Министерство образования и науки Российской Федерации
Государственное образовательное учреждение высшего профессионального образования «Московский физико-технический институт (государственный университет)»
Факультет инноваций и высоких технологий
Кафедра алгоритмов и технологий программирования.

Выпускная квалификационная работа по направлению 010900 «Прикладные математика и информатика»

Разработка расширенного интерфейса пользователя к базе пространственных структур РНК

Работу выполнил:
Борисов Артём Александрович _____
Научный руководитель:
Доктор физико-математических наук, Доцент
Ройтберг Михаил Абрамович _____

Москва
2016

Оглавление

Введение

Рибонуклеиновая кислота (РНК) является, наряду с дезоксирибонуклеиновой кислотой (ДНК) и белками, одной из важнейших биологических макромолекул в живых организмах. Долгое время считалось, что единственной функцией РНК является обеспечение синтеза белков. В этом процессе задействованы три типа молекул РНК: матричная РНК, транспортная РНК и рибосомальная РНК. Матричная РНК является копией, снятой с молекулы дезоксирибонуклеиновой кислоты (ДНК), чтобы потом служить матрицей для синтеза определённого белка. Рибосомальная РНК вместе с группой определенных белков образуют комплекс, называемый рибосомой. Именно в рибосоме происходит процесс построения молекулы белка, называемый трансляцией. Транспортная РНК в этом процессе осуществляет доставку необходимых аминокислот, соответствующих триплетам нуклеотидов на матричной РНК (кодонам).

Недавние исследования показали, что существует множество так называемых некодирующих РНК, выполняющих ряд важных функций в организме, например, участие в регуляции экспрессии генов. Функция, которую выполняет молекула РНК, напрямую связана со структурой и её стабильностью. Такая связь открывает потенциальные возможности создания терапевтических стратегий, основанных на РНК, для чего необходимо детально понимать устройство её структуры и уметь её предсказывать. К сожалению, на данный момент РНК является наименее изученным классом биополимеров.

Одним из важных направлений в исследовании структур РНК является разработка, поддержка и развитие баз данных аннотированных пространственных структур, полученных с помощью экспериментальных техник, таких как, например, рентгеновская кристаллография, электронная микроскопия, ядерный магнитный резонанс и другие. На данный момент существует около 15 таких баз данных, которые в основном используются для обучения алгоритмов предсказания структур РНК, основанных на knowledge-based методах. Указанные базы данных содержат аннотации различных структурных мотивов, включая

элементы вторичной структуры РНК, РНК-белковые и РНК-ионные контакты и др, и отличаются друг от друга различным охватом структур (например, наличием или отсутствием РНК-белковых комплексов), сложностью рассматриваемых структур (например, наличием или отсутствием псевдоузловых структур РНК), возможностями поиска и другими важными параметрами. Наиболее популярные базы данных будут рассмотрены и проанализированы в части «Обзор литературы».

Несмотря на существование различных баз данных пространственных структур РНК, их потенциал в плане проведения именно биологических исследований остаётся нереализованным. Причиной этого является недостаточная функциональность веб-интерфейсов, созданных с целью обеспечения пользователей возможностями работы с базами данных. Зачастую их функциональность ограничивается работой с отдельными структурами и формированию элементарных запросов к множеству структур.

Целью данной работы является разработка такого веб-интерфейса, который обеспечит пользователям возможность в режиме онлайн формировать сложные многокритериальные запросы к базе данных, выводить различные графики и экспортировать полученные результаты. Для выполнения данной цели были поставлены следующие задачи:

- Разработка архитектуры веб-интерфейса. Основные требования:
    - Универсальность (возможность работы на всех уровнях структурной организации молекул РНК);
    - Масштабируемость (возможность добавлять новые структурные мотивы, дополнительные параметры, а также связи между элементами);
- Написание прототипа веб-интерфейса на основе разработанной архитектуры (реализация тестовой версии веб-интерфейса, охватывающего небольшое число структурных элементов; доработка архитектуры по результатам тестирования);

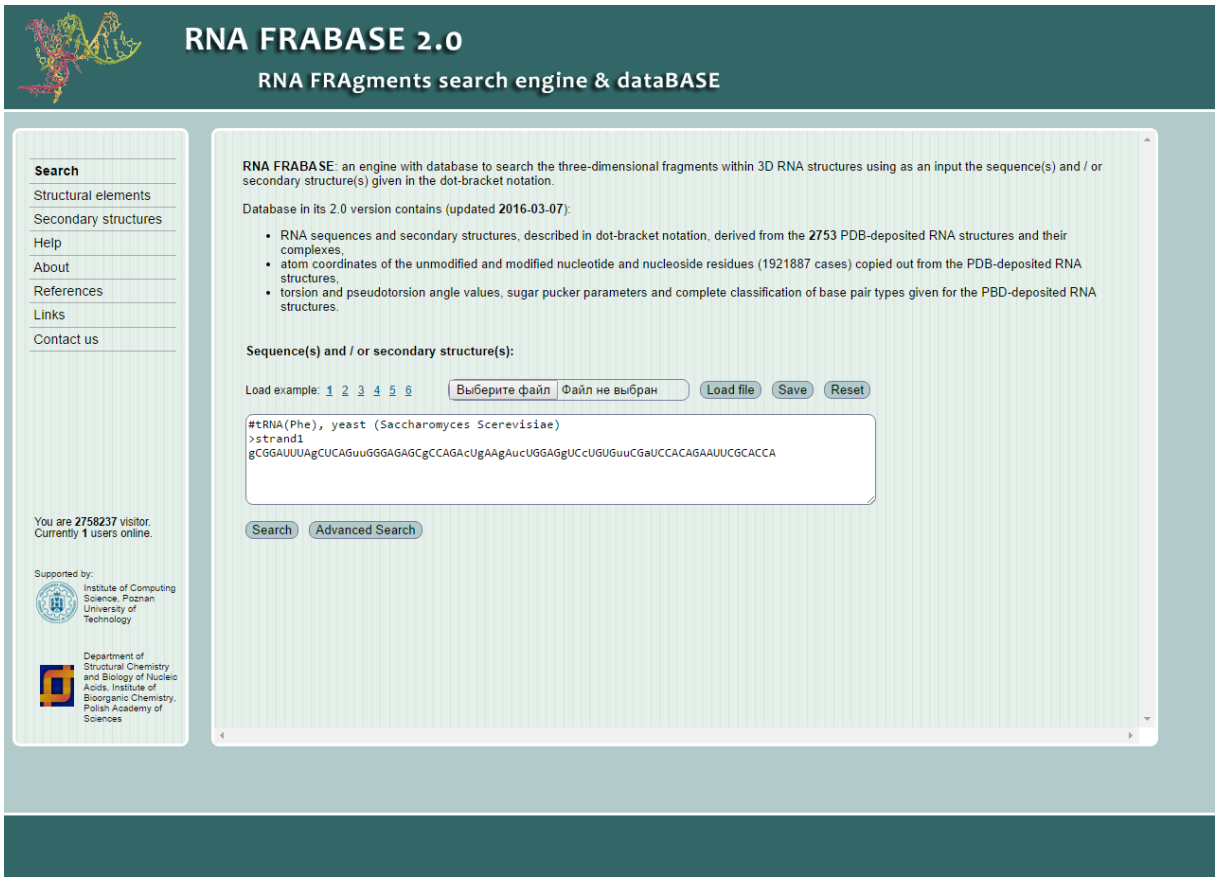
- Интеграция разработанной архитектуры в веб-интерфейс базы данных URSDB

Глава 1. Обзор литературы

Базы данных

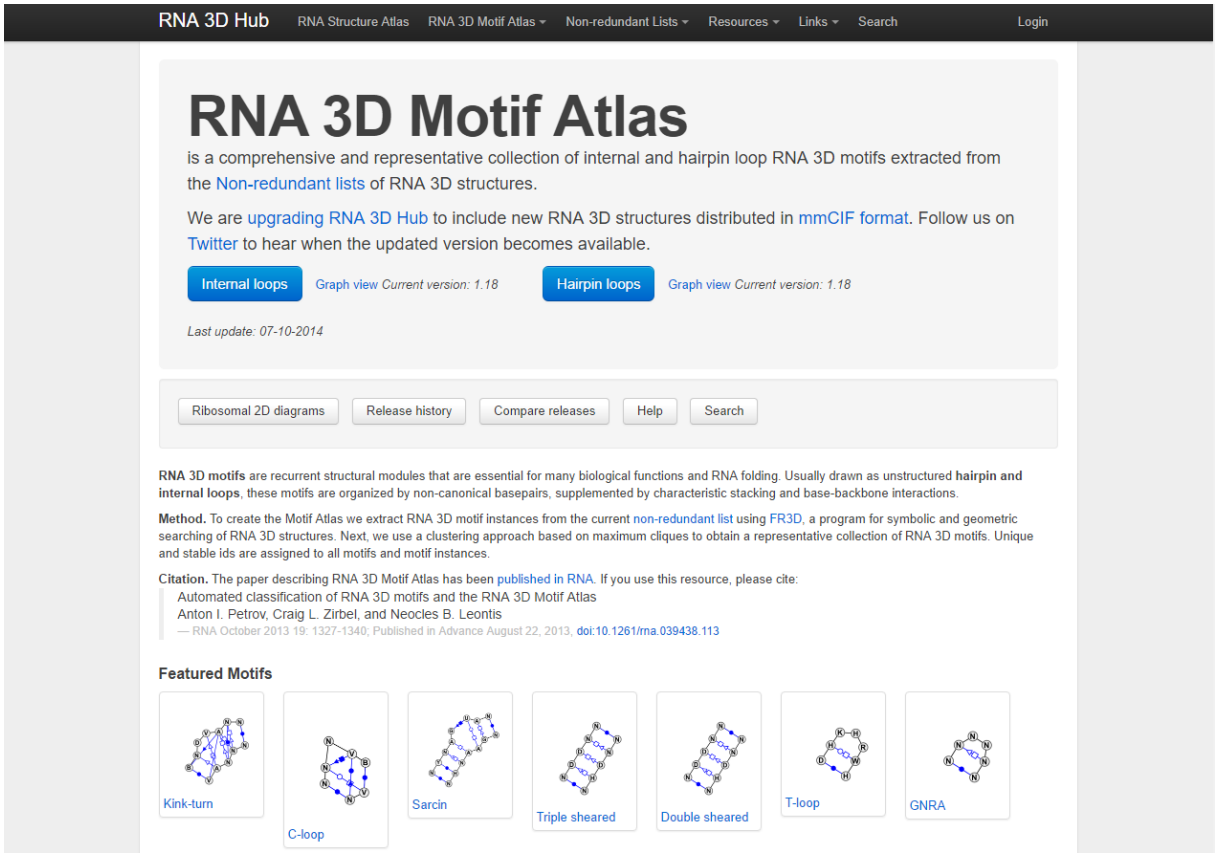
Среди предыдущих работ в данной области лучшими по качеству исполнения и доступности материала являются 9 рассмотренных ниже существующих проектов:

В качестве первой можно рассмотреть RNA Frabase 2.0,



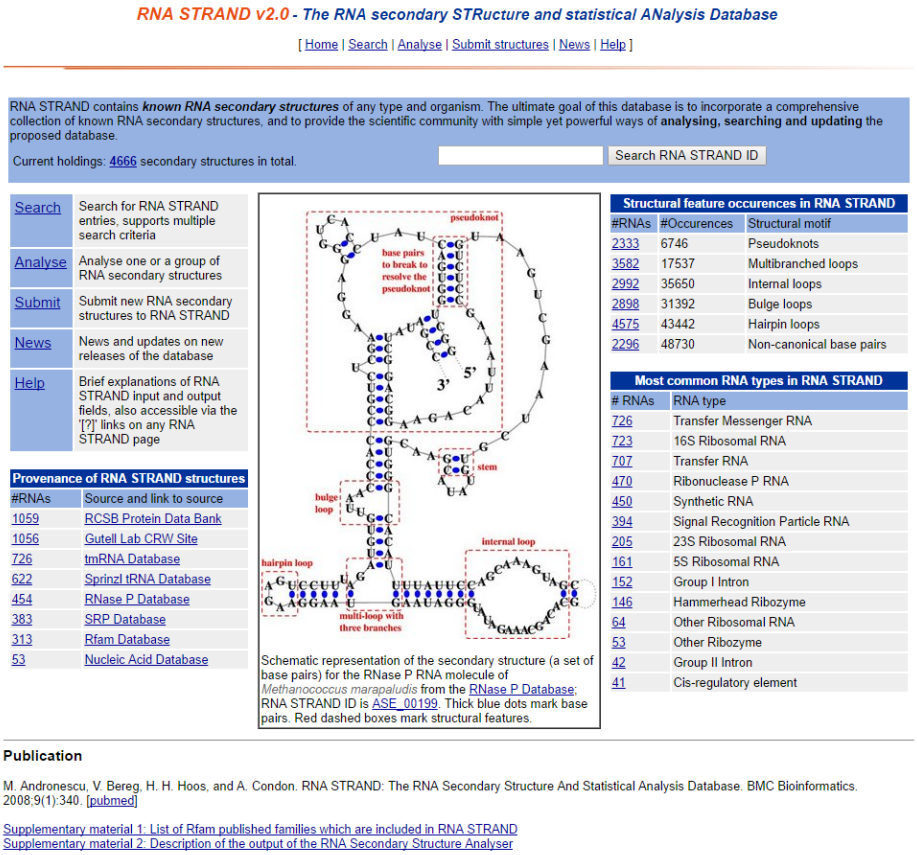
которая хранит вторичные структуры РНК, в частности их точечно-скобочную запись, и их структурные элементы. Так же в ней можно найти информацию о углах поворотов и псевдоскручиваний.

RNA 3D Motif Atlas



обеспечивает детальной информацией о трехмерных мотивах РНК и их компонентах (парах оснований, стекинг оснований, взаимодействия оснований с фосфатами) Все трехмерные мотивы кластеризованы и данные, проверяемые вручную. База Нуклеиновых Кислот предоставляет поисковый движок с широким набором параметров. Впрочем, при всей функциональности база как продукт достаточно сырая и периодически выдает ошибки PHP по многим запросам.

RNA Strand хранит вторичные структуры РНК включая те, что с неизвестной трехмерной структурой.



Также он позволяет пользователю искать с варьированием различных свойств вторичных структур РНК и элементов вторичных структур, включая псевдоузлы, в результатах поиска. Вдобавок предоставляется возможность вывода статистики запрошенных структур.

RNA Bricks хранит трехмерные мотивы РНК и предоставляет информацию о локальном контексте собранных мотивов, включая связи с белками и ионами металлов. Также в нем содержатся данные о связях между симметричными копиями одной молекулы в кристаллах и между молекулами из различных элементов PDB. На данный момент стала доступна улучшенная версия RNA Bricks2 с лучшей обработкой трехмерных структур РНК, доступными для загрузки двухмерными структурами РНК в формате точечно-скобочной записи.

RNA Bricks2

Home | Browse | Search | Statistics | Help

About RNA Bricks

RNA Bricks is a database of RNA 3D structure motifs and their contacts, both with themselves and with proteins. The database provides structure-quality score annotations and tools for the RNA 3D structure search and comparison.

Currently you can find here over 2400 RNA structures and RNA-protein complexes from the PDB (see statistics for details). We hope that your favorite ones are in. If not, keep visiting us - the database is updated weekly.

Authors and contact

RNA Bricks database was developed by Grzegorz Chojnowski, Tomek Waleń, and Janusz Bujnicki. Feel free to contact us at gcho...@genesilico.pl with any comments, suggestions or bug-reports.

If you find this tool useful please cite:
Chojnowski G, Waleń T, and Bujnicki JM **RNA Bricks—a database of RNA 3D motifs and their interactions**
Nucl. Acids Res. (2014) 42 (D1): D123-D131. doi: 10.1093/nar/gkt1084 [pdf]

RNA 2D structures were annotated using ClaRNA

^ back to top

Engine version:

News

17 Jun 2016 The database has been updated with 2 new RNA structures

10 Jun 2016 The database has been updated with 3 new RNA structures

03 Jun 2016 The database has been updated with 11 new RNA structures

28 May 2016 The database has been updated with 18 new RNA structures

19 May 2016 The database has been updated with 5 new RNA structures

13 May 2016 The database has been updated with 2 new RNA structures

RNA Bricks are maintained by the Laboratory of Bioinformatics and Protein Engineering at

International Institute of Molecular and Cell Biology in Warsaw

RNA CoSSMos хранит мотивы вторичных структур,

Dark/Light

Register | Login

RNA CoSSMos
Characterization of Secondary Structure Motifs

Home | Search | FAQ | Contact

Znosko Lab CoSSMos Database

Welcome to the Znosko Lab RNA Characterization of Secondary Structure Motifs (RNA CoSSMos) database! This tool allows the systematic searching of all catalogued three-dimensional nucleic acid PDB structures that contain secondary structure motifs such as mismatches, (a)symmetric internal loops, hairpin loops, and bulge loops.

To utilize the search function of the database, please select "Search" at the top menu.

If you have any questions, please check out our Wiki here or e-mail us via the Contact Us page. Also, please consider registering with the CoSSMos database if this is your first visit.

This site functions best at resolutions of at least 1440x900 and is fully compatible with these and later versioned browsers: Internet Explorer 8, Firefox 3.6, Safari XX, and Google Chrome.

Lastly, users of the RNA CoSSMos database should cite:

Vanegas, P. L., Hudson, G. A., Davis, A. R., Kelly, S. C., Kirkpatrick, C. C., and Znosko, B. M. (2012) RNA CoSSMos: Characterization of Secondary Structure Motifs- a searchable database of secondary structure motifs in RNA three-dimensional structures, *Nucleic Acids Res. 40*, D439-D444.

and are encouraged to cite:

Gendron, P., Lemieux, S., and Major, F. (2001) Quantitative analysis of nucleic acid three-dimensional structures., *J. Mol. Biol.* 308, 919-936.

Lemieux, S., and Major, F. (2002) RNA canonical and non-canonical base pairing types: a recognition method and complete repertoire., *Nucleic Acids Res.* 30, 4250-4263.

Search the RNA CoSSMos Database

Website design © 2013 Znosko Group

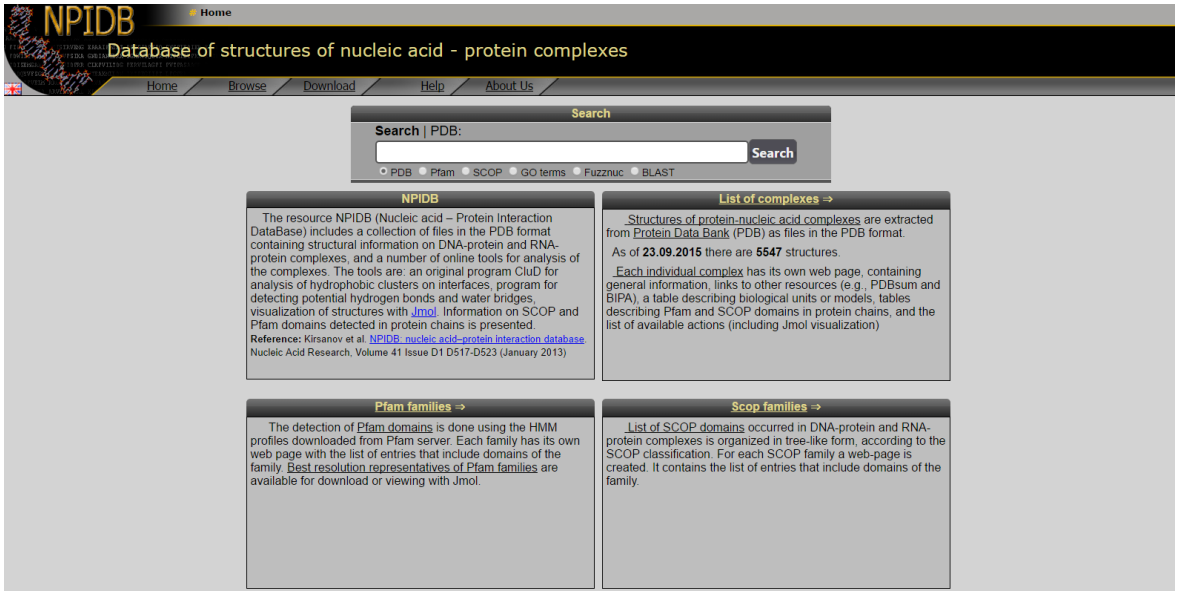
Funded by NIH Award 1R15GM085699-01A1

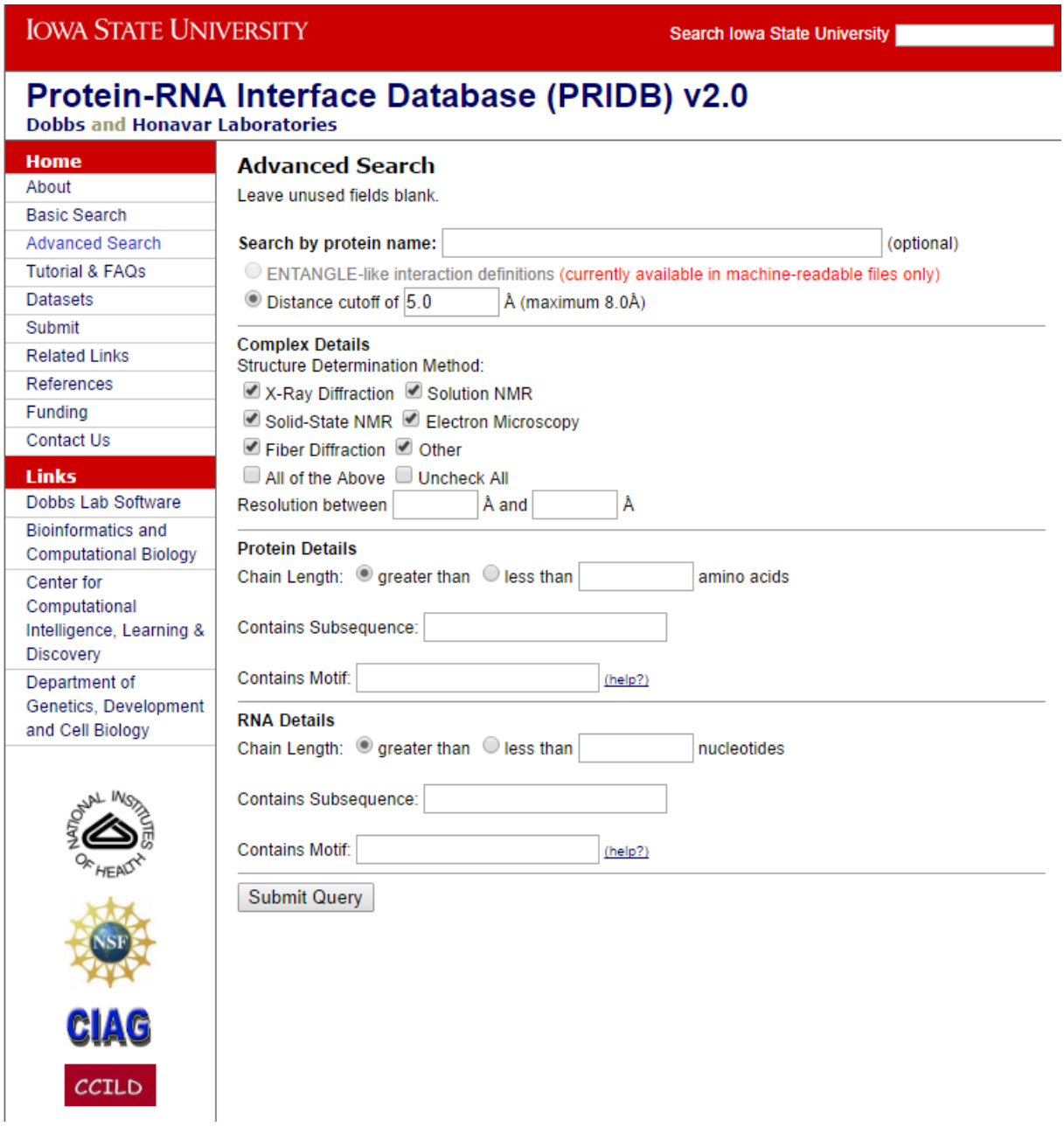
такие как несовпадения, внутренние петли, заколки и выпуклости, а также обеспечивает систематический поиск этих мотивов.

PseudoBase++ хранит различные псевдоузлы и предоставляет детальные описания. Также доступна поисковая система.

NPIDB и PRIDB хранят РНК-белковые взаимодействия со справками к вторичным

структурам белка.

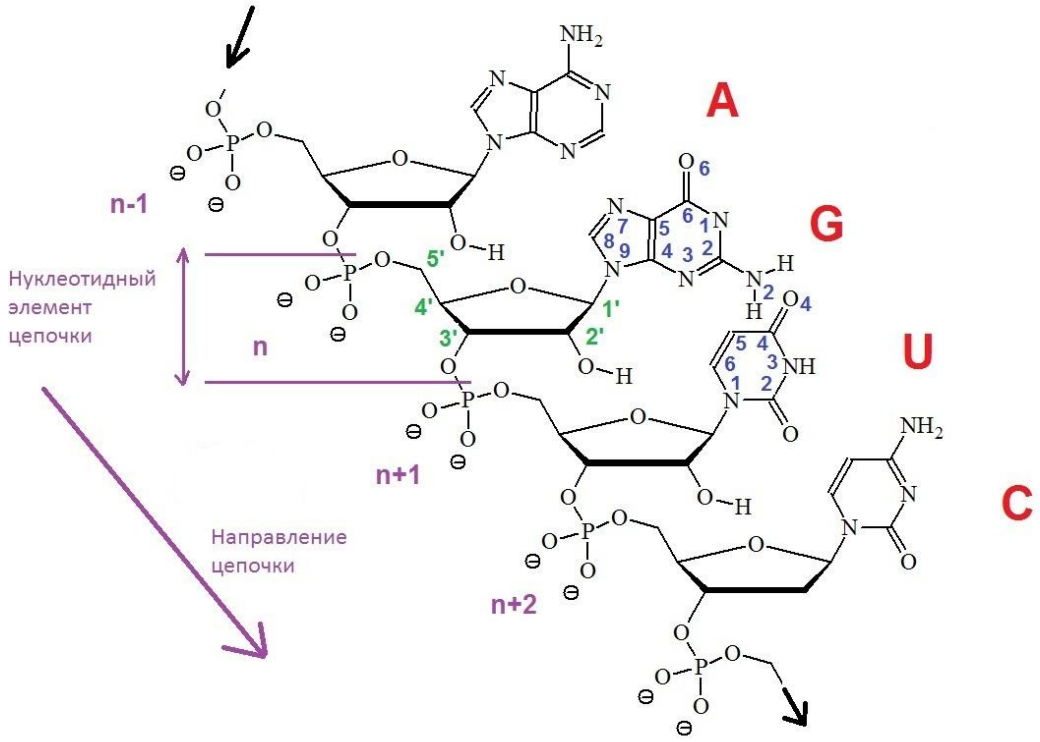




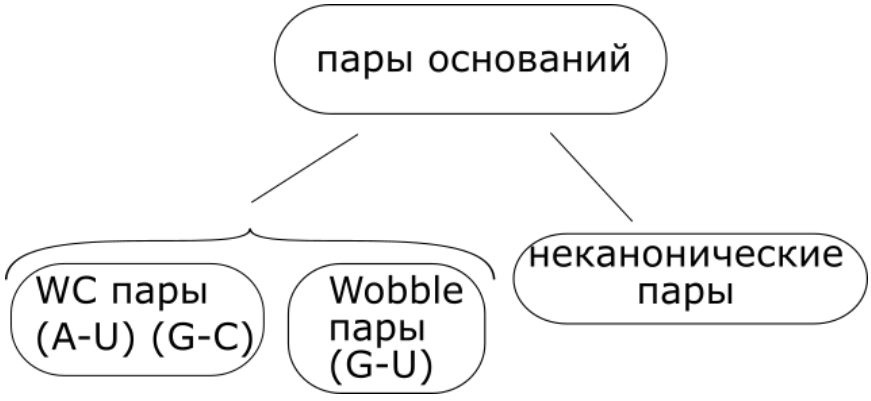
Несмотря на широкий спектр баз пространственных структур РНК к настоящему моменту, не существует базы данных содержащей все доступные трехмерные пространственные структуры РНК и в то же время предоставляющей возможность осуществлять поиск по структурам, учитывая только конкретные свойства каждой структуры и задавая их параметры. Большинство существующих баз данных ограничиваются структурами РНК без псевдоузлов.

Терминология

Рассматриваем РНК как последовательность нуклеотидов: Аденина, Цитозина, Гуанина и Урацила (далее будут обозначены A, C, G и U соответственно). Индексация идет в направлении 5’ – 3’ атомов углерода в рибозе.



Пара оснований(Base pair) – пара нуклеотидов (i, j) с $i < j$, которые формируют водородные связи.



Говорят, что пара оснований (p, q) - нулевого уровня, если она не имеет конфликтов ни с одной другой парой оснований (m, n) такой, что $m < p$. Все пары оснований структур без псевдоузлов являются парами оснований нулевого уровня.

Стем(Stem) – последовательность пар оснований формы $(i,j),(i+1,j-1),\ldots,(i+k,j-k)$ таких, что:

- $k \geq 1$
- $i + k < j + k$
- Все пары $(i+x, j-x)$, где $x = 0,\ldots,k$ образуют базовые пары и все они являются парами Уотсона-Крика(Watson-Crick) или Уобла(Wobble).

Пара (i,j) называется внешней парой или поверхностью стема и пара $(i+k, j-k)$ внутренней парой стема.

В стеме фрагмент $[i, i+k]$ называется левым крылом и фрагмент $[j-k, j]$ – правым крылом.

Нить(или неспаренный регион) – фрагмент $[i,j]$, такой что:

- Нет такой пары оснований (k,t), такой, что $i \leq k \leq$ о или $i \leq t \leq j$
- Один или оба нуклеотида $i-1,\ j+1$ содержится в паре оснований.

Глава 2. Результаты

Архитектура

По результатам анализа существующих баз данных и анализа требований, предъявляемых для обеспечения требуемых возможностей, была разработана следующая архитектура:

Есть множество типов сущностей. Между ними есть отношения принадлежности (одни сущности являются частью других). Причем если родительская сущность всегда единственна или отсутствует, то дочерних может быть множество. У каждой сущности есть определенный набор свойств.

На каждом шаге пользователь работает с определенным типом сущностей и множество всех сущностей данного типа. При этом пользователь имеет возможность:

- Ограничить набор сущностей данного типа, по заданным свойствам принадлежащих ему сущностей.
- Ограничить набор сущностей одного из дочерних типов только теми сущностями, которые бы наследовались от сущностей текущего типа, имеющих выбранные свойства.

- Вывести статистику по одному конкретному признаку.
- Вывести все сущности, удовлетворяющие заданным свойствам.

Примером неразветвленной структуры отношений типов может служить следующая последовательность:

$$\text{Нуклеотиды} \rightarrow \text{Пары оснований} \rightarrow \text{Стемы} \rightarrow \text{Цепи} \rightarrow \text{Структуры}$$

В качестве примера сеанса работы пользователя можно привести следующую последовательность:

- Выбор структуры по порогу разрешения
- Выбор среди них цепей РНК
- В цепях выбираются стемы по шаблону <*,WB,WB,*>
- Из отобранных стемов отбираем все Wobble пары оснований
- Из этих спариваний выбираем только пиримидины
- В выбранных пиримидинах смотрим распределение конформаций сахара.

Прототип

Входные данные.

В качестве входных данных используются .txt файлы следующей разметки:

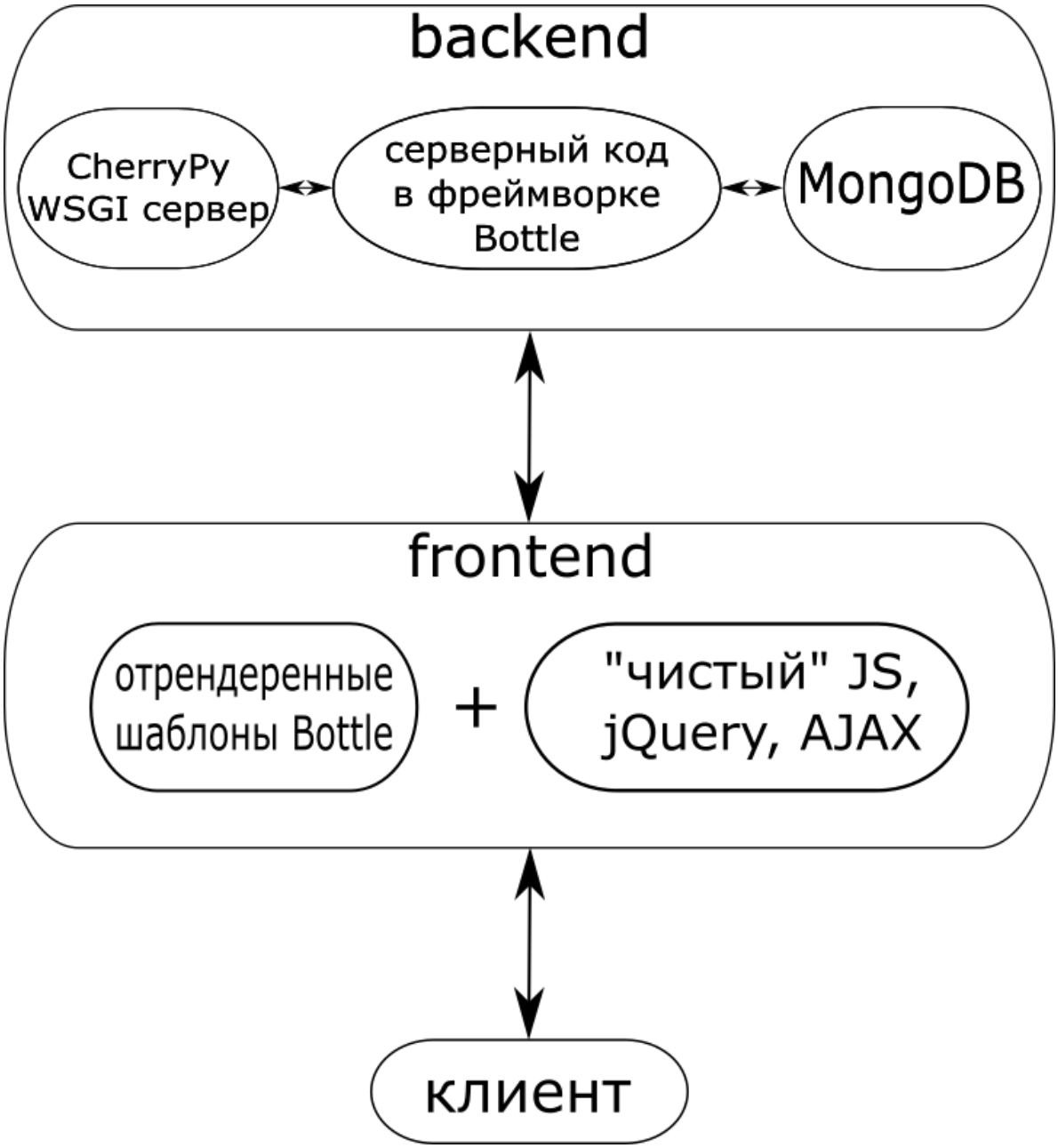
- Строка с именем типа объектов
- Строка с численными параметрами объектов
- Строка с нечисловыми параметрами объектов
- Строка с параметрами объектов предназначенных для построения гистограмм
- Строка с JSON объектом хранящим информацию о имени родительского типа, свойствах устанавливающих принадлежность объектов текущего типа объектам родительского типа, дочерних типах
- Строка с параметрами, не предназначенными для отображения и выдачи пользователю
- Строка с именами параметров объекта данного типа, в точности соответствующая идущим далее построчно свойствам объектов данного типа.

- Построчные (по строке на объект) значения параметров в соответствии с их последовательностью в строке из пункта выше.

При этом, оставляя заголовок из первых 7 пунктов, описывающих свойства типа и его объектов оставшуюся часть можно разбивать на несколько .txt файлов, при этом в результате заполнения базы данных будет занесена абсолютно вся предоставленная информация о каждом из описанных в указанных файлах объектах и типе.

Структура

Структурно можно представить проект следующей схемой:



Хранение данных

В качестве базы данных с элементами как таковой мною была выбрана MongoDB, предоставляющая:

- Выразительный язык запросов и вторичные индексы (манипулирование данными в сложных отношениях “из коробки”).
- Высокую согласованность данных (позволяет обеспечить пользователя новейшими копиями данных).
- Гибкость (модель данных, которая вмещает различные типы данных, доминирующих современных приложений).

- Удобное управление и интеграцию (база данных может быть: защищена, находиться под постоянным мониторингом, автоматизирована и интегрирована в существующую инфраструктуру).
- Масштабирование и производительность.

Особенно важна её(базы данных) документоориентированность: каждый объект каждого типа, не важно стем это или единичный нуклеотид являет собой объект. При этом отпадает необходимость хранить много таблиц со свойствами(в случае реляционной базы данных это потребовало бы минимум создания количества таблиц, равного суммарному количеству свойств объектов всех типов). Каждый документ представляется JSON-объектом, благодаря удобному интерфейсу драйвера PyMongo каждый вытащенный и базы данных объект будет являть собой питоновскую структуру. При всем этом удобстве снаружи скорость работы с данными и производительность в общем нисколько не пострадали: для эффективного использования памяти все структуры хранятся в BSON(Binary javaScript Object Notation) в виде Б-дерева используя именно те самые индексы. Сейчас в базе данных в коллекции объектов используется составной индекс из типа и идентификатора вместо дефолтного _id. То есть все объекты на данный момент хранятся в коллекции objects, а описание же всех их свойств, описание соотношений между типами и прочая глобальная информация хранится в коллекции types. Со временем планируется смена данной структуры на отдельные коллекции для каждого типа объектов. На данный момент примером документа, описывающего тип, может служить следующая запись:

{"parentprops": {"model": "model", "dssr": ["nucl1", "nucl2"]}, "name": "nucleotide", "props": {"miss": {"min": 0, "max": 1}, "res": ["G", "A", "U", "C", "DC", "DA", "DT", "DG", "4SU", "5MC", "5MU", "PSU", "GDP"], "dssr": [], "bps": {"min": 0, "max": 5}, "slbracket": ["(", ".", ")", "[", "]", "\\N", "-", "{", "}", "<", "!", ">", "?"], "bracket": ["(", ".", ")", "[", "]", "\\N", "-", "{", "<", "}", ">"], "fstems": {"min": 0, "max": 3}}, "_id": {"$oid": "5735d547cf56e6fd3fa230eb"}, "hist": ["bps", "miss", "res", "bracket", "slbracket"], "length": {"oldwing": 8, "cifid": 5, "bracket": 3, "diagram": 7, "place": 3, "res": 5, "mult": 8, "num": 5, "model": 6, "letter": 3, "miss": 3, "lumult": 8,

"chnumindiag": 3, "dssr": 12, "wing": 8, "bps": 3, "slbracket": 3, "fstems": 3, "zip": 7, "chain": 7, "thread": 8}, "child": [], "parent": "basepair"}

И, соответственно пример объекта данного типа:

{"oldwing": 367357, "slbracket": "(", "chnumindiag": 1, "diagram": 11118, "num": 1, "type": "nucleotide", "thread": null, "bracket": "(", "cifid": 1, "lumult": null, "zip": null, "id": 3303889, "wing": 389651, "mult": null, "fstems": 1, "_id": {"$oid": "57559fe0db20e91234ab7eed"}, "model": 9934, "dssr": "A.G.1.", "letter": "A", "miss": 0, "place": 0, "res": "G", "bps": 1, "chain": 55707}

В качестве механизма хранения данных выбран WiredTiger. Это сторонний механизм хранения позволяющий сжимать данные. Среди прочих достоинствWiredTiger следует отметить блокировки на уровне документа (в дефолтном MMAPv1 идут блокировки на уровне коллекций) и чекпоинты каждую секунду, что исключает необходимость в журналировании.

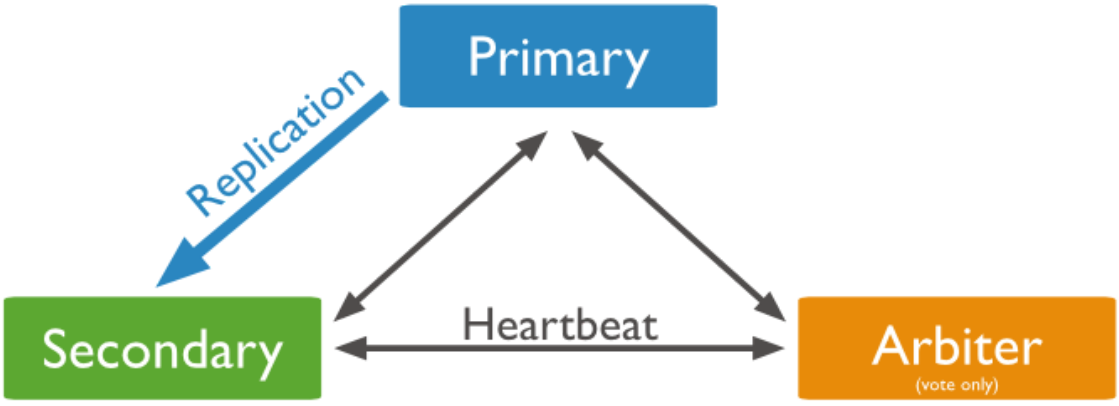
Под данные MongoDB выделяет память в последовательности 64Mb->128Mb->256Mb->512Mb->1024Mb->2048Mb->…->2048Mb, причем последний выделенный чанк всегда пустой. Иными словами для хранения 200Mb данных будет выделено 960Mb. Еще один варьируемый параметр в плане сильного перерасходования теперь уже оперативной памяти – кэш WiredTiger.

По этому для тестовых запусков базы данных флаги были следующими:

mongod --port27001 --smallfiles --nojournal –dbpath *путькdb* --logpath *путьклогам* --storageEngine wiredTiger --wiredTigerCacheSizeGB 2

Для стабильности данной системы была настроена репликация, сет реплик состоит из трех серверов: primary нода, secondary нода и арбитр. Арбитр не несет какой-либо информации из хранящихся данных и нужен здесь исключительно для успешного завершения election на случай, если по каким-либо причинам

становится недоступной primary нода.



Итоговая конфигурация серверов запускалась так:

- mongod --port 27003 --dbpath /data/arb --replSet bioreplica
- mongod —port 27001 —replSet bioreplica —dbpath /data/primary —logpath /logpath/log1 —logappend —oplogSize 50 —smallfiles
- mongod —port 27002 —replSet bioreplica —dbpath /data/secondary —logpath /logpath/log2 —logappend —oplogSize 50 —smallfiles

Далее сет реплик настраивался из главной ноды:

- mongo –port 27001

И теперь уже в консоли primary ноды:

- ~ rs.initiate()
- ~rs.add(‘*host*:27002’)
- ~rs.addArbiter(‘*host*:27003’)

Таким образом, в случае отказа вторичной ноды или арбитра не произойдет ничего, а в случае отказа primary ноды пройдут «выборы» новой главной ноды и единственным возможным результатом их будет выдача вторичной ноде статуса primary ноды.

Веб-сервер

В качестве языка программирования, используемого для написания backend-части проекта, был выбран язык программирования Python3(3.4) ввиду его читабельности и относительно высокой скорости разработки сложных проектов. В качестве основного фреймворка, на котором строится проект, используется Bottle его стабильная версия Bottle 0.12. Причин использования данного фреймворка несколько, среди них:

- Минимальные зависимости.
- Минималистичность самого питоновского модуля(весь фреймворк умещен в один файл Bottle.py).
- Роутинг в bottle осуществляется с помощью декораторов.
- Возможность задания динамических роутов регулярными выражениями.
- Встроенный шаблонизатор. Причем имеется возможность писать питоновский код в шаблонах. Более того, интерполяция html код по умолчанию блокируется во избежание XSS атак.
- Полноценный POST-роутинг и обработка форм.
- Более-менее простой механизм работы с Cookies.
- Наличие простого сервера для быстрой отладки.

Нельзя недооценивать последний пункт. Но как и было сказано, внутренний сервер Bottle годится только для отладки. В Python предусмотрен такой стандарт как WSGI (Web Server Gateway Inteface), предоставляющий унифицированную модель для взаимодействия между Python-программой, выполняющейся на стороне web-сервера и самим сервером. Благодаря WSGI становится возможным поставить работу всего backend кода на сервер от стороннего фреймворка или даже Apache/Nginx.

Ввиду наличия достаточной сложности и требования глубоких профессиональных познаний в настройке Apache/Nginx, несмотря на их, лучшую в своем классе, производительность, мною был выбран WSGI сервер от фреймворка CherryPy. Данный сервер является очень стабильным и активно

использует многопоточность. Огромную разницу в работе между выбранным сервером и сервером по умолчанию легко увидеть заменив запуск проекта с

```
run(server='cherrypy', host='localhost', port = 8080)
```

на

```
run(host='localhost', port = 8080)
```

В дальнейшем планируется подключение nginx в связи с его, лучшей в классе, работой с коннектами пользователей.

Код серверной части

Как уже было сказано выше, весь серверный код написан в фреймворке Bottle. По сути, нужно было обеспечить url-ами каждый информативный запрос от клиента: будь то новая страница или AJAX запрос. В работе активно использовались как POST так и GET запросы со стороны клиента.

В основном все роуты были статичными, за исключением путей для javascript кода и подключаемых javascript модулей. Декоратор для всех этих путей был сделан на регулярных выражениях и работает динамически.

Обработчики остальных url статичны. Из них GET запросами получаются:

- '/' - корневой путь
- '/middle_choice' – путь для перехода от родительского типа к дочернему
- '/export_props' – путь для задания параметров экспорта
- '/favicon.ico' – путь для получения браузером favicon

POST запросы для путей:

- '/choosin' – путь для изменения свойств нужных объектов данного типа
- '/datagathering' – путь получения статистики для построения гистограммы
- '/export' – путь с предоставляемыми, по итогам работы с сайтом за сеанс, данными

Кроме того в ходе работы были написаны обработчики наиболее часто встречающихся ошибок: соответственно 404 и 405 ошибки. Первая появляется при обращении к несуществующему url сайта, вторая же при неправильном типе запроса к конкретному url(например посылая GET вместо POST на '/choosin').

Почти каждый метод срабатывающий по определённому url в результате выдает свой уникальный отрендеренный темплейт. «Почти» - потому, что:

- по сути переход к дочернему типу равно как и работа с текущим типом одинаково подгружают один темплейт с выбором свойств типа, с которым работаем в новый момент времени.
- Экспорт в html-таблицу в виде html кода генерируется прямо в главном скрипте проекта.

Для отсутствия повторений больших кусков кода, в проекте я максимально старался объединить их в функции. Однако это вызвало определенные трудности. Оказалось что, только что установленная путем response.set_cookie('cookiename','cookievalue') cookie, на самом деле моментально не изменилась. И дальнейшие попытки её извлечь, путем request.get_cookie('cookiename') в течение обработки всё того же запроса, приведут либо к получению пустой строки '', либо к получению предыдущего значению cookie 'cookiename'. Этот факт явно отражался на достоверности выдачи данных сайтом. После исследования кода самого фреймворка, я нашел вот такой кусочек кода:

```
class HTTPResponse(Response, BottleException):
    def __init__(self, body='', status=None, headers=None,
**more_headers):
        super(HTTPResponse, self).__init__(body, status,
headers, **more_headers)

    def apply(self, response):
        response._status_code = self._status_code
```

```
        response._status_line = self._status_line
        response._headers = self._headers
        response._cookies = self._cookies
        response.body = self.body
```

Как выяснилось, все изменения текущего(self) объекта класса HTTPResponse применяются к response только по завершении обработки роута. Единственным выходом был найден и реализован механизм передачи сложных структур от каждого из методов, обеспечивающих разные роуты, но посредством одной функции выдававших отрендеренную страницу шаблона с правильными данными в эту самую функцию.

Все темплейты имеют формат .tpl и хранятся в ./views/ относительно главного исполняемого файла.

Код клиентской части

Поначалу в клиентской части использовались только отрендеренные шаблоны Bottle, приходившие по заранее заданным путям плюс картинка для браузера по /favicon.ico. Но с расширением функционала оказалось просто невозможно все делегировать для выполнению серверу. Так, например, выбранные поля для каждого из типов в конкретной выборке запоминаются на сервере в cookies. Размер cookie ограничен 4 килобайтами на всю пару ключ-значение. При появившейся необходимости запоминать предыдущие запросы пользователей для последующей возможности выбрать из ранее введенных наборов параметров к типам нагрузка на сервер возрастает. Более того, становится очевидной нецелесообразность такого подхода при наличии средств для складирования данных клиента в его же браузере, таких как LocalStorage и SessionStorage. К тому же размер одного значения имеет более чем в тысячу раз большее значение: 10 мегабайт.

В связи с использованием LocalStorage на стороне клиента возникла необходимость в использовании javascript. Для работы с полями форм использовалась библиотека jQuery. jQuery есть по сути своей библиотека javascript упрощающая взаимодействие javascript и HTML. Несмотря на предоставление API для работы с AJAX, данная возможность показалась мне реализованной не сильно проще или удобнее стандартного способа создания и обработки AJAX-запросов. А вот сильно упрощенное получение доступа к любому

элементу DOM(Document Object Model в HTML) сыграло огромную роль в реализации той части клиентского кода, что работает с формами на страницах.

jQuery можно было бы выдавать по url, как выдаются остальные javascript файлы, но сейчас я решил, использовать CDN (Content Delivery Network или Content Distribution Network) jQuery от Microsoft и в коде шаблона можно встретить:

```
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.12.2.min.js"></script>
```

По сути значения полей складываются в JSON-объект и далее размещаются в LocalStorage. Размещение происходит парами ключ-значение с типом в качестве ключа и остальными данными в виде словаря свойство-значение.

На странице выбора свойств ‘/choosin’ помимо отображения количества подходящих под заданные значения свойств объектов присутствует возможность построить гистограмму по этим самым удовлетворяющим заданным условиям сущностям.

Построение гистограммы реализовано при помощи Chart.js . Данная библиотека наделяет графики красивым плоским дизайном. Для рендеринга она использует HTML5 canvas. Конечно, всё это не работает, и не будет работать в старых браузерах вроде IntenetExplorer7/8. В Chart.js доступны самые разные типы графиков (core, столбчатый, кольцевой, лепестковый, линейчатый и полярный), мною был использован столбчатый (bar chart).

Сам chart.min.js файл с кодом возвращается по специальному роуту в ‘/static/’. Именно таким способом и планируется возвращать jQuery.

Для отрисовки гистограммы по специальной кнопке посылается POST запрос на сервер. Сервер обращается к базе данных и возвращает XML-документ, который на лету обрабатывается и преобразованный в JSON-объект выводится в canvas внизу страницы посредством Chart.js.

При проектировании и реализации клиентской части часть javascript кода была размещена на сервере, а часть предоставлена клиенту для скачивания со сторонних ресурсов. Если код, который написан в пределах данного проекта, предоставлять клиенту с сервера единственно логично, то с jQuery и Chart.js вопрос остается открытым:

Со временем следует изменить способ предоставления контента в зависимости от географического распределения клиентов. То есть сама технология CDN нужна, и имеет смысл лишь тогда, когда пинг до сервера приложения больше пинга до стороннего ресурса, размещающего .min.js файл с кодом. Предполагается, что со временем, когда станет ясно, откуда основная аудитория приложения можно

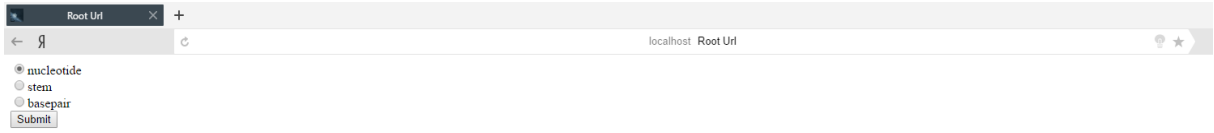
будет при помощи VPN подключиться оттуда к различным ресурсам и, замеряв значение, окончательно решить вопрос размещения javascript библиотек.

Причем указанный выше – не единственный нюанс. При затягивании библиотеки с некоторых ресурсов может оказаться, что некоторые пользователи не смогут работать с функционирующими на подключаемых библиотеках частями приложения вообще: так например некоторые компании блокируют адреса *.microsoft.com как часть процесса блокировки обновлений Windows. И хотя это абсолютно не правильно, факт остается фактом: такое сегодня встречается часто.

Другой проблемой может оказаться хранение сторонними ресурсами библиотеки с урезанным функционалом: к примеру, Google не предоставляет jQuery.Validate, а Microsoft – jQuery-UI.

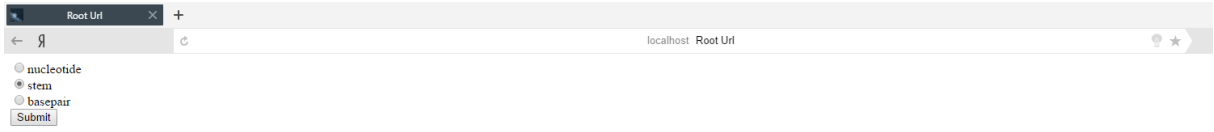
Пример пользовательского сеанса работы.

Первое, что видит пользователь, это заглавная страница с формой выбора типов реализованной в виде radio:

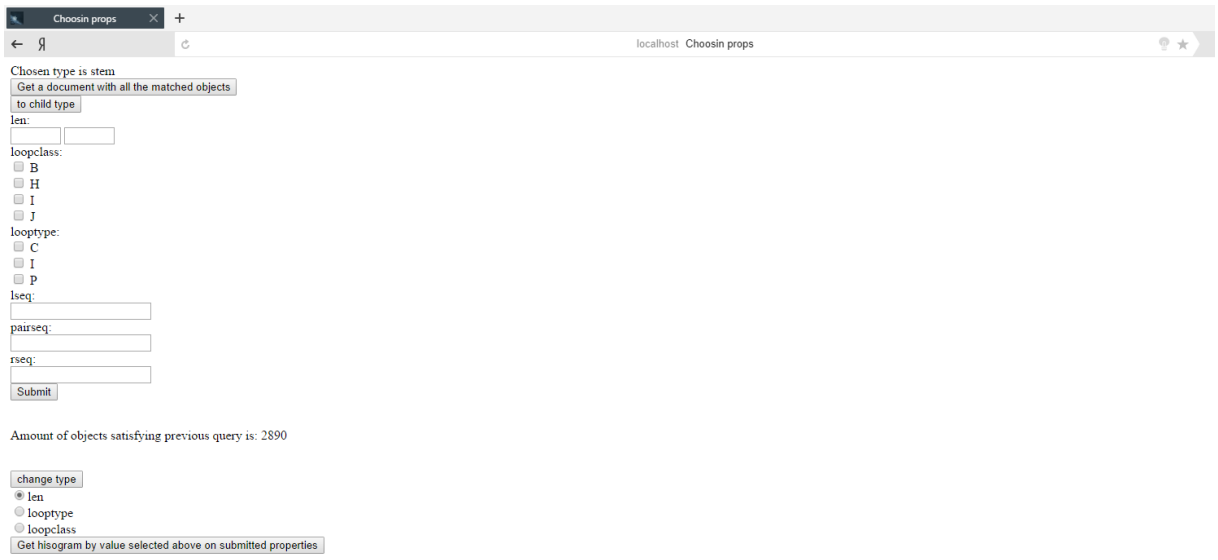


Как и во всех остальных местах использования формы данного типа везде маркер по умолчанию проставлен первый пункт.

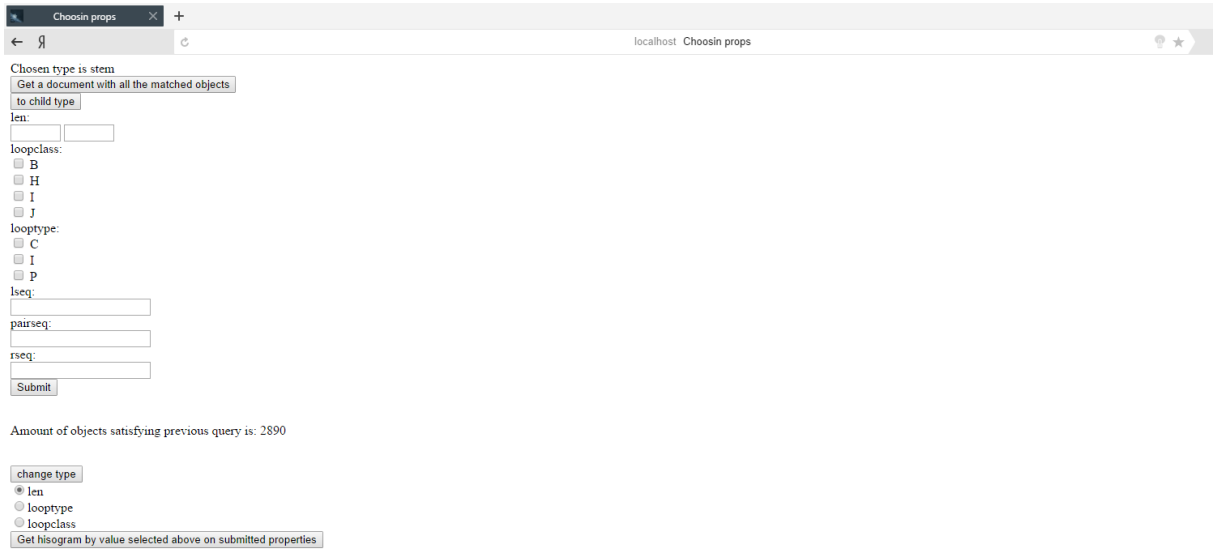
Для демонстрации всех возможностей приложения, включая структурную принадлежность одних типов другим, для наглядности выберем тип stem. Он является родительским для типа basepair:



Отправив данные формы, получаем самую тяжелую из страниц приложения:

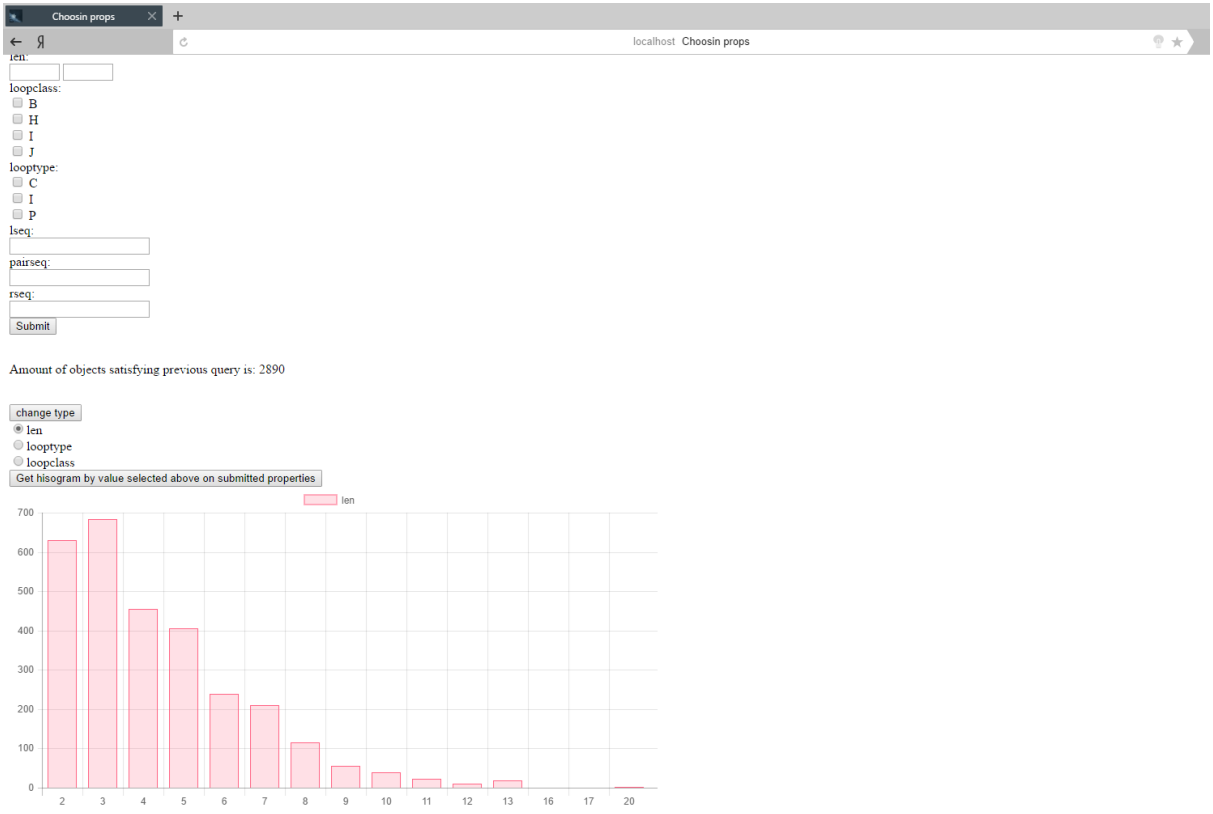


Для каждого типа на данной странице будут свои свойства для выбора. Числовые свойства представлены диапазонными формами, свойства имеющие лимитированное число значений, не превышающее 20, представлены формами типа checkbox и последний тип свойств – нечисловой и имеющий более 20 значений (вплоть до уникального значения для каждого из элементов) представлен обычными текстовыми формами:



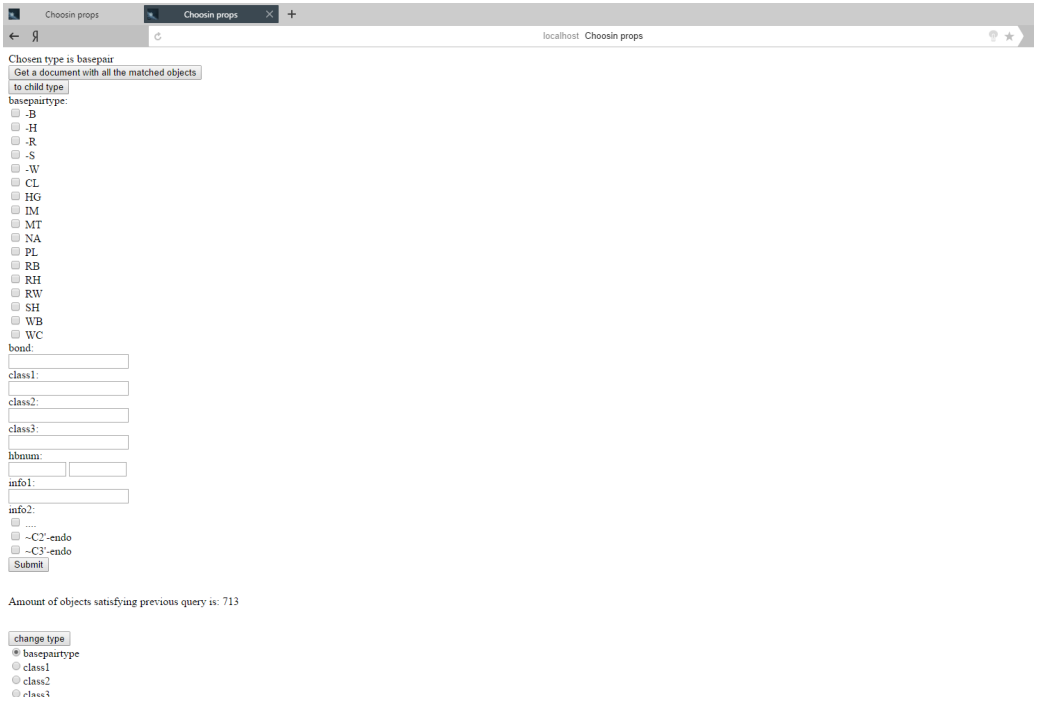
Так же на странице можно найти форму типа radio для выбора параметра и кнопку для построения гистограммы на объектах, удовлетворяющих выбранным свойствам, по заданному параметру. Предположим, что нас интересуют пары

оснований принадлежащие стемам верхней половины диапазона длин стемов. Для начала посмотрим, какую часть общего количества родительских объектов мы ограничим – строим гистограмму по длине стемов:



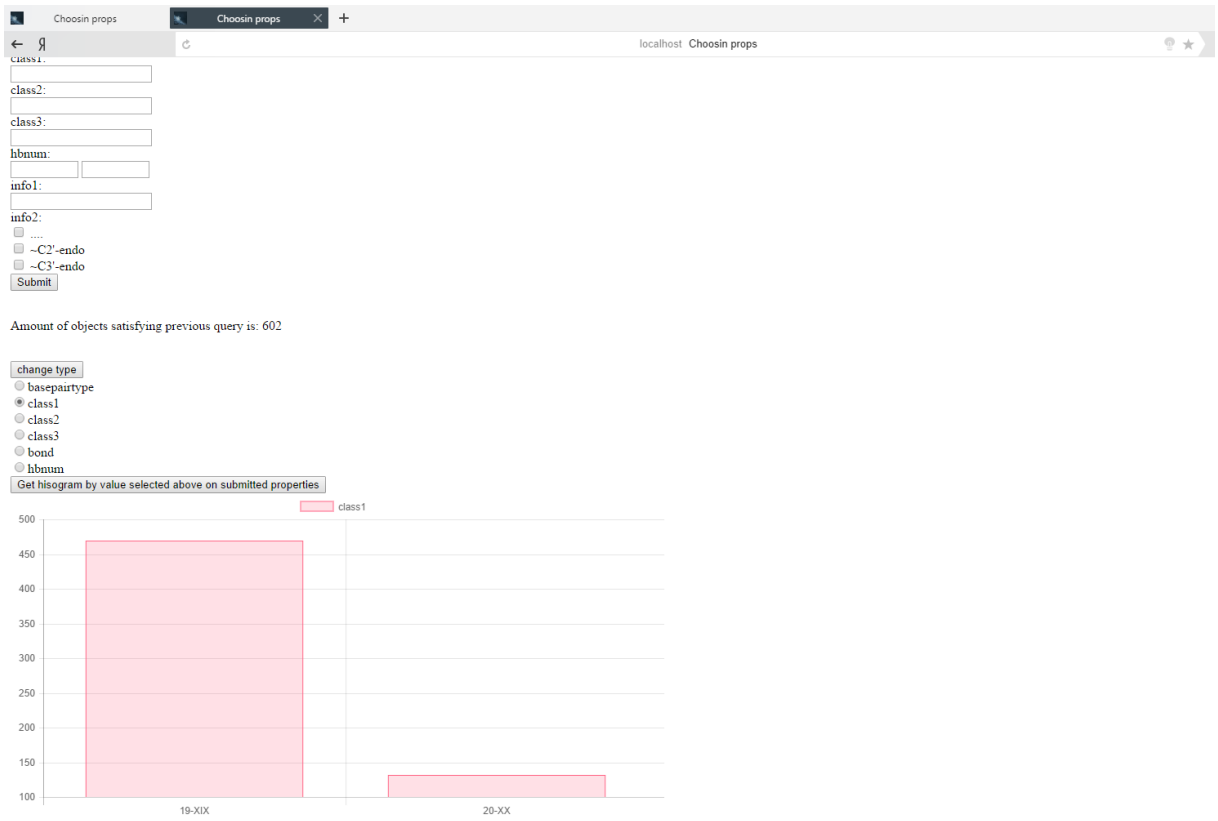
Оказалось, что основная часть стемов лежит вне выбранного диапазона. Проставим приемлемые границы длины стемов: от 11 до 20. Отправим формы и построим гистограмму заново:

Итак, всего нашему запросу удовлетворяет 57 стемов. Распределение стемов по длинам в заданном диапазоне мы тоже получили. Нажимаем «перейти к дочернему типу» и переходим к рассмотрению пар оснований, составляющих эти 57 стемов:

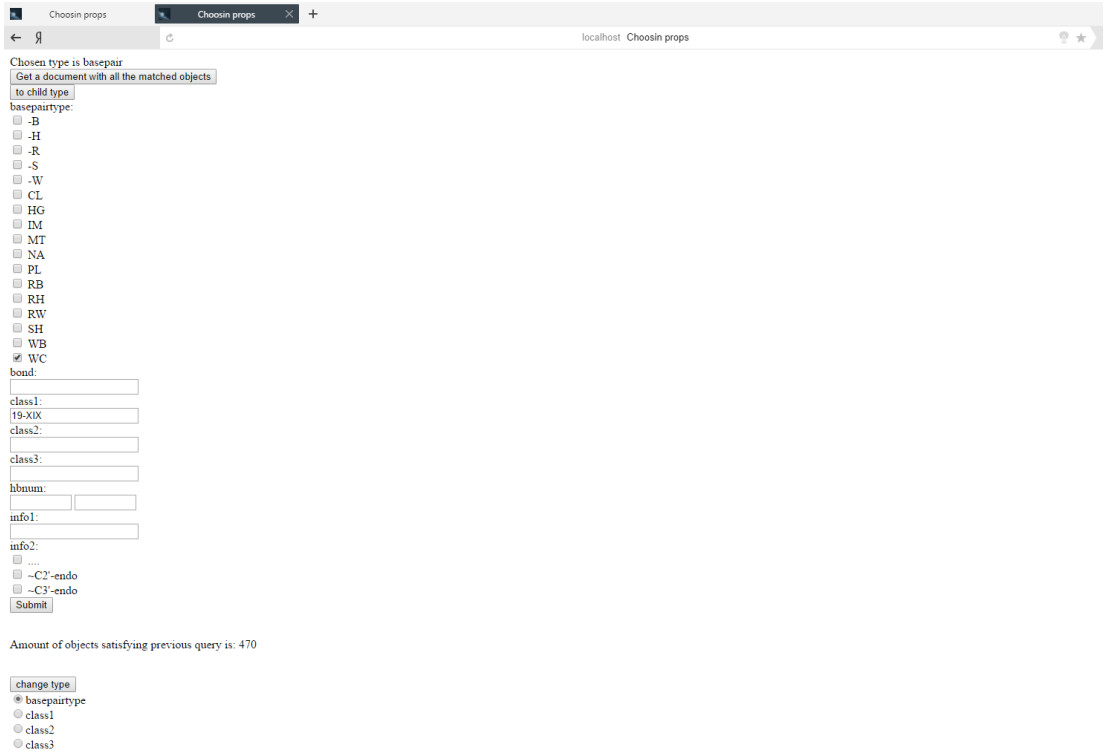


:

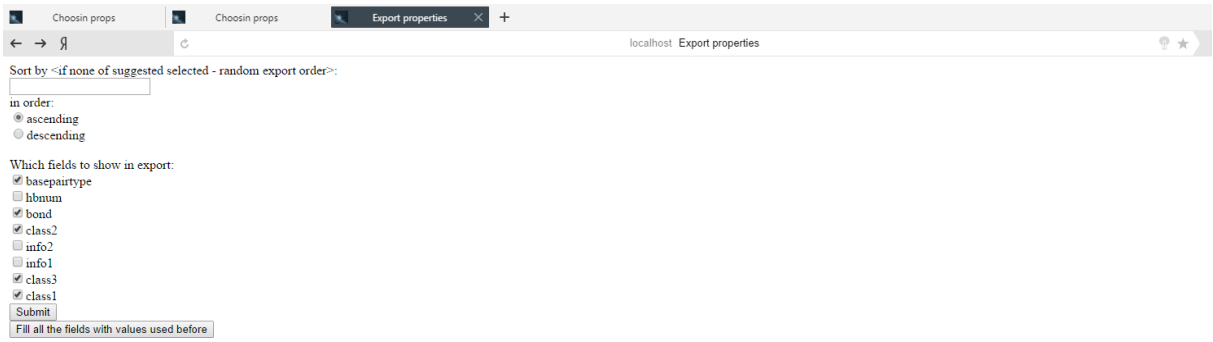
Видно, что эти 57 стемов образованы 713 парами оснований. Ограничим набор лишь парами оснований Уотсона-Крика: таковых нашлось только 602. По отобранным 602 парам оснований построим гистограмму по классам Сенгера:



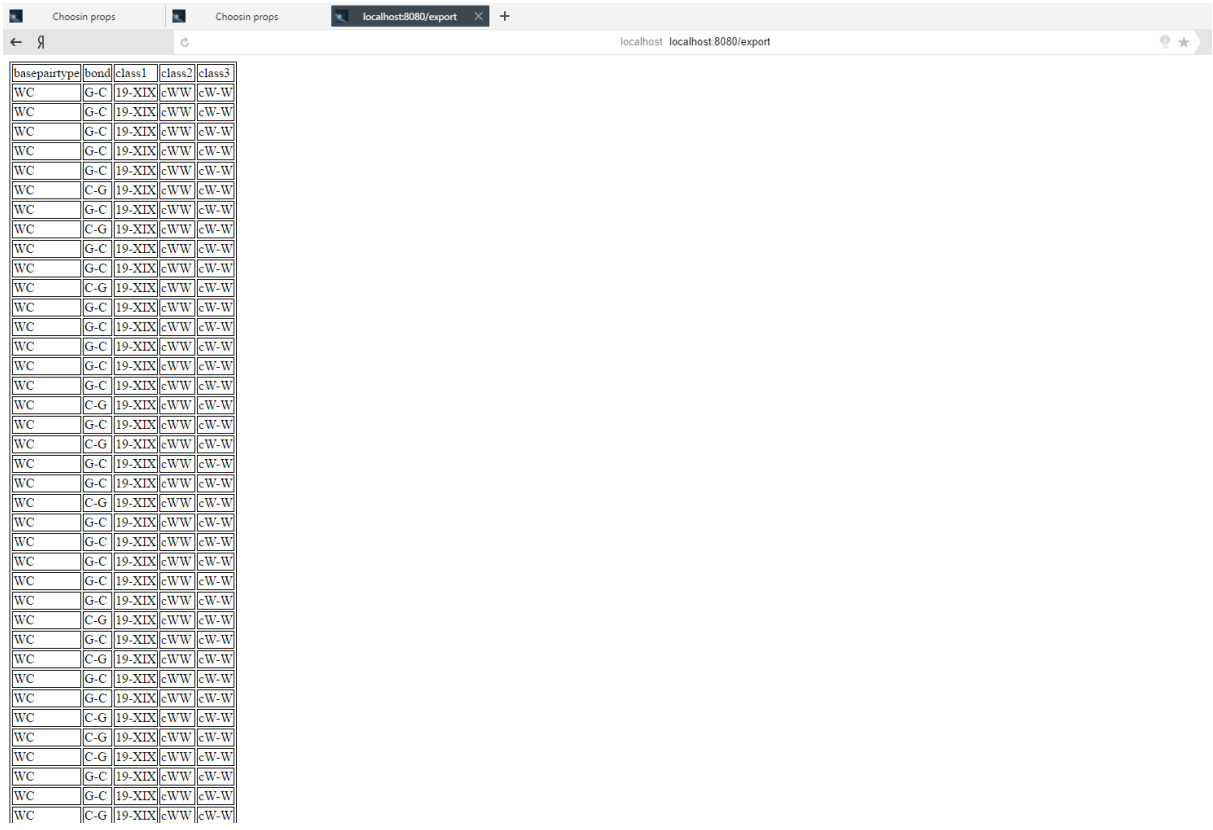
Выберем исключительно пары оснований класса 19-XIX:



Нажмем на получение данных. Выберем поля для вывода:



И получаем результат:



basepairtype	bond	class1	class2	class3
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	G-C	19-XIX	cWW	cW-W
WC	C-G	19-XIX	cWW	cW-W

Заключение

Поддержка и развитие баз данных пространственных структур биологических макромолекул, является одним из важных направлений в их исследовании, так как на основе этих данных происходит обучение machine-learning алгоритмов, применяющихся для предсказания структур РНК. Существующие базы таких данных являются узкоспециализированными и предоставляют крайне скудные средства для получения и, тем более, анализа данных. В результате выполнения данной работы была разработано архитектура и создан прототип решения, позволяющий хранить и систематизировать данные всех уровней структурной организации РНК. Он предоставляет возможности: добавления новых как

существующих элементов, так и новых типов, отвечающих ранее не занесенным структурным уровням РНК, получения выборки данных по широкому кругу свойств с учетом наследования типов, получения статистики – распределения по элементам, обладающими заданными свойствами. Благодаря использованию MongoDB и её объектно-ориентированному подходу удалось добиться абсолютно беспрепятственной и простой в настройке масштабируемости проекта.

Помимо того, что сделано, остались и нереализованные запланированные в прототипе функции, которые необходимо будет создать и встроить в него позднее, это:

- Автоматическое обновление содержания сущностей уже имеющихся в базе типов посредством затягивания новых структур через API PDB.
- Построение конъюнктивно-дизъюнктивных наборов условий при создании выборки.

Помимо этого будет правильным разделить в будущем все типы по разным коллекциям. В зависимости от нагрузки и доступного оборудования WSGI сервером сделать nginx.

После всех вышеперечисленных улучшений отдельные элементы данной конструкции будут встроены в URSDB, разработанную отечественными учёными из лаборатории прикладной математики института математических проблем биологии.

Список литературы.

1. Dahm R (2005). «Friedrich Miescher and the discovery of DNA». Developmental Biology: 274–88
2. Nierhaus KH, Wilson DN. Protein Synthesis and Ribosome Structure. — Wiley-VCH, 2004
3. Szathmáry E. The origin of the genetic code: amino acids as cofactors in an RNA world // Trends Genetics — 1999_ 223–9
4. Cooper GC, Hausman RE. The Cell: A Molecular Approach. — 3rd edition. — Sinauer.: 261–76, 297, 339–44
5. Higgs PG (2000). «RNA secondary structure: physical and computational aspects». Quarterly Reviews of Biophysics 33: 199–253.
6. Nissen P, Hansen J, Ban N, Moore PB, Steitz TA (2000). «The structural basis of ribosome activity in peptide bond synthesis». Science 289 (5481): 920–30.
7. Popenda, M. et al. (2010) RNA FRABASE 2.0: an advanced web-accessible database with the capacity to search the three-dimensional fragments within RNA structures. BMC bioinformatics, 11 (1), 231.

8. Petrov, A.I., Zirbel, C.L., Leontis, N.B. (2013) Automated classification of RNA 3D motifs and the RNA 3D Motif Atlas. RNA, 19 (10), 1327-1340
9. Narayanan, B. C., Westbrook, J., Ghosh, S., Petrov, A. I., Sweeney, B., Zirbel, C. L., ... & Berman, H. M. (2014). The Nucleic Acid Database: new features and capabilities. Nucleic acids research, 42.
10. Andronescu, M. et al. (2008) RNA STRAND: the RNA secondary structure and statistical analysis database. BMC bioinformatics, 9 (1), 340.
11. Chojnowski, G., Waleń, T., Bujnicki, J.M. (2014) RNA Bricks – a database of RNA 3D motifs and their interactions. Nucleic acids research, 42.
12. Vanegas, P.L., Hudson, G.A., Davis, A.R, Kelly, S.C., Kirkpatrick, C.C., Znosko, B.M. (2012) RNA CoSSMos: Characterization of Secondary Structure Motifs – a searchable database of secondary structure motifs in RNA three-dimensional structures. Nucleic acids research, 40.
13. Taufer, M., Licon, A., Araiza, R., Mireles, D., Batenburg, F.H., Gultyaev, A.P., Leung, M.-Y. (2009) PseudoBase++: an extension of PseudoBase for easy searching, formatting and visualization of pseudoknots. Nucleic acids research, 37.
14. Kirsanov, D.D., Zanegina, O.N., Aksianov, E.A., Spirin, S.A., Karyagina, A.S., Alexeevski, A.V. (2013) NPIDB: nucleic acid – protein interaction database. Nucleic acids research, 41.
15. Lewis, B.A., Walia, R.P., Terribilini, M., Ferguson, J., Zheng, C., Honavar, V., Dobbs, D. (2011) PRIDB: a protein-RNA interface database. Nucleic acids research, 39.

Приложения.

Приложение 1 – основной скрипт проекта.

```
#!/usr/bin/env python3

import pymongo

from pprint import pprint

from csv import writer

import xml.etree.cElementTree as ET

from tempfile import TemporaryFile

from bson.json_util import dumps

from warnings import warn
```

```
from bottle import route, template, run, debug \
,static_file ,request, response, redirect, post, error, url
from cherrypy import wsgiserver

__author__ = 'Artiom Borisov *Amphyby*'

def num(s):
    try:
        return int(s)
    except ValueError:
        return float(s)

def especially4countdown(curr_type, entered_props,elder_passed={}):
#    pprint(entered_props)
    type_fields =
types.find_one({"name":curr_type},{"props":True,"length":True,'parentprops':True,"_id":False})
    parentprops = type_fields['parentprops']
#    print(list(exp_props))
    query_to_db = {'$and':[], 'type':curr_type}
#elder_BEGIN
#    response.delete_cookie('elder_type')
    if elder_passed:
        elder_type = elder_passed['type']
        dictWithElderProps = elder_passed['set_props']
        temp_query = {'$and':[], 'type':elder_type}
        props = types.find_one({'name':elder_type},{"props":True,"_id":False})['props']
        for item in props.keys():
            if not dictWithElderProps.get(item):
```

```
            continue
        elif isinstance(props[item],list):
            if len(props[item]) == 0:
                temp_query[item] = dictWithElderProps[item]
            else:
                temp_array = []
                for elem in dictWithElderProps[item].split('|DELIMITER|'):
                    temp_array.append({item:elem})
                temp_query['$and'].append({'$or' : temp_array})
        else:
            firs,seco = list(map(int,dictWithElderProps[item].split('|DELIMITER|')[0:2]))
            if firs and not seco:
                temp_query[item] = {'$gte' : firs}
            else:
                temp_query[item] = {'$lte' : seco,'$gte':firs}
    if not temp_query['$and']:
        del temp_query['$and']
    cursor = objects.find(temp_query,{"_id":False})
    global_str = {'$or':[]}
    for cursor_object in cursor:
        local_str = {'$and':[]}
        for everyparentprop in parentprops.keys():
            if isinstance(parentprops[everyparentprop],list):
                temporary_add = []
                for every_i in parentprops[everyparentprop]:
                    temporary_add.append({everyparentprop:cursor_object[every_i]})
                local_str['$and'].append({'$or':temporary_add})
            else:
```

```
local_str['$and'].append({everyparentprop:cursor_object[parentprops[everyparentprop]]})
            if len(local_str['$and']) > 1:
                global_str['$or'].append(local_str)
            elif len(local_str['$and']) == 1:
                global_str['$or'].append(local_str['$and'][0])
        if global_str['$or']:
            query_to_db['$and'].append(global_str)
    elif request.get_cookie('elder_type'):
        elder_type = request.get_cookie('elder_type')
        temp_query = {'$and':[], 'type':elder_type}
        props = types.find_one({'name':elder_type},{"props":True,"_id":False})['props']
        for item in props.keys():
            if not request.get_cookie('elder_'+item):
                continue
            elif isinstance(props[item],list):
                if len(props[item]) == 0:
                    temp_query[item] = request.get_cookie('elder_'+item)
                else:
                    temp_array = []
                    for elem in request.get_cookie('elder_'+item).split('|DELIMITER|'):
                        temp_array.append({item:elem})
                    temp_query['$and'].append({'$or' : temp_array})
            else:
                firs,seco = list(map(int,request.get_cookie('elder_'+item).split('|DELIMITER|')[0:2]))
                if firs and not seco:
                    temp_query[item] = {'$gte' : firs}
                else:
```

```
                temp_query[item] = {'$lte' : seco,'$gte':firs}
        if not temp_query['$and']:
            del temp_query['$and']
        cursor = objects.find(temp_query,{"_id":False})
        global_str = {'$or':[]}
        for cursor_object in cursor:
            local_str = {'$and':[]}
            for everyparentprop in parentprops.keys():
                if isinstance(parentprops[everyparentprop],list):
                    temporary_add = []
                    for every_i in parentprops[everyparentprop]:
                        temporary_add.append({everyparentprop:cursor_object[every_i]})
                    local_str['$and'].append({'$or':temporary_add})
                else:

local_str['$and'].append({everyparentprop:cursor_object[parentprops[everyparentprop]]})
            if len(local_str['$and']) > 1:
                global_str['$or'].append(local_str)
            elif len(local_str['$and']) == 1:
                global_str['$or'].append(local_str['$and'][0])
        if global_str['$or']:
            query_to_db['$and'].append(global_str)
#elder_END
    props, length = type_fields["props"], type_fields["length"]
    for item in props.keys():
        if not item in entered_props.keys() and not item+'_first' in entered_props.keys():
            continue
        elif isinstance(props[item],list):
```

```
            if len(props[item]) == 0:
                query_to_db[item] = entered_props[item]
            else:
                temp_array = []
                for elem in entered_props[item]:
                    temp_array.append({item:elem})
                query_to_db['$and'].append({'$or' : temp_array})
        else:
            firs = entered_props[item+'_first']
            seco = entered_props.get(item+'_second')
            if firs and not seco:
                query_to_db[item] = {'$gte' : num(firs)}
            else:
                query_to_db[item] = {'$lte' : num(seco),'$gte':num(firs)}
    if not query_to_db['$and']:
        del query_to_db['$and']
    pprint(query_to_db)
    return query_to_db



def multiple_query_builder(projection = [], exp_props = []):
    curr_type = request.get_cookie('type')
    type_fields =
types.find_one({"name":curr_type},{"props":True,"length":True,'parentprops':True,"_id":False})
    parentprops = type_fields['parentprops']
#    print(list(exp_props))
    query_to_db = {'$and':[], 'type':curr_type}
#elder_BEGIN
```

```
    elder_type = request.get_cookie('elder_type')
#    response.delete_cookie('elder_type')
    if elder_type:
        temp_query = {'$and':[], 'type':elder_type}
        props = types.find_one({'name':elder_type},{"props":True,"_id":False})['props']
        for item in props.keys():
            if not request.get_cookie('elder_'+item):
                continue
            elif isinstance(props[item],list):
                if len(props[item]) == 0:
                    temp_query[item] = request.get_cookie('elder_'+item)
                else:
                    temp_array = []
                    for elem in request.get_cookie('elder_'+item).split('|DELIMITER|'):
                        temp_array.append({item:elem})
                    temp_query['$and'].append({'$or' : temp_array})
            else:
                firs,seco = list(map(int,request.get_cookie('elder_'+item).split('|DELIMITER|')[0:2]))
                if firs and not seco:
                    temp_query[item] = {'$gte' : firs}
                else:
                    temp_query[item] = {'$lte' : seco,'$gte':firs}
        if not temp_query['$and']:
            del temp_query['$and']
        cursor = objects.find(temp_query,{"_id":False})
        global_str = {'$or':[]}
        for cursor_object in cursor:
            local_str = {'$and':[]}
```

```
            for everyparentprop in parentprops.keys():
                if isinstance(parentprops[everyparentprop],list):
                    temporary_add = []
                    for every_i in parentprops[everyparentprop]:
                        temporary_add.append({everyparentprop:cursor_object[every_i]})
                    local_str['$and'].append({'$or':temporary_add})
                else:

local_str['$and'].append({everyparentprop:cursor_object[parentprops[everyparentprop]]})
            if len(local_str['$and']) > 1:
                global_str['$or'].append(local_str)
            elif len(local_str['$and']) == 1:
                global_str['$or'].append(local_str['$and'][0])
        if global_str['$or']:
            query_to_db['$and'].append(global_str)
#elder_END


    props, length = type_fields["props"], type_fields["length"]
    for item in props.keys():
        if exp_props and not (item in exp_props):
            projection[item] = False
        if not request.get_cookie(item):
            continue
        elif isinstance(props[item],list):
            if len(props[item]) == 0:
                query_to_db[item] = request.get_cookie(item)
            else:
                temp_array = []
```

```
            for elem in request.get_cookie(item).split('|DELIMITER|'):
                temp_array.append({item:elem})
            query_to_db['$and'].append({'$or' : temp_array})
        else:
            firs,seco = list(map(int,request.get_cookie(item).split('|DELIMITER|')[0:2]))
            if firs and not seco:
                query_to_db[item] = {'$gte' : firs}
            else:
                query_to_db[item] = {'$lte' : seco,'$gte':firs}
    if not query_to_db['$and']:
        del query_to_db['$and']
#    pprint(query_to_db)
    return (projection, query_to_db, props)



def common_response(curr_type, option = 1, elder_params = {}):
    query_as_dict = {}
    temp = dict(types.find_one({"name":curr_type},{"props":True,"_id":False,"child":True, "hist" :
True}))
    props = temp["props"]
    hist = temp["hist"]
    child = temp["child"][0] if temp["child"] else ''
#    for every in props.keys():
#        response.delete_cookie(every)
    if option:
        for item in props.keys():
            if not (request.forms.get(item) or request.forms.get(item+'_first') or
request.forms.get(item+'_second')):
                response.delete_cookie(item)
```

```
            continue
        if isinstance(props[item],list):
            if len(props[item]) == 0:
                response.set_cookie(item,request.forms.get(item))
                query_as_dict[item] = request.forms.get(item)
            else:
                response.set_cookie(item,'|DELIMITER|'.join(request.forms.getall(item)))
                query_as_dict[item] = request.forms.getall(item)
        elif request.forms.get(item+'_first') and request.forms.get(item+'_second'):
            query_as_dict[item+'_first'] = str(min(num(request.forms.get(item+'_first')),
num(request.forms.get(item+'_second'))))
            query_as_dict[item+'_second'] = str(max(num(request.forms.get(item+'_first')),
num(request.forms.get(item+'_second'))))
    #          warn(query_as_dict[item+'_second'])
            response.set_cookie(item,
'|DELIMITER|'.join([query_as_dict[item+'_first'],query_as_dict[item+'_second']]))
        elif request.forms.get(item+'_first'):
            query_as_dict[item+'_first'] = request.forms.get(item+'_first')
            response.set_cookie(item, query_as_dict[item+'_first'] + '|DELIMITER|')
        elif request.forms.get(item+'_second'):
            query_as_dict[item+'_first'] = request.forms.get(item+'_second')
            response.set_cookie(item, query_as_dict[item+'_first'] + '|DELIMITER|')
    query_to_db = especially4countdown(curr_type, query_as_dict, elder_params)
    amount = num(objects.count(query_to_db))
    return template('choosin_props',{"props":props, "type":curr_type, 'url' : url,
'entered_props':query_as_dict,'child':child, 'hist' : hist, 'amount': amount })


@route('/')
def home_page():
```

```
    elder_one = request.get_cookie('elder_type')
    response.delete_cookie('elder_type')
    if elder_one:
        props = types.find_one({"name":elder_one},{"child":False,"_id":False})['props']
        for every_prop  in props:
            if not request.get_cookie('elder_'+every_prop):
                continue
            response.delete_cookie('elder_'+every_prop)
    elder_one = request.get_cookie('type')
    response.delete_cookie('type')
    if elder_one:
        props = types.find_one({"name":elder_one},{"child":False,"_id":False})['props']
        for every_prop  in props:
            if not request.get_cookie(every_prop):
                continue
            response.delete_cookie(every_prop)
    coretypes = []
    cursor = types.find({},{"name":1,"_id":-1})
    for doc in cursor:
        coretypes.append(doc["name"])
    return template('root_template',{"coretypes" : coretypes})


@route('/middle_choice')
def middle_choice_done():
    curr_type = request.get_cookie('type')
    response.set_cookie('elder_type',curr_type)
    type_fields = types.find_one({"name":curr_type},{"props":True,"child":True,"_id":False})
    props, child = type_fields["props"], type_fields["child"][0]
```

```
    elder_props4search = {'type' : curr_type, 'set_props' : {}}
    for item in props.keys():
        if not request.get_cookie(item):
            continue
        temp = request.get_cookie(item)
        elder_props4search['set_props'][item] = temp
        response.delete_cookie(item)
        response.set_cookie('elder_'+item, temp)
    response.set_cookie('type',child)
    return common_response(child, 0, elder_props4search)


@route('/export_props')
def export():
# если переделовать на post
#    if not request.forms.get('type'):
    curr_type = request.get_cookie('type')
#    else
    props = types.find_one({"name":curr_type},{"props":True,"_id":False})["props"]
    return template('export_props', {"props" : props, 'url' : url, 'type' : curr_type})


@route('/static/:path#.+#', name='static')
def static(path):
    return static_file(path, root='static')


@post('/export')
def export_results():
    exp_props = request.forms.getall('props4export')
```

```
    projection, query_to_db,props = multiple_query_builder(projection = {'_id' : False,'id' : False, 'type'
: False}, exp_props = request.forms.getall('props4export'))
    sortby = request.forms.get('sortby')
    if sortby and isinstance(props[sortby], dict):
        order = request.forms.get('order')
        if order == 'descending':
            order = pymongo.DESCENDING
        else:
            order = pymongo.ASCENDING
        cursor_of_results = objects.find(query_to_db,projection).sort(sortby,order)
    else:
        cursor_of_results = objects.find(query_to_db,projection)



    the_result = '<table><tr>'
    cursor_of_results = list(cursor_of_results)
    '''#вывод csvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
    fields = sorted(cursor_of_results[0].keys())
    for head in fields:
        the_result += head+';'
    the_result += '\n'
    for i in cursor_of_results:
        for j in fields:
            if not i[j]:
                i[j] = '\\N'
            the_result += str(i[j]) + ';'
        the_result += '\n' #neeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeen'''
    if not cursor_of_results:
```

```
        return "no matched structures"
    fields = sorted(cursor_of_results[0].keys())
    for head in fields:
        if not head in exp_props:
            continue
        the_result += '<td>' + head + '</td>'
    the_result += '</tr>'
    for i in cursor_of_results:
        the_result += '<tr>'
        for j in fields:
            if not j in exp_props:
                continue
            if not i[j]:
                i[j] = '\\N'
            the_result += ('<td>' + str(i[j]) + '</td>')
        the_result += '</tr>'
#    print(the_result)
#    response.set_header('Content-Type', 'text/plain')
#    response.set_header('Content-Disposition', 'attachment; filename="requestedexport.csv"')
    return '<html><head><style>table, th, td {    border: 1px solid black;}</style></head><body>' +
the_result + '</table></body></html>'


@post('/choosin')
def choosin():
    curr_type = request.forms.get('type')
    response.set_cookie('type', curr_type)
#    pprint(dict(request.cookies))
    return common_response(curr_type)
```

```
@post('/datagathering')
def data_gathering():
    value = request.forms.get('forchart')
    query_to_db = multiple_query_builder()[1]
    distinct_results = sorted(objects.distinct(value,query_to_db))
    answer = ET.Element('response')
    for every in distinct_results:
        sub_el = ET.SubElement(answer, 'item')
        sub_key = ET.SubElement(sub_el, 'key')
        sub_key.text = str(every)
        sub_val = ET.SubElement(sub_el, 'val')
        query_to_db[value] = every
#       print(every,str(objects.count(query_to_db)))
        sub_val.text = str(objects.count(query_to_db))
#    warn(ET.tostring(answer, method='xml'))
    response.set_header('Content-Type', 'text/xml')
    return ET.tostring(answer, method='xml')

@error(404)
def error404(error):
    return 'Nothing here, sorry'

@error(405)
def error405(error):
    return 'smth happened to methods, maybe here must be POST or GET?'

@route('/favicon.ico')
```

```
def favicon_for_everyone():
    return static_file(root='.\\',filename = 'favicon.ico')


connection = pymongo.MongoClient("127.0.0.1",27001)
database = connection.diplomatasks
types = database.types
objects = database.objects


debug(True)
run(server='cherrypy', host='localhost', port = 8080)
#run(host='localhost', port = 8080)
```

Приложение 2 – шаблон корневого каталога.

```
<!DOCTYPE html>
<html>
<head>
    <title>Root Url</title>
</head>
<body>
<form action="/choosin" method="POST">
    <input type='radio' name='type' value='{{coretypes[0]}}' checked>{{coretypes[0]}}<br>
    %for item in coretypes[1:]:
    <input type='radio' name='type' value='{{item}}'>{{item}}<br>
    %end
    <input type="submit" value="Submit">
</form>
</body>
</html>
```

Приложение 3 – шаблон выбора свойств.

```
<!DOCTYPE html>

<html>

  <head>

    <title>Choosin props</title>

    <script type="text/javascript" src="{{ url('static', path='Chart.min.js') }}" charset="utf-8"></script>

    <script type="text/javascript" src="{{ url('static', path='statistics.js') }}" charset="utf-8"></script>

  </head>

  <body>

    <h>Chosen type is {{type}}</h><br>

    <button onclick=" window.open('/export_props','_blank')">Get a document with all the matched objects</button>

%from warnings import warn

%if child:

    <br><button onclick=" window.open('/middle_choice','_blank')">to child type</button><br>

%end

    <form action="/choosin" method="POST">

      <input type="hidden" name="type" value="{{type}}">

%for item in sorted(props.keys()):

%if isinstance(props[item], list):

      {{item}}:<br>

%if len(props[item]) == 0 and entered_props.get(item):

      <input type='text' name="{{item}}" value="{{entered_props.get(item)}}"><br>

%end

%if len(props[item]) == 0 and not entered_props.get(item):

      <input type='text' name="{{item}}"><br>

%end

%if len(props[item]) != 0:

%for every in sorted(props[item]):
```

```
%if  entered_props.get(item) and every in entered_props.get(item):
      <input type="checkbox" name="{{item}}" value="{{every}}" checked>   {{every}}<br>
%end
%if  not entered_props.get(item) or not every in entered_props.get(item):
      <input type="checkbox" name="{{item}}" value="{{every}}">   {{every}}<br>
%end
%end
%end
%end
%if not isinstance(props[item], list):
%if  not (entered_props.get(item + "_first") or entered_props.get(item + "_second")):
      {{item}}:<br>
      <input type="number" name="{{item}}_first" min="{{props[item]["min"]}}"
max="{{props[item]["max"]}}">
      <input type="number" name="{{item}}_second" min="{{props[item]["min"]}}"
max="{{props[item]["max"]}}"><br>
%end
%if  entered_props.get(item + "_first") or entered_props.get(item + "_second"):
      {{item}}:<br>
      <input type="number" name="{{item}}_first" min="{{props[item]["min"]}}"
max="{{props[item]["max"]}}" value="{{entered_props.get(item + "_first")}}">
      <input type="number" name="{{item}}_second" min="{{props[item]["min"]}}"
max="{{props[item]["max"]}}" value="{{entered_props.get(item + "_second")}}"><br>
%end
%end
%end
      <input type="submit" value="Submit">
    </form>
    <br>
```

```
      <p>Amount of objects satisfying previous query is:      {{amount}}</p>

    <br>

    <button onclick="location.href='/'">change type</button><br>

%if hist:

<input type='radio' name='forchart' value='{{hist[0]}}' checked>{{hist[0]}}<br>

%for every_hist in hist[1:]:

<input type='radio' name='forchart' value='{{every_hist}}'>{{every_hist}}<br>

%end

    <button id='datashow' onclick=" return datashow()">Get hisogram by value selected above on submitted properties</button><br>

    <canvas id="myChart" width="800" height="400"></canvas>

%end

  </body>

</html>
```

Приложение 4 – шаблон выбора параметров экспорта.

```
<!DOCTYPE html>

<html>

<head>

  <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.12.2.min.js"></script>

  <script type="text/javascript" src="{{ url('static', path='cachingrealization.js') }}" charset="utf-8"></script>

  <script>should_be_displayed('{{type}}');</script>

  <title>Export properties</title>

</head>

<body>

  <form name="exportProps" onsubmit="return someFormAction('{{type}}')" action="/export" method="POST">

      Sort by &ltif none of suggested selected - random export order&gt:<br>

      <input list="types" name="sortby">
```

```
    <datalist id="types">
%for item in sorted(props.keys()):
%if isinstance(props[item],dict):
        <option value="{{item}}">
%end
%end
    </datalist>
    <br>in order:<br>
    <input type='radio' name='order' value='ascending' checked>ascending<br>
    <input type='radio' name='order' id='desc_radio' value='descending'>descending<br>
    <br>Which fields to show in export:<br>
%for every_property in props.keys():
    <input type='checkbox' name='props4export' id = '{{every_property}}'
value='{{every_property}}'>{{every_property}}<br>
%end
    <input type="submit" value="Submit">
  </form>
  <button id='retreive' onclick=" return props_retriever('{{type}}')">Fill all the fields with values used
before</button>
</body>
</html>
```

Приложение 5 – javascript методы для кеширования параметров экспорта на стороне пользователя.

```
function should_be_displayed(type){
  $(document).ready(function(){
    if(localStorage.getItem(type) !== null){
      $("#retreive").show();
    }
    else{
```

```
        $("#retreive").hide();
      }})
};
function someFormAction(type){
   var boxes = document.querySelectorAll("input[type='checkbox']");
   var submitted_boxes = [];
   for (var i = 0; i < boxes.length; i++) {
      var box = boxes[i];
      if (box.checked) {
         submitted_boxes.push(box.value);
      }
   }
   if (submitted_boxes || document.querySelector('input[list]').value ||
document.forms["exportProps"]["order"].value) {
      localStorage.setItem(type,JSON.stringify({'boxes' :
submitted_boxes,"sortby":document.querySelector('input[list]').value,'order':document.forms["expor
tProps"]["order"].value}));
   }
}
function props_retriever(type) {
   var maitem = JSON.parse(localStorage.getItem(type));
   var boxes = maitem['boxes'];
   for (var i=0; i<boxes.length; i++) {
      document.getElementById(boxes[i]).checked=true;
   };
   var sortby = maitem['sortby'];
   document.querySelector('input[list]').value = sortby;
   var order = maitem['order'];
   if (order == 'descending') {
```

```
    document.exportProps.order[1].checked=true;
  };
}
```

Приложение 6 – javascript методы для вывода статистики.

```
function datashow() {
  var radios = document.getElementsByName('forchart');
  value2pass = '';
  for (var i = 0, length = radios.length; i < length; i++) {
    if (radios[i].checked) {
      // do whatever you want with the checked radio
      value2pass = radios[i].value;
      break;
    }
  }
  var ctx = document.getElementById("myChart");
  var data = {
  labels: [],
  datasets: [
    {
      label:  value2pass,
      backgroundColor: "rgba(255,99,132,0.2)",
      borderColor: "rgba(255,99,132,1)",
      borderWidth: 1,
      hoverBackgroundColor: "rgba(255,99,132,0.4)",
      hoverBorderColor: "rgba(255,99,132,1)",
      data: [],
    }
    ]
```

```
};
var xhttp;
if (window.XMLHttpRequest) {
    xhttp = new XMLHttpRequest();
    } else {
    // code for IE6, IE5
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
xhttp.onreadystatechange = function(){
    if (xhttp.readyState == 4 && xhttp.status == 200) {
        xmlDoc = xhttp.responseXML;
        result = xmlDoc.getElementsByTagName("response")[0].getElementsByTagName("item");
        for (i=0; i<result.length;i++){
            temp = result[i];
            //console.log(temp.getElementsByTagName('key')[0].childNodes[0].nodeValue)
            //console.log(temp.getElementsByTagName('val')[0].childNodes[0].nodeValue)
            data.labels.push(temp.getElementsByTagName('key')[0].childNodes[0].nodeValue)

data.datasets[0].data.push(parseFloat(temp.getElementsByTagName('val')[0].childNodes[0].nodeValue))
        //xmlDoc = xhttp.responseText;
        //alert(xmlDoc) childNodes[1].nodeValue
        }
        //console.log(data['labels'])
        //console.log(data['datasets'][0]['data'])
        var myBarChart = new Chart(ctx, {
            type: 'bar',
            data: data,
            options: {
```

```
                responsive: false
            }
        });
        return myBarChart;
    }
  }
  xhttp.open('post','datagathering',true)
  xhttp.send('forchart='+value2pass);

}
function datashowtest() {
  var ctx = document.getElementById("myChart");
  var data = {
  labels: ["January", "February", "March", "April", "May", "June", "July"],
  datasets: [
    {
      label: "My First dataset",
      backgroundColor: "rgba(255,99,132,0.2)",
      borderColor: "rgba(255,99,132,1)",
      borderWidth: 1,
      hoverBackgroundColor: "rgba(255,99,132,0.4)",
      hoverBorderColor: "rgba(255,99,132,1)",
      data: [65, 59, 80, 81, 56, 55, 40],
    }
  ]
};
var myBarChart = new Chart(ctx, {
  type: 'bar',
```

```
    data: data,
    options: {
        responsive: false
    }
});
return myBarChart;
}
```