library will skip the prepartion step and directly appear in the pooling step. To view all samples awaiting preparation of a selected protocol, click on the plus/minus icon to expand/collapse the list of samples.

To start sample preparation, mark all samples to be prepared. Either check each individual sample or right click on the header to select all samples from the respective protocol.

Click "Download Benchtop Protocol" to generate a spreadsheet with all the information. This spreadsheet will be the basis to start sample input normalization by amount and volume. The spreadsheet is appointed with formulas needed to calculate starting volume of sample and buffer and contains the layout for index assignment.

To edit the records in the Parkour preparation table, mark the respective cell and enter or paste a value. Press enter or update to apply the changes. To edit the tables column-wise, choose the topmost cell press Esc and paste the data, e.g., concentration measurements into the respective column. To fill columns with identical values, type in a value press enter, right click, select "Apply to All" to fill subsequent cells with identical information.

Once all mandatory fields are filled, evaluate the library result and choose "passed" to approve library quality or "fail" to stop further processing. To evaluate individual or all samples belonging to a sample preparation protocol, mark the individual samples or mouse over the protocol header, right click, choose "select all" and carry out the evaluation for all marked samples. Once evaluated, libraries will disappear from the Preparation tab and appear in the Pooling tab. All failed libraries will be rejected and are marked in the Libraries and Samples tab with the status "failed".

Table 5: Editable Preparation parameters

Parameter	Explanation	Field type
ng/µl samples	Concentration of sample. Filled with value from quality check. Can get overwritten if needed	Float
Starting amount (ng)	Amount of nucleic acid used for library preparation	Float
Spike-in	Description of Spike-In	Float
Spike-in µl	Volume of Spike-In (DF * ng/µl * µl)	Float
Cycles	Number of PCR cycles used to enrich sequencing library	Integer
ng/µl library	Concentration library	Float
bp (mean fragment size)	Mean size distribution of library	Integer
nM	Calculation of library yield. Conversion of ng/µl into nM	Automatically calculated field
Comment	Additional information on library	String

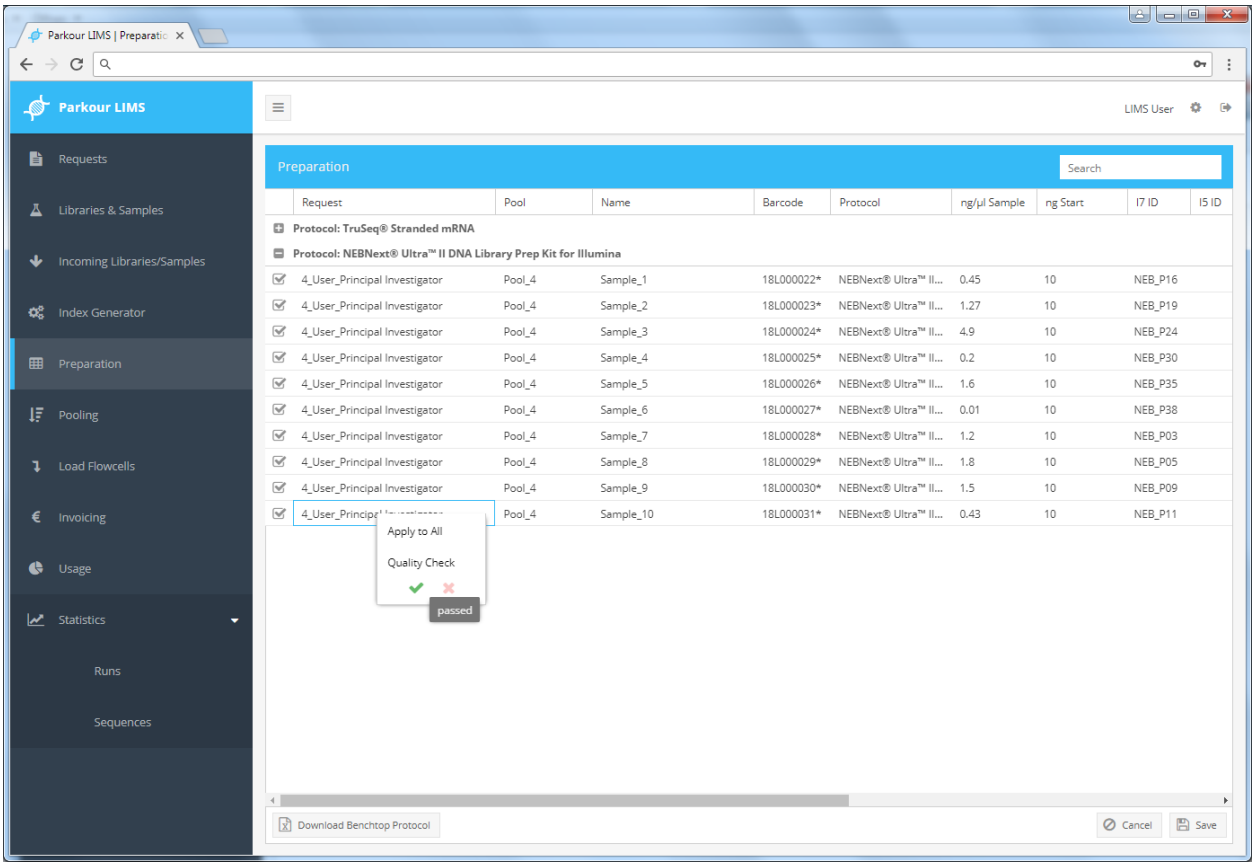

Fig. 15: Preparation module.

1.1.7 Pooling

In the Pooling stage, you can see all samples that reach the status library quality check passed. This can either be libraries generated in the core laboratory (submitted as sample) or libraries submitted to Parkour LIMS. Libraries

are grouped by pool number. All pools, ready for processing (all libraries in the pool reached the status "libraries prepared") are colored in green. In contrast, pools containing non-finished libraries are colored in red.

To start pooling, select a Pool and expand the list of associated libraries by clicking on the plus/minus icon on the left.

To choose all libraries of a pool, right click on the header, click "Select All" and click "Download Benchtop Protocol". All pooling-relevant information will be printed into a spreadsheet that can be used for further editing and serves as template to start library pooling.

Once pooling is completed, select all libraries in a pool and choose "passed". Status changes to Pooling completed, ready for sequencing and Pool is cleared from Pooling window.

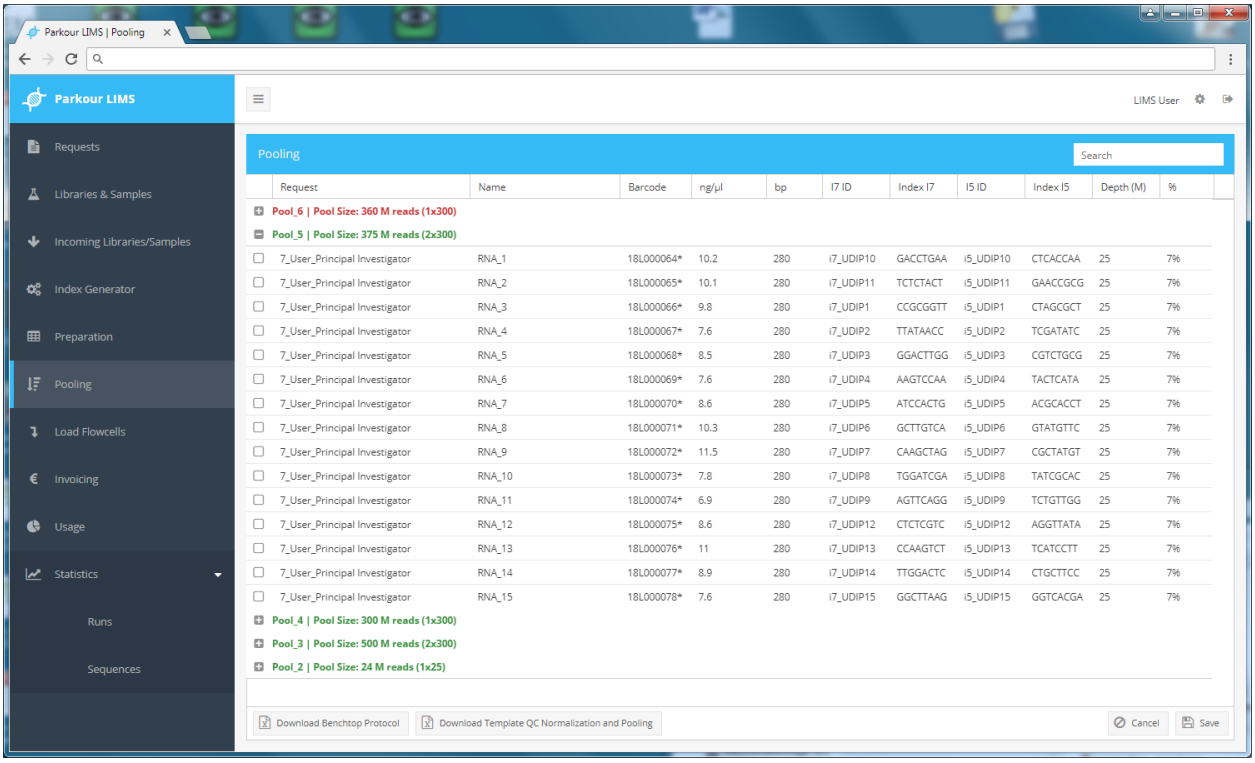

Fig. 16: Pooling module.

1.1.8 Load Flowcells

In the Flowcells tab, you can observe all flowcells loaded in a given time period and load pools onto flowcells to start a sequencing run.

Loading a flowcell

To assign pools to lanes of a flowcell, click "Load" in the top right corner of the load flowcell window. The load flowcell dialogue appears. All Pools ready for loading (status pooled) appear in green. Pools colored in red are not ready for loading and can not be dragged onto a lane.

To start choose a sequencing instrument using the dropdown menu. Note that depending on the sequencer, you will see either one or eight lanes. Enter the flowcell ID. Start dragging the desired pools into the displayed lanes. Note that for loading of an 8-lane flowcell, read length and size of all pools must be identical (parameters are shown in the

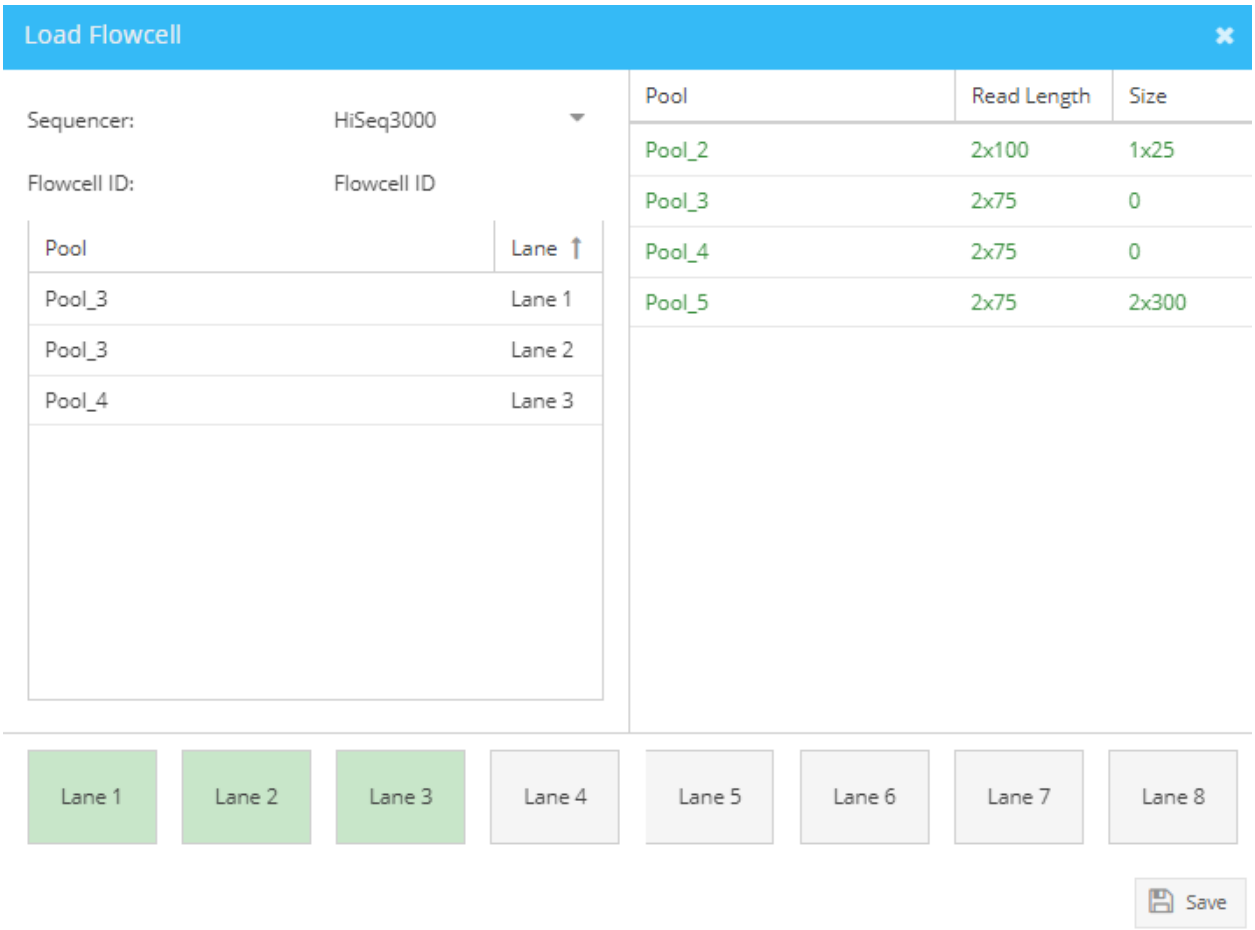

Fig. 17: Load Flowcell window.

pool list). Also, all lanes must be loaded to complete the flowcell loading. Click "Save". Sample status changes to "Sequencing".

View the newly generated flowcell in the load flowcell window. Expand the flowcell to see all lanes loaded. Edit the loading concentration per lane as well as the percentage of Spike-in (i.e. PhiX) in use. To download the flowcell layout, select all lanes of a flowcell and click the "Download Benchtop Protocol" button. Especially for 8-lane flowcells this function is convenient to correctly match pools to lanes/8-well strips in the laboratory.

Generating a sample sheet

To download the sample sheet in CSV format expand the flowcell, select all lanes and click "Download Sample Sheet". The generated CSV file contains all information needed to start the demultiplexing after a sequencing run is completed. Note: Parkour generates for all instruments and index designs sample sheets with identical formats.

1.1.9 Reporting

Parkour provides reporting functionality by generating a quality report, summarizing the quality matrices of received samples, library construction details as well as details on cluster generation and sequencing.

To generate a per request quality report open the request view page and right click on the respective request. Choose from the dropdown menu Complete Report." Use the quality report to view quality matrices of received samples, details on library construction, cluster generation and sequencing. Furthermore, a detailed appendix provides detailed information on library construction, sequencing devices and software versions to be used as template for convenient editing of your publication.

1.1.10 Statistics

Usage

At any time a user with the staff permission and higher can access the interactive statistics to monitor usage of laboratory and workflows for a chosen period of time. Choose "Usage" in the navigation panel on the left side and set a date range. Pie charts illustrate proportions of submitted samples and libraries, organizations, principal investigators and library types. Bar charts represent the number of submitted samples or libraries per principle investigator or library type. Any chart can be downloaded as png file.

Runs and Sequences

To evaluate the success of a given sequencing run specifications are imported into Parkour LIMS and shown in the tabs runs and sequences. Note that Parkour mainly imports and displays the listed parameters. Calculations are done using non-Parkour software and scripts.

Runs

To view run specifications, navigate to the "Runs" tab and select a time range.

Table 6: Run parameters

Parameter	Explanation
Lane	The lane number on the flowcell. Lane number will depend on the sequencing instrument in use. MiSeq: 1 lane, HiSeq2500 Rapid: 2 lanes, NextSeq: 4 lanes, HiSeq3000: 8 lanes.
Pool	Information on pool loaded on a given lane. Per lane only one pool can be loaded.
Request	Information on request(s) loaded on a given lane
Preparation Method	Information on library preparation method
Library Type	Information on library type
Loading Concentration	DNA loading concentration per lane
Cluster Pass Filter (PF) %*	Percentage of clusters passing Illumina's quality filter
Reads PF (%)*	Percentage of reads (read-pairs for paired-end datasets) passing quality filter
Undetermined Indeces (%)*	Percentage of undetermined indeces after demultiplexing
Spike-In (%)*	Percentage of Spike-In used to accomplish nucleotide diversity (typically PhiX library, Illumina)
Read 1 % bases >= Q30*	Quality measure for read 1, percentage of bases in read 1 having a Phred-scaled score of at least 30
Read 2 % bases >= Q30*	Quality measure for read 2, Êpercentage of bases in read 2 having a Phred-scaled score of at least 30

Sequences

To assess quality of each sequenced sample, navigate to "Sequences" tab. To download parameters into a spreadsheet, check samples and click "Download Report".

Table 7: Sequences parameters

Parameter	Explanation
Request	Running number and information on user and principal investigator
Barcode	Sample barcode, running number automatically generated from Parkour upon request generation
Name	Sample name given by user
Lane	Lane number sample was sequenced on
Pool	Group of samples the sequenced sample was part of
Library Protocol	Information on library preparation method
Library Type	Information on library type
Reads (M) (requested)	Number of read (read-pairs for paired end sequencing) requested by user
Reads (M) sequenced*	Number of reads (read-pairs for paired end sequencing) generated by sequencer
Confident off-species reads*	Percentage of reads uniquely aligned to another, but the target organism
% optical duplictes*	Percentage of duplicates in nearby wells on patterned flowcells. Reason: During cluster generation a library can occupy two adjacent wells
% dupped reads*	Percentage of duplicated reads

1.1.11 Invoicing

An automated invoicing system is part of the Parkour LIMS. Each request, that reaches status "Sequencing", will appear in the "Invoicing" tab. Processed requests are presented per month. For data editing and sharing information needs to be download into a spreadsheet. An upload functions is installed for documentation of final cost reports.

Price calculation is based on predefined costs for service, consumables and reagents. Princes are stored in Parkour and are freely editable.

Prices are calculated for sample preparation or performed quality control by multiplication of the number of samples, that reach the status "quality approved", times costs for the respective protocol. If prepared libraries are submitted to Parkour, preparation costs are calculated by multiplication of the number of approved libraries times preset costs for quality control per sample. Prices for sequencing of samples are calculated by multiplication of requested sequencing depth times preset prices for sequencing on a certain instrument with preset run conditions. Note that the user is billed exclusively for the number of requested reads.

Final costs for a request are the sum of the calculated costs for sample preparation (can be a quality control only) and the calculated costs for sequencing. Additionally, to account for labor and instrument usage, overhead costs can be added to the calculated costs for preparation and sequencing.

1.1.12 Metadata Export

To deposit sequences in a public archive (ENA) detailed documentation of the conducted experiment will be requested. Public repositories provide templates to standardize documentation and data upload. Parkour LIMS provides a convenient ENA Export tool that prepares data required for a successful upload to ENA. The system collects the data related to the experiment and asks the user to enter the remaining information. The upload is done in multiple steps. First, Parkour LIMS generates four TSV files, *studies.tsv*, *experiments.tsv*, *samples.tsv*, and *runs.tsv*. Then, they need to be uploaded to Galaxy, which will convert these TSV files into XML files accepted by ENA according to ENA rules. Secondary analysis can be done on metadata utilyzing Galaxy tools and workflows. Finally, an ENA upload tool in Galaxy will handle the upload.

To start working with the export tool, right-click on a request in the *Requests* tab and select *ENA Export*. The tool window will be shown where the user has to fill in general information about the experiment. Galaxy URL and API key must be specified in order to upload the files directly to Galaxy. Alternatively, this can be done by downloading the TSV files, previewing and manually uploading them to Galaxy.

The window's second tab allows the user to enter and edit all sample metadata. The same editing capabilities, which are implemented across the whole system, can be used here, i.e., *Apply to All* and *per-cell editing*. Input validation ensures all the information is present.

When all the required data is provided, the user can either download the TSV files or push them directly to Galaxy, which will upload the data to ENA.

1.2 Installation

1.2.1 Quick start

It is assumed you have a recent version of Docker running and the docker-compose tool installed.

Clone the repository:

```
git clone https://github.com/maxplanck-ie/docker-parkour.git
cd docker-parkour
```

Build the images and start the services:

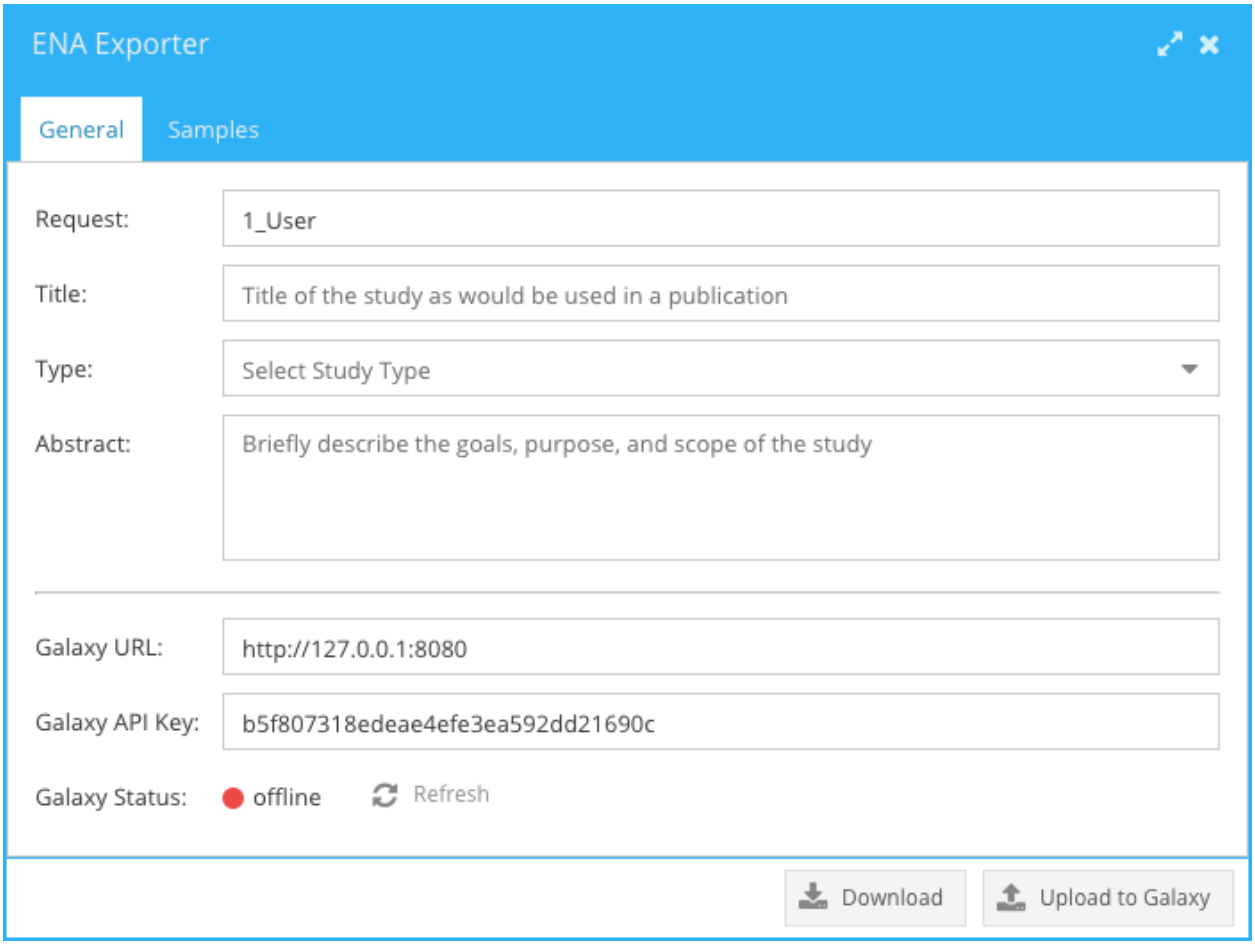

Fig. 18: ENA Exporter. Entering general information about an experiment.

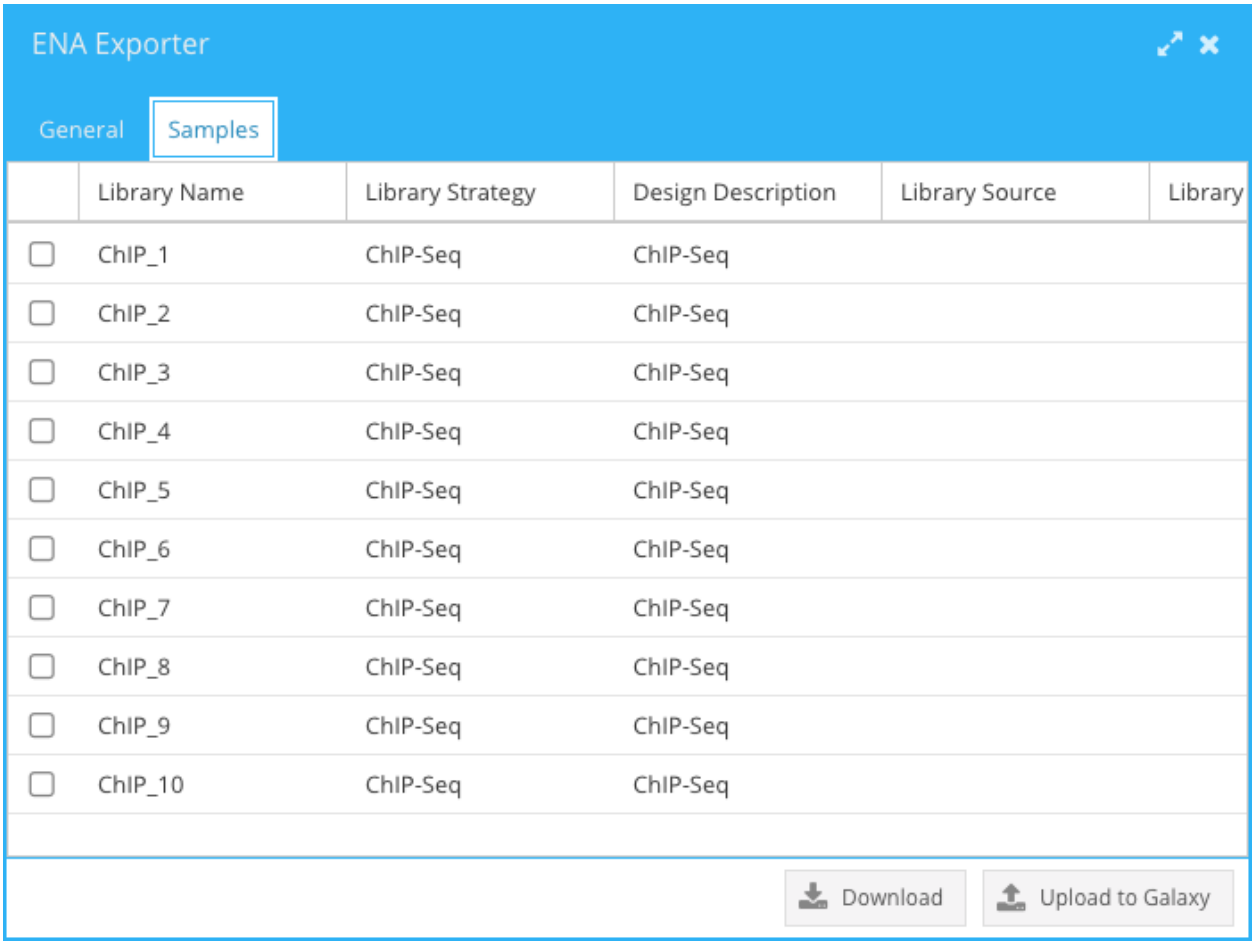

Fig. 19: ENA Exporter. Editing sample metadata.

```
docker-compose up -d --build
```

Migrate the database tables:

```
docker-compose run parkour-web python manage.py migrate
```

Collect static files:

```
docker-compose run parkour-web python collectstatic --no-input --verbosity 0
```

Create a superuser (admin):

```
docker-compose run parkour-web python manage.py createsuperuser
```

Open Parkour LIMS at `http://localhost/`

1.2.2 Manual setup

Prerequisites

- Python 3.6
- PostgreSQL

Configure the database

```
CREATE DATABASE <DB_NAME>;
CREATE USER <DB_USER> WITH PASSWORD <DB_PASS>;
GRANT ALL PRIVILEGES ON DATABASE <DB_NAME> TO <DB_USER>;
```

Export environment variables

```
export SECRET_KEY=<SECRET_KEY>
export DJANGO_SETTINGS_MODULE=wui.settings.dev
export DATABASE_URL=postgres://<DB_USER>@<DB_HOST>:<DB_PORT>/<DB_NAME>
```

Installation steps

Clone the repository:

```
git clone https://github.com/maxplanck-ie/parkour.git
cd parkour
```

Install the requirements:

```
pip install -r requirements/dev.txt
```

Migrate the database tables:

```
python manage.py migrate
```

Create a superuser (admin):

```
python manage.py createsuperuser
```

Run the server:

```
./manage.py runserver
```

1.3 Web API

Parkour provides a web API that can be used to facilitate both downstream processing and returning of alignment and other metrics back for record keeping and potential future analysis.

Examples below are given in python using python 3, though any programming language could be used in practice provided it can construct JSON strings.

1.3.1 General considerations

The user account used to interact with the web API must have "staff" permissions. Further, since the password is used for authentication, you are strongly encouraged to ensure that this password cannot be read by other users.

1.3.2 Submitting per-lane statistics to Parkour

Illumina sequencing runs produce a variety of metrics that are useful to track. These include:

- The percentage of clusters that pass filtering
- The number of reads passing filtering
- The percentage of reads with undetermined indices
- The percentages of bases (for read 1 and read 2) with quality scores of at least 30

The *api/run_statistics/upload* URL can be used to submit metrics of this sort per-lane per-flowcell. To do so, one must first create a JSON string of the following form:

```
{
  "matrix": [
    {
      "name": "Lane 1",
      "cluster_pf": "90.41",
      "reads_pf": "107641925",
      "undetermined_indices": "9.26%",
      "read_2": "85.23",
      "read_1": "94.08"
    },
    {
      "name": "Lane 2",
      "cluster_pf": "90.39",
      "reads_pf": "105981493",
      "undetermined_indices": "9.31%",
      "read_2": "85.04",
      "read_1": "94.11"
    }
  ],
```

(continues on next page)

(continued from previous page)

```
  "flowcell_id": "HVKWLBGX7"
}
```

The *flowcell_id* should match what is in Parkour. The rest of the content is that described above, where *cluster_pf* is the percentage of clusters passing filtering, *reads_pf* is the number of reads passing filtering, *undeterminded_indices* is the percentage of reads with undetermined indices, and *read_1* and *read_2* are the percentages of bases in reads 1 and 2 with quality scores of at least 30. The following python code demonstrates how to submit this to Parkour:

```
import json
import requests

URL = "https://parkour-demo.ie-freiburg.mpg.de/api/run_statistics/upload"
user = "some email address"
password = "A password that you should keep secret!"

d = dict()
d['flowcell_id'] = "HVKWLBGX7"
m = [{
      "name": "Lane 1",
      "cluster_pf": "90.41",
      "reads_pf": "107641925",
      "undetermined_indices": "9.26%",
      "read_2": "85.23",
      "read_1": "94.08"
    },
    {
      "name": "Lane 2",
      "cluster_pf": "90.39",
      "reads_pf": "105981493",
      "undetermined_indices": "9.31%",
      "read_2": "85.04",
      "read_1": "94.11"
    }]
d['matrix'] = json.dumps(m)
res = requests.post(URL, auth=(user, password), data=d)
```

Within Parkour, users with "staff" accounts can view these metrics by clicking on "Statistics" and then "Runs".

1.3.3 Query Parkour for information on a flowcell

In order to process samples, downstream processes need to know the following information:

1. What organism the sample comes from.
2. The type of experiment (e.g., RNA-seq or ChIP-seq) a sample comes from.
3. The library preparation protocol.

This can be queried on a per-flowcell basis using the *api/analysis_list/analysis_list/* web API. This take a single *GET* query with a flowcell ID that must already exist in Parkour:

```
import requests

URL = "https://parkour-demo.ie-freiburg.mpg.de/api/analysis_list/analysis_list/"
user = "some email address"
password = "A password that you should keep secret!"
```

(continues on next page)

(continued from previous page)

```
d = {"flowcell_id": "HVKWLBGX7"}
res = requests.get(URL, auth=(user, password), params=d)
if res.status_code == 200:
    # do something with res.json()
```

An example of the output is as follows:

```
{
 "528_Ryan_Boenisch": {
                       "18L008007": ['Input', 'ChIP-Seq', 'NEBNext Ultra II DNA␣
↪Library Prep Kit for Illumina', 'mouse'],
                       "18L008008": ['H3K4me3', 'ChIP-Seq', 'NEBNext Ultra II DNA␣
↪Library Prep Kit for Illumina', 'mouse']
                      },
 "529_Anatskiy_Manke": {
                       "18L008009": ['Brain', 'single-cell RNA-seq', '10xGenomics for␣
↪single cell RNA-Seq', 'mouse'],
                       "18L008010": ['Liver', 'single-cell RNA-seq', '10xGenomics for␣
↪single cell RNA-Seq', 'mouse']
                      }
}
```

The result is a dictionary of dictionaries. Each element of the outer-most dictionary is a single project in Parkour (*528_Ryan_Boenisch* in this case). The inner-most dictionary has keys of the library ID (e.g., *18L008007*) and values an orderd list of: sample name, library type, library protocol, and organism.

1.3.4 Reporting downstream metrics back to Parkour

Standard metrics such as alignment rate can be returned to Parkour so that the sequencing facility can track how changes to library preparation protocols affect downstream results. The downstream and per-sample metrics that we report back include:

1. Reads passing filter (*reads_pf_sequenced*)
2. Confidently off-species alignment rate (*confident_reads*)
3. Optical duplication rate (*optical_duplicates*)
4. Percentage mapped (*mapped_reads*)
5. Percentage marked as duplicates (*dupped_reads*)
6. Median insert size (*insert_size*)

Each of these metrics is optional! To submit these metrics back to Parkour, one can use the *api/sequences_statistics/upload/* URL with a POST method. As above, a JSON string is created that stores each of these metrics and associates them to a library ID:

```
import requests
import json

URL = "https://parkour-demo.ie-freiburg.mpg.de/api/sequences_statistics/upload/"
user = "some email address"
password = "A password that you should keep secret!"

m = [{"barcode": "18L008007",
```

(continues on next page)

(continued from previous page)

```
      "reads_pf_sequenced": 123456,
      "confident_reads": 0.001,
      "optical_duplicates": 0.01,
      "mapped_reads": 95.20,
      "dupped_reads": 5.23,
      "insert_size": 150},
     {"barcode": "18L008008",
      "reads_pf_sequenced": 250743,
      "confident_reads": 0.003,
      "optical_duplicates": 0.02,
      "mapped_reads": 94.71,
      "dupped_reads": 4.92,
      "insert_size": 152}]

d = {"flowcell_id": "HVKWLBGX7"}
d['sequences'] = json.dumps(m)
res = requests.post(URL, auth=(user, password), data=d)
```

Users with "staff" accounts can then view this metrics from within Parkour by clicking on "Statistics" and then "Sequences".

1.4 API Documentation

1.4.1 Request API

API operations on requests.

class `request.views.`**PDF**(*title='Title', font='Arial'*)

Bases: `fpdf.fpdf.FPDF`

footer()

Footer to be implemented in your own inherited class

header()

Header to be implemented in your own inherited class

info_row(*title, value*)

multi_info_row(*title, value*)

table_row(*index, name, barcode, type, depth, bold=False*)

class `request.views.`**Report**(*title='Report', font='Arial'*)

Bases: `fpdf.fpdf.FPDF`, `fpdf.html.HTMLMixin`

footer()

Footer to be implemented in your own inherited class

generate_html_table(*data*)

header()

Header to be implemented in your own inherited class

page_header(*text*)

text_block(*text, style='', size=11, multi=False*)

class `request.views.`**RequestViewSet**(***kwargs*)

Bases: `rest_framework.viewsets.ModelViewSet`

create (*request*)
Create a request.

download_RELACS_Pellets_Abs_form (*request*)

download_complete_report (*request*, *pk=None*)

download_deep_sequencing_request (*request*, *pk=None*)
Generate a deep sequencing request form in PDF.

edit (*request*, *pk=None*)
Update request with a given id.

filter_backends = (<class 'rest_framework.filters.SearchFilter'>,)

get_files (*request*, *pk=None*)
Get the list of attached files for a request with a given id.

get_files_after_upload (*request*)

get_queryset (*showAll=False*)
Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

You may want to override this if you need to provide different querysets depending on the incoming request.

(Eg. return a list of items that is specific to the user)

get_records (*request*, *pk=None*)
Get the list of record's submitted libraries and samples.

list (*request*)
Get the list of requests.

mark_as_complete (*request*, *pk=None*)
Mark request as complete, set sequenced to true

pagination_class
alias of `common.views.StandardResultsSetPagination`

samples_submitted (*request*, *pk=None*)

search_fields = ('name', 'description', 'user__first_name', 'user__last_name')

send_email (*request*, *pk=None*)
Send an email to the user.

serializer_class
alias of `request.serializers.RequestSerializer`

upload_deep_sequencing_request (*request*, *pk=None*)
Upload a deep sequencing request with the PI's signature and change request's libraries' and samples' statuses to 1.

upload_files (*request*)

1.4.2 Library API

API operations on libraries.

class library.views.**LibrarySampleTree**(***kwargs*)
Bases: rest_framework.viewsets.ViewSet

get_queryset(*showAll=False*)

list(*request*)
Get the list of libraries and samples.

class library.views.**LibraryViewSet**(***kwargs*)
Bases: library_sample_shared.views.LibrarySampleBaseViewSet

serializer_class
alias of library.serializers.LibrarySerializer

1.4.3 Sample API

API operations on samples.

class sample.views.**NucleicAcidTypeViewSet**(***kwargs*)
Bases: rest_framework.viewsets.ReadOnlyModelViewSet

Get the list of nucleic acid types.

get_queryset()
Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

You may want to override this if you need to provide different querysets depending on the incoming request.

(Eg. return a list of items that is specific to the user)

serializer_class
alias of sample.serializers.NucleicAcidTypeSerializer

class sample.views.**SampleViewSet**(***kwargs*)
Bases: library_sample_shared.views.LibrarySampleBaseViewSet

serializer_class
alias of sample.serializers.SampleSerializer

1.4.4 Quality Check API

API operations on Quality Check.

class incoming_libraries.views.**IncomingLibrariesViewSet**(***kwargs*)
Bases: common.mixins.LibrarySampleMultiEditMixin, rest_framework.viewsets.ViewSet

library_model
alias of library.models.Library

library_serializer
alias of incoming_libraries.serializers.LibrarySerializer

list(*request*)
Get the list of all incoming libraries and samples.

permission_classes = [<class 'rest_framework.permissions.IsAdminUser'>]

sample_model
alias of `sample.models.Sample`

sample_serializer
alias of `incoming_libraries.serializers.SampleSerializer`

1.4.5 Index Generator API

API operations on pools.

class `index_generator.views.`**GeneratorIndexTypeViewSet**(***kwargs*)
Bases: *index_generator.views.MoveOtherMixin*, `rest_framework.viewsets.ReadOnlyModelViewSet`

Get the list of index types.

queryset

serializer_class
alias of `library_sample_shared.serializers.IndexTypeSerializer`

class `index_generator.views.`**IndexGeneratorViewSet**(***kwargs*)
Bases: `rest_framework.viewsets.ViewSet`, `common.mixins.LibrarySampleMultiEditMixin`

generate_indices(*request*)
Generate indices for given libraries and samples.

library_model
alias of `library.models.Library`

library_serializer
alias of `index_generator.serializers.IndexGeneratorLibrarySerializer`

list(*request*)
Get the list of libraries and samples ready for pooling.

permission_classes = [<class 'rest_framework.permissions.IsAdminUser'>]

sample_model
alias of `sample.models.Sample`

sample_serializer
alias of `index_generator.serializers.IndexGeneratorSampleSerializer`

save_pool(*request*)
Create a pool after generating indices, add libraries and "converted" samples to it, update the pool size, and create a Library Preparation object and a Pooling object for each added library/sample.

class `index_generator.views.`**MoveOtherMixin**
Bases: `object`

Move the *Other* option to the end of the returning list.

list(*request*)

class `index_generator.views.`**PoolSizeViewSet**(***kwargs*)
Bases: `rest_framework.viewsets.ReadOnlyModelViewSet`

Get the list of pool sizes.

queryset

serializer_class
alias of `index_generator.serializers.PoolSizeSerializer`

class `index_generator.index_generator.`**IndexRegistry**(*mode*, *index_types*, *start_coord='A1'*, *direction='right'*)

Bases: `object`

Class for storing fetched and sorted indices i7/i5 and index pairs.

static create_index_dict(*format=''*, *index_type=''*, *prefix=''*, *number=''*, *index=''*, *coordinate=''*, *is_library=False*)

fetch_indices(*index_type*)
Fetch indices i7 and i5 for a given index type.

fetch_pairs(*index_type*, *char_coord*, *num_coord*, *direction*)
Fetch index pairs (Index i7 + Index i5) for a given index type, start coordinate, and direction.

get_diagonal(*index_pairs*)
Sort index pairs diagonally.

get_indices(*index_type_id*, *index_group*)
Return a list of indices for a given index type id and index group.

get_pairs(*index_type_id*)
Return a list of index pairs for a given index type id.

static split_coordinate(*coordinate*)
Split a submitted coordinate into a character and a numeric parts.

to_list(*format*, *index_type*, *indices*)
Return a list of index dicts.

class `index_generator.index_generator.`**IndexGenerator**(*library_ids*, *sample_ids*, *start_coord*, *direction*)

Bases: `object`

Main class that fetches provided libraries and samples, checks the compatibility of their index types, generates indices, and assigns them to the libraries and samples.

MAX_ATTEMPTS = 30

MAX_RANDOM_SAMPLES = 5

add_libraries_to_result()
Add all libraries directly to the result.

calculate_color_distribution(*indices*, *sequencing_depths*, *sample*)

calculate_scores(*current_sample*, *current_converted_index*, *current_color_distribution*, *total_depth*)
Calculate the scores for a given sample.

Score is an absolute difference between the sequencing depths of the two indices divided by the total sequencing depth (in %).

The ideal score is 0.0 (50% green and 50% red), an acceptable score is 60.0 (80%/20% or 20%/80%).

If the score > 60%, then the indices are not compatible.

static convert_index(*index*)
Convert A/C into R (red) and T into G (green).

static create_result_dict(*obj*, *index_i7*, *index_i5*)

find_index (*sample*, *index_group*, *current_indices*, *depths*)
Helper function for *find_indices()*.

find_indices (*samples*, *depths*, *index_group*, *init_indices*)
Generate indices for given samples and index group (I7/I5).

find_pair (*sample*, *depths*, *current_pairs*)
Helper function for *find_pairs()*.

find_pairs (*samples*, *depths*, *init_pairs*)
Generate index pairs for given samples.

find_pairs_fixed (*plate_samples*)
Return subsequent index pairs from the Index Registry starting from the first one.

find_random (*sample*)
Find a pair of random indices I7/I5 for a given sample.

format = ''

generate ()
Main method that generates indices.

index_length = 0

index_registry = None

libraries = None

mode = ''

num_libraries = 0

num_samples = 0

result
Construct a list of all records and their indices.

samples = None

static sort_indices (*indices*)
Sort indices I7/I5 by ID.

static sort_pairs (*pairs*)
Sort index pairs (only by Index I7 ID).

static sort_sequencing_depths (*depths*)
Sort sequencing depths to improve the result.

validate_index_types (*records*)
Check the compatibility of provided libraries and samples.

1.4.6 Library Preparation API

API operations on Library Preparation.

class library_preparation.views.**LibraryPreparationViewSet** (***kwargs*)
Bases: common.mixins.MultiEditMixin, rest_framework.viewsets.ReadOnlyModelViewSet

download_benchtop_protocol (*request*)
Generate Benchtop Protocol as XLS file for selected samples.

get_context (*queryset*)

get_queryset()
Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

You may want to override this if you need to provide different querysets depending on the incoming request.

(Eg. return a list of items that is specific to the user)

list(*request*)

permission_classes = [<class 'rest_framework.permissions.IsAdminUser'>]

serializer_class
alias of `library_preparation.serializers.LibraryPreparationSerializer`

1.4.7 Pooling API

API operations on Pooling.

class `pooling.views.`**PoolingViewSet**(***kwargs*)
Bases: `common.mixins.LibrarySampleMultiEditMixin`, `rest_framework.viewsets.ModelViewSet`

download_benchtop_protocol(*request*)
Generate Benchtop Protocol as XLS file for selected records.

download_pooling_template(*request*)
Generate Pooling Template as XLS file for selected records.

edit_comment(*request*, *pk=None*)

get_context(*queryset*)

get_queryset()
Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

You may want to override this if you need to provide different querysets depending on the incoming request.

(Eg. return a list of items that is specific to the user)

library_model
alias of `library.models.Library`

library_serializer
alias of `pooling.serializers.PoolingLibrarySerializer`

list(*request*)
Get the list of all pooling objects.

permission_classes = [<class 'rest_framework.permissions.IsAdminUser'>]

sample_model
alias of `sample.models.Sample`

sample_serializer
alias of `pooling.serializers.PoolingSampleSerializer`

1.4.8 Flowcell API

API operations on flowcells.

class `flowcell.views.`**FlowcellAnalysisViewSet**(***kwargs*)
Bases: `rest_framework.viewsets.ViewSet`

analysis_list(*request*)
This returns a dictionary of the information required to run an automated analysis on the flow cell's contents The keys of the dictionary are projects. The values are then a dictionary dictionaries with library name keys and tuple values of (sample/library name, library type, library protocol type, organism).

permission_classes = [<class 'rest_framework.permissions.IsAdminUser'>]

class `flowcell.views.`**FlowcellViewSet**(***kwargs*)
Bases: `common.mixins.MultiEditMixin`, `rest_framework.viewsets.ReadOnlyModelViewSet`

create(*request*)
Add a flowcell.

download_benchtop_protocol(*request*)
Generate Benchtop Protocol as XLS file for selected lanes.

download_sample_sheet(*request*)
Generate Benchtop Protocol as XLS file for selected lanes.

get_queryset()
Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

You may want to override this if you need to provide different querysets depending on the incoming request.

(Eg. return a list of items that is specific to the user)

list(*request*, **args*, ***kwargs*)

permission_classes = [<class 'rest_framework.permissions.IsAdminUser'>]

pool_list(*request*)

serializer_class
alias of `flowcell.serializers.LaneSerializer`

class `flowcell.views.`**PoolViewSet**(***kwargs*)
Bases: `rest_framework.viewsets.ReadOnlyModelViewSet`

permission_classes = [<class 'rest_framework.permissions.IsAdminUser'>]

queryset

retrieve(*request*, *pk=None*)
Get libraries and samples for a pool with a given id.

serializer_class
alias of `flowcell.serializers.PoolInfoSerializer`

class flowcell.views.**SequencerViewSet**(***kwargs*)

Bases: rest_framework.viewsets.ReadOnlyModelViewSet

Get the list of sequencers.

queryset

serializer_class

alias of flowcell.serializers.SequencerSerializer

A demonstration instance is available at http://parkour-demo.ie-freiburg.mpg.de. The following accounts are available on that instance:

- A typical "staff" account with the username "parkour-staff@parkour-demo.ie-freiburg.mpg.de" and password "parkour-staff".
- A typical "admin" account with the username "parkour-admin@parkour-demo.ie-freiburg.mpg.de" and password "parkour-admin".

Please note that the instance is reset every 12 hours!

Python Module Index

Index

M

N

P

Q

R

S

T

U

V