**National Research University Higher School of Economics**

**Faculty of Computer Science**

**Bachelor’s Programme in Data Science and Business Analytics**

**GROUP TERM PAPER**

**Software Project**

**Digital Student Assistant**

**Prepared by students: of Group 192 in Year 3,**

**Rutsinskiy Arsenty Andreevich, Group 192, Year 3,**

**Fakhretdinov Emil Vilevich, Group 192, Year 3,**

**Kukarkins Nikolajs, Group 192, Year 3,**

**Dryagin Artem Sergeevich, Group 192, Year 3.**

**Term Paper Supervisor:**

**Faculty of Computer Science, Senior Lecturer, Parinov Andrey Andreevich**

**Moscow**

**2022**

[**1. Key Terms & Definitions**](#_bi5v82vln6r7) **4**

[**2. Introduction**](#_iyzc9wd9v4v1) **5**

[2.1 Goals](#_4ir7nt8yy9x9) 5

[2.2 Tasks](#_yj5gjxk2jeuc) 5

[**3. Review and Contemporary Analysis of sources and analogous**](#_1zlbepmh6r8h) **6**

[3.1 Analogous materials](#_ypjapz443liy) 6

[3.2 Sources](#_s9d4ft4cgb2c) 7

[**4. Data gathering**](#_ymikh2fpvjf2) **8**

[4.1 Source 1](#_wc96poygv6qx) 8

[4.2 Source 2](#_78c9spdo3vim) 8

[**5. NLP Theoretical Part**](#_jno4j8qku60) **9**

[5.1 Tokenization & Conventional Embeddings](#_8yyzghq17wo9) 9

[5.2 BERT](#_ilijco7oxchw) 11

[5.3 Cosine Similarity](#_b0b4zsf3y2s9) 13

[**6. Recommender system implementation**](#_bts7qgwjc2e4) **14**

[6.1 Functional and Non-Functional Criteria](#_erckls3m6sr9) 14

[6.2 Initial “Name-to-Tag” Approach](#_e116lb4mkor5) 14

[6.3 DeepPavlov Classifier](#_xdeufo8ttifg) 16

[6.4 “Tags-to-Name” Approach](#_ezir9kgs517w) 18

[**7. Website Implementation**](#_ts8fcowes761) **20**

[7.1 Basic resources for creating a website:](#_om03rexos68j) 20

[7.2 My goals](#_7jg8b9vzp901) 20

[7.3 Technical basis](#_wz68f1n9tbp) 20

[7.4 Functional and Non-functional requirements](#_lszfg2serwrk) 21

[7.5 Implementation](#_mefinj3syp2l) 21

[**8. Visual Analytics**](#_g8xuji7bik4v) **26**

[8.1 Comparative analysis of Business Intelligence Tools](#_fmlfv116110n) 26

[8.2 Functional and Non-Functional criteria](#_74jbloe06f8d) 29

[Functional criteria](#_ub8q57ami8uf) 29

[Non-functional criteria](#_iqd1mlcucfpm) 29

[8.3 Visual Analytics Implementation](#_gv2m1xm2ir7o) 29

[8.4 Conclusion](#_mfdkkp1faw8h) 32

[**9. Key Conclusions and future plans**](#_wsr60ooqid2f) **33**

[**10. Bibliography**](#_m5dlnc9rx8xs) **34**

[**11. Appendix**](#_506d4hmbaozu) **35**

# 1. Key Terms & Definitions

Flask - A framework for creating web applications in the Python programming language, using the Werkzeug toolset, as well as the Jinja2 template engine. It belongs to the category of so-called microframes - minimalistic frameworks of web applications that deliberately provide only the most basic features.

PostgrqSQL (pronounced post-gress-Q-L) - is an open source relational database management system (DBMS) developed by a worldwide team of volunteers.

SQLAlchemy - is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

NLP (Natural Language Processing) - subfield of mathematical linguistics and arificial intelligence aimed at making computers learn, process, manipulate textual data.

Corpus - a huge collectin of text (poems, novels, questions) generated by native speaking humans. Used for initial training.

Folksonomy (Collaborative Tagging) - system of classification, where the user decides with the certain tags of some content, unlike Taxonomy, where the owner solely identifies the tags.

Lemmatization - process of grouping similar words into one single item identified by its lemma (canonical form). For example, “Good”, “better”, “best” turn into just “good”.

Embedding - process of turning high-dimensional inputs such as words into low-dimensional vectors. Embeddings are putting similar meaning items close to each other making machine learning easier on large inputs.

Tokenization - process of replacing textual data (each word) with non-sensitive token (integer) that keeps all the information about the word in a number.

Attention - a deep learning technique that mimics human congnitive attention. The method focuses its “attention” on the most important part of the words while diminishing other parts.

Transformer - machine learning model that incorporates self-attention technique and gives different words different weights of their significance.

# 

# 2. Introduction

Today’s system at HSE of choosing project, either final year or extracurricular is a nightmare. Students have to choose between a “Ярмарка проектов”, where there is a very limited number of projects for not all faculties. On the other hand, we may choose projects via spread sheet in google, where 500+ projects are listed and not sorted, there are also no requirements and very limited descriptive information about the topics. So, students must go to great length to find the most interesting theme.

Our systems aim to create a useful and easy system than can be used throughout all faculties by students to find great projects and apply for them.

## 2.1 Goals

1. Develop a NLP system that gives recommendations on the projects to the students.
2. Create a useful visualization, which provides insights on the project’s list.
3. Create a web application for user and NLP system interaction
4. Create a tool that can collect information from different websites, such as pf.hse.ru.

## 2.2 Tasks

1. Find the data and find the best way to process it.
2. Parse the data
3. Gain information about most useful NLP algorithms, find best instruments and implement several possible NLP recommender models.
4. Perform data analytics and draw conclusions.
5. Find framework to develop web application
6. Develop python backend and html frontend for web app and embed NLP system

# 

# 3. Review and Contemporary Analysis of sources and analogous

## 3.1 Analogous materials

The most important and the hardest part on the behalf of team was getting appropriate data.

The problem with any dataset found in the net is that they are not suitable for my part. The most important feature for the dataset to be truly informative and effective at training the model it must have very specific information: tags for example, titles that look like project names, it also better have a short description. All of this was present at “Ярмарка проектов”

Initially we decided to use the data from this service made by the university itself. The data there appears to be very limited. Even though there are a lot of different categories and characteristics of the project such as tags date the teacher and a short description the overall size of the data set was very small. So all what we could do was to mine about 50 projects overall which is definitely not enough to implement the models I'm trying to. It seems that all the projects at the HSE are appearing in some specific time of the academic year and are only present for some period, which we did not catch.

Then we also considered google spread sheet with all the projects. There was also very limited amount of data. It is also extremely bulky and hard to navigate.

## 3.2 Sources

We found 3 different resources of data:

1. “Ярмарка проектов”

Website made and supported by HSE. Very informative data, however very limited.

1. Google Spreadsheet.

Assembled each year collection of year projects. It has almost everything: Names, Dates, Type of project, Form of application, links. Tags are not present, however.

1. Ru\_kw\_eval\_dataset

Also we found 1 more dataset, that was provided for us by some Russian student. It consists of 4 subsets: Each of batch 4000 mined from Habrahabr, RT, Cyberleninka and ng. There were also keywords field present. The habr data was not ideal but somewhat relatable with my work.

# 4. Data gathering

## 4.1 “Ярмарка проектов”

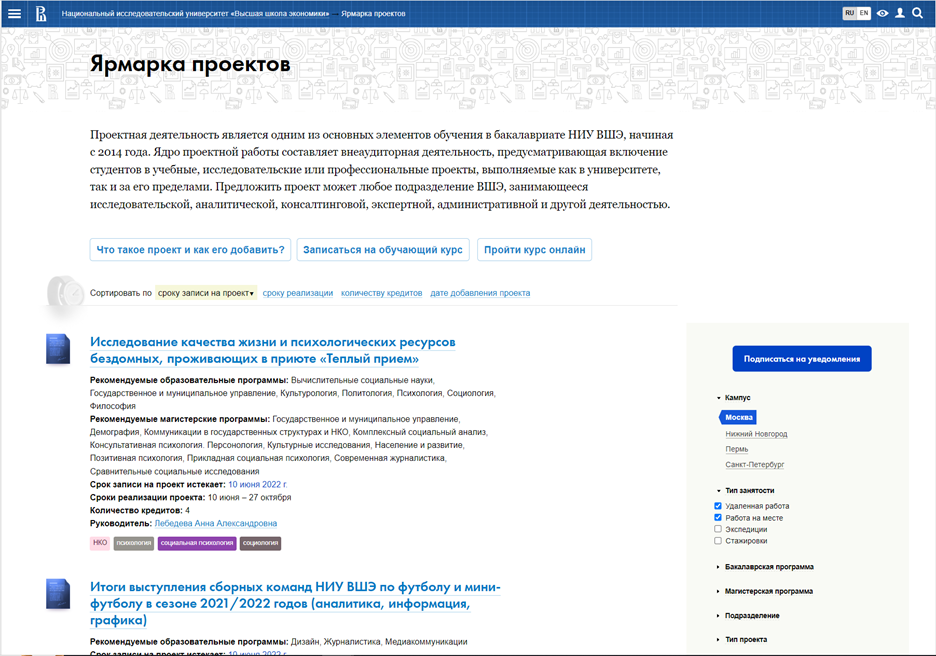
First step of gathering data was finding the place, from where we can actually take something. As it was already mentioned before, one of the resources we took was “pf.hse.ru” (Ярмарка проектов).

Figure 4.1.1 Search page

The first thing that was done is parsing the params, that can be used to make a request, like campus or type of project. For the parsing selenium package was used. After gathering this small amount of information, it was saved on PostgreSQL database as a table.

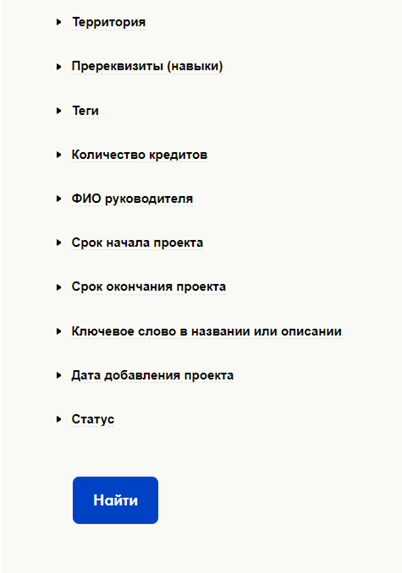
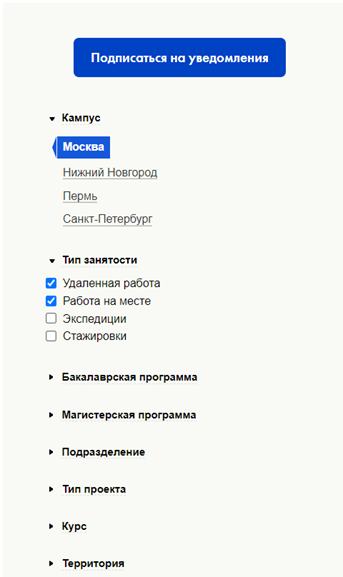
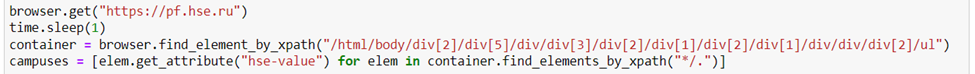


Figure 4.1.2 Params

Second step was getting the information about the projects on this website. To do that, I made requests for all possible campuses, so that way we can parse every project (link to it) that is availavle.



Talking about structure, all the projects are in one big container (class), named main content. We can parse some information already at the search page, but getting the links of the projects and parse information that it contain is much more valuable. To navigate through serach pages i just added page{number}.hmtl to the search link, where number is from 1 to max number of the page.



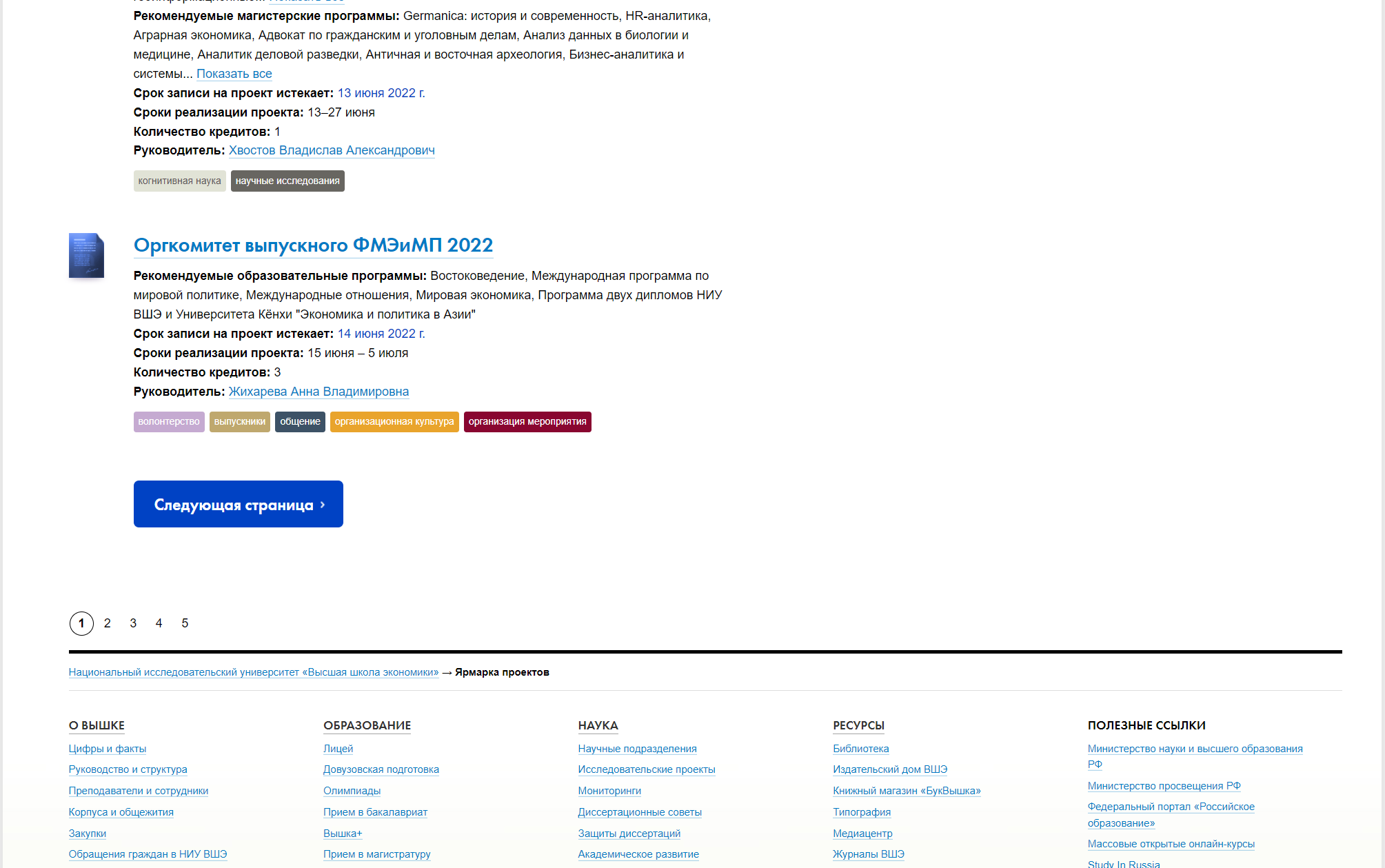
Like in this case it will be 5:

Figure 4.1.3 Page navigation

One of the good things, that the values of campuses are not hardcoded, so even if HSE will build a new campus in new city, everything will still work. So, after we get the values of every campus, we just need to go through “https://pf.hse.ru/?campus=VALUE” and get the links of the projects, which looks like “https://pf.hse.ru/639856393.html”.

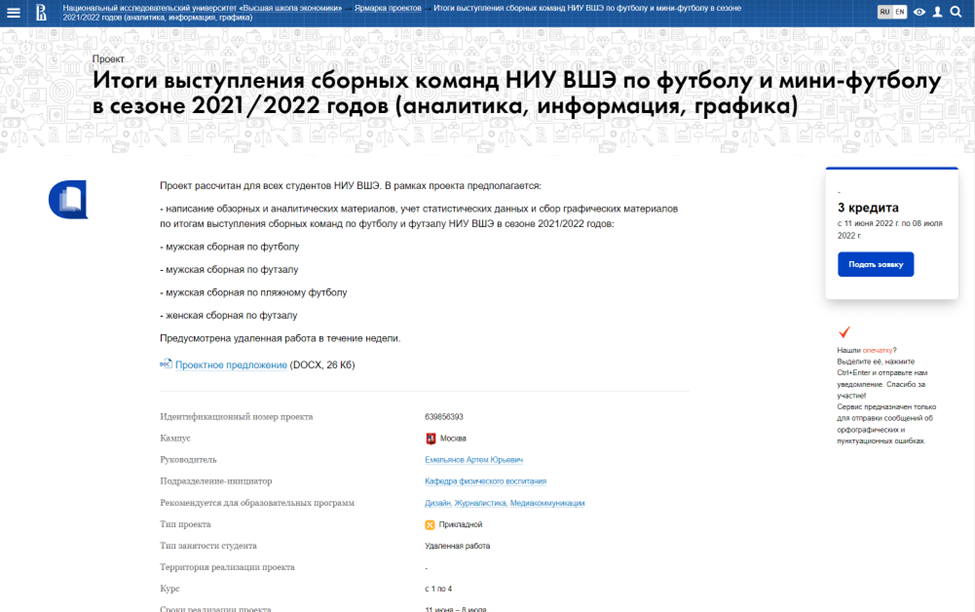


Figure 4.1.4 Page of the project

The parser works that firstly it gets all the links to the project on 1 page for campus n, after that i goes through them, collection all the information that can be usefull for further research: "Name", "Id", "Campus", "Supervisor", "Co-supervisor", "Initiating unit", "Recommended for educational program", "Recommended for master's programs", "Recommended","Type of project","Type of student employment", "Project implementation area", "Course", "Project implementation timeline", "Applications are accepted until", "Number of vacancies on the project", "Amount of credits", "Intensity of project activity", "Way of setting goals", "Necessary", "Tags", "Link", and store it in pandas data frame. The same as on the search page, all the information is inside class named “content”.

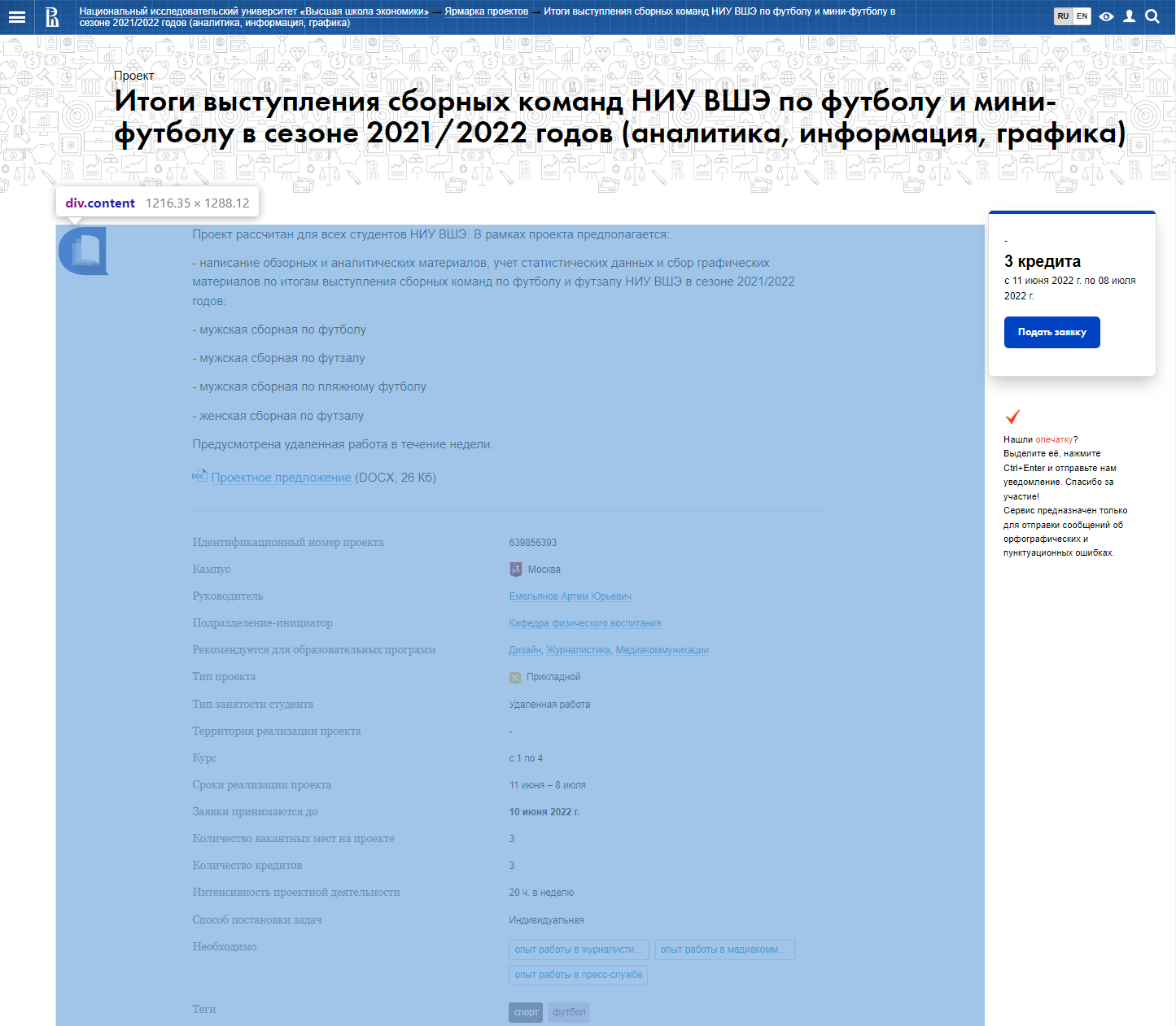


Figure 4.1.5 Main information of the project

The next step is changing page number, and if there is no more pages, it changes page\_number to 1 and campus to the different one.

After getting full data frame, one of the last steps we need to do is to delete all the duplicates, because some projects are available in 2 different campuses, so we will have 2 copies of the same project in our dataframe.

Last but not least before we add out dataframe to the PostgreSQL database, we add special column called “Iteration”, using which we can refer to another tabel in our database, containing “Data” and “Iteration”. So we will know, when this piece of data was collected. So, we do not relace our table each time we parse, we add it add store the information when it was parsed.

Some of them are missed for some projects, for example sometimes there is no Co-supervisor or "Recommended for educational program".

## 4.2 Google spreadsheet

As a second resource we used google spreadsheet with projects for second and third year DSBA.

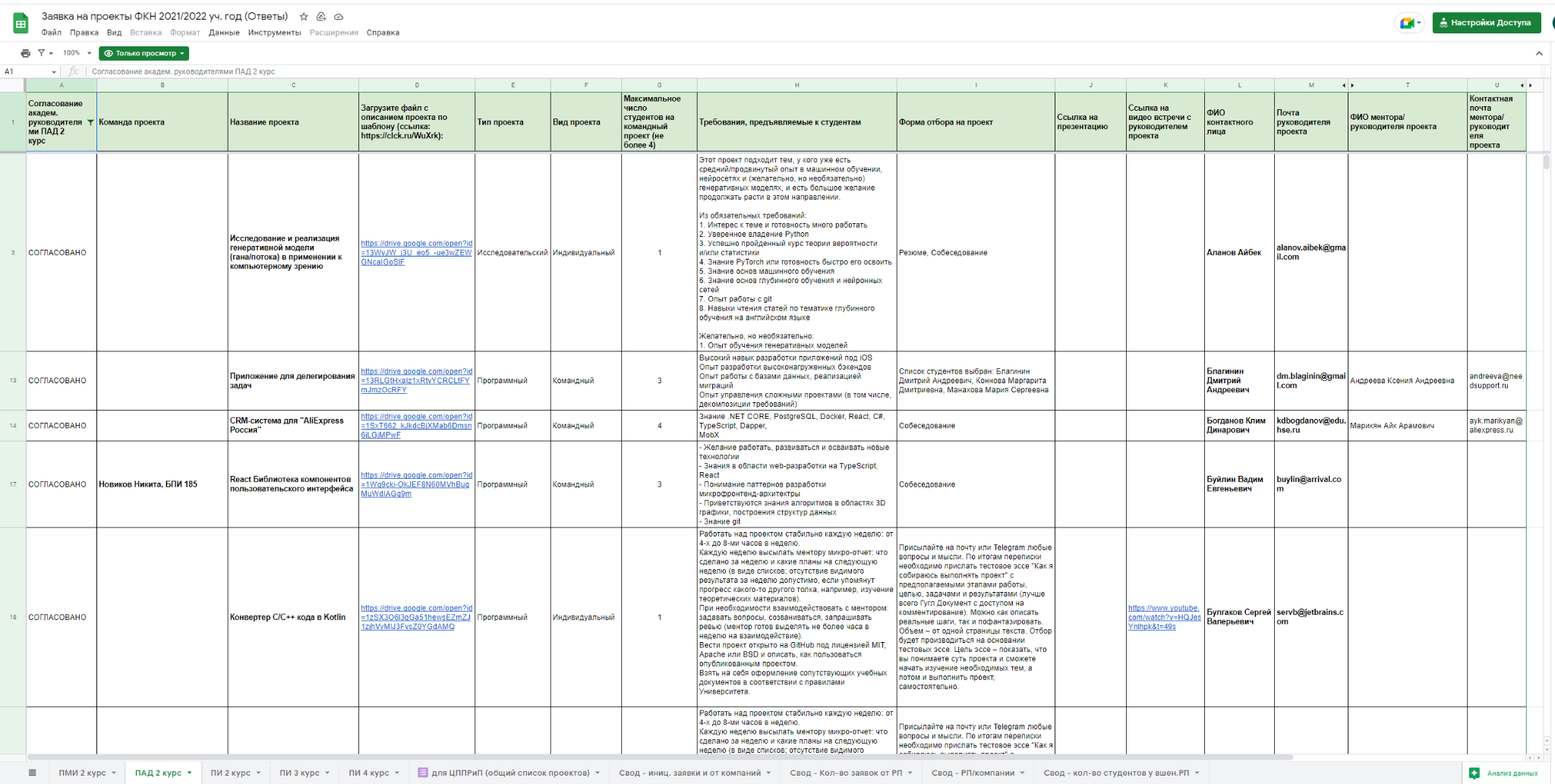


Table 4.2.1 Google spreadsheet DSBA 2 / 3 Course

There is not much information about the project itself there, only Supervisor and Name of the project basically, but this google spreadsheet can be used for further development, thire is link with more detailed description of each project that can be parsed.

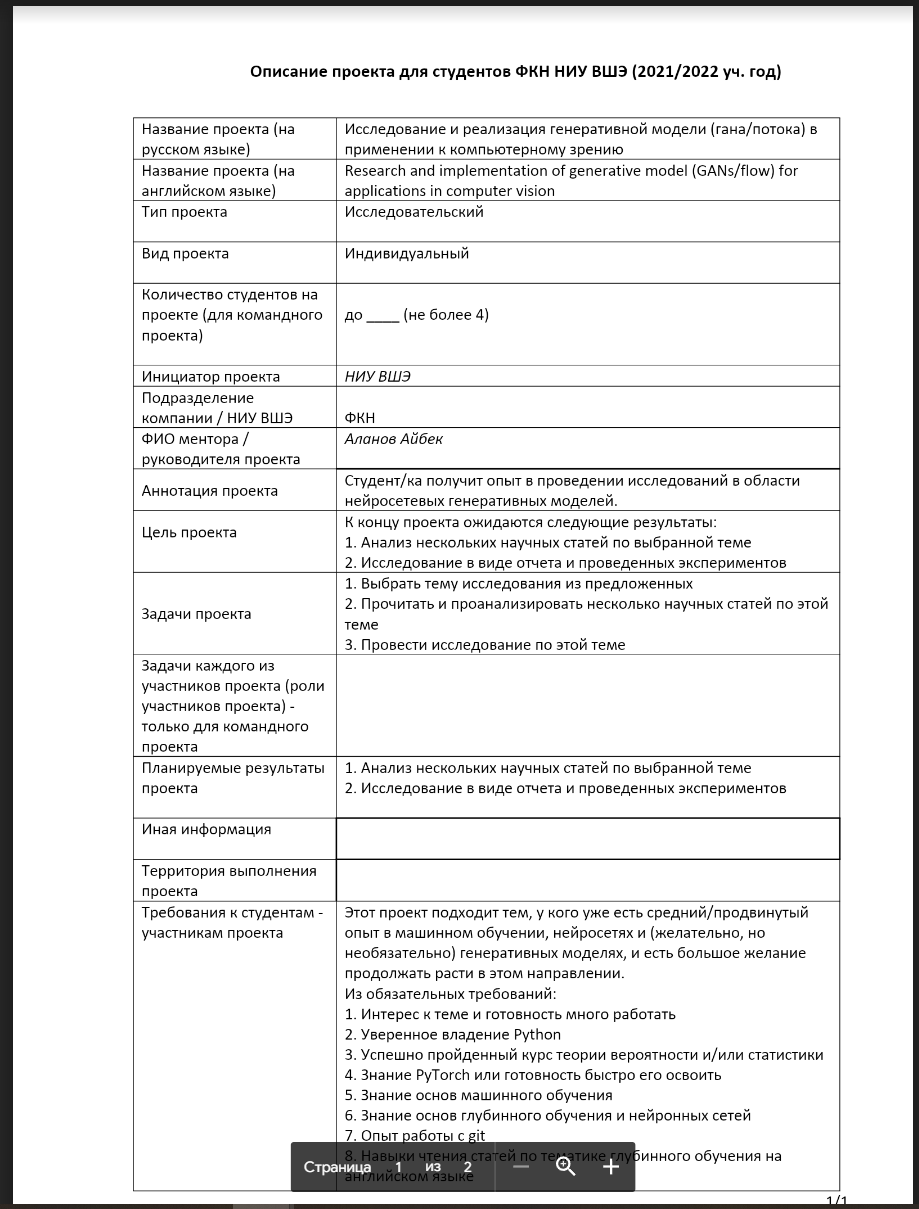


Figure 4.2.2 Additional information about project

## 4.1 Source 3

One of the resource, used to get the training set is a set of 4 subsets: Habrahabr, RT, Cyberleninka and ng. Input data is .json, that contains full titles and few keywords for them.

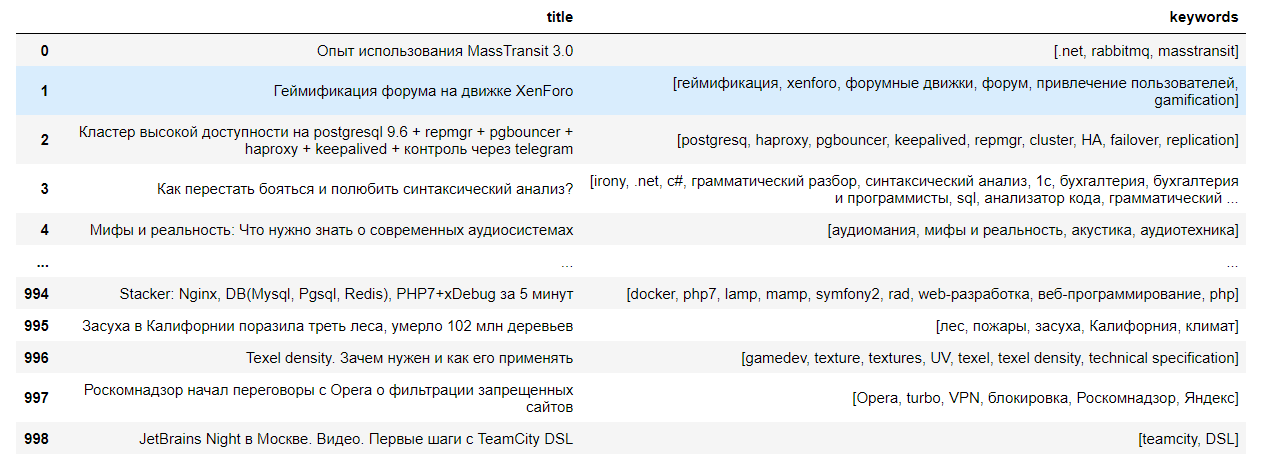


Table 4.3.1

It is tranformed into pandas dataframe, so it will be easier to work with.

# 5. NLP Theoretical Part

## 5**.1 Tokenization & Conventional Embeddings**

First, data should be analysed and **preprocessed**. Typically, the most common way to preprocess sentences is to remove stopwords: articles, for instance. In many cases it can significantly improve the overall performance, but only for big texts, for example whole passages from books etc. In our case it was not desirable to remove any words, because the system is already working with very limited data, removing it can lead to losing some semantics and certain context, which could be better interpreted with embedders.

To actually start any NLP framework it needs a number-like input, while it cannot procedure words. Tokenization is the way to solve the problem. Tokenizers take a sentence and transform it into a sequence of numbers, where each number represents a word by its place in the dictionary of all possible words. However, assigning numbers to a certain words is not enough to feed it to any kind of model.

Embedders are systems that are doing almost the same thing, but on the next level: they represent words in vectors, not just integers. This allows to store everything in a 2D vector space, where certain dependencies between words are captured. The easiest and most popular example is as follows: If we have a vector that represents a “King”, then by subtracting a vector for “man” and adding a vector for “woman”, we should get a vector representing “Queen”. It means that to some degree the algorithm can catch those dependencies and “understand” the structure of the language and meanings of most words.

There are a lot of different Embeddings: word2vec, FastText, ELMo, GloVe, Ngrams, Skipgram etc. Each of them has a specific task for which they are better than the others.

* **ELMo:** uses bi-directional LSTM to create embeddings for the whole sentence, not just single words. This is perfect if we need some specific meaning of the word. This is extremely helpful for English for example, where 1 word can mean tens of things simultaneously (word “set” has 400+ definitions). In our case it won’t be so helpful, as most words in Russian have the one and only meaning. Also, our sentences are very short, and typically any model can get the right meaning out of them.
* **Word2Vec:** most popular and common embedding. The biggest problem is that word2vec only analyzes the word in a given context, so it is not able to understand dependencies good enough for small texts. Word2Vec can be of 2 types: Skipgram and Continuous bag of words (CBOW), which are almost the same, however Skip-Gram tends to deal with out of vocabulary words better than CBOW, which for us could have been great, as the project titles are very vague and are not frequently reoccurring.
* **GloVe:** Incorporates the same features as Word2Vec, however it not only has a local context, it also incorporates global words statistics, which allows it to understand specific semantics better. For the DeepPavlov classifer implementation GloVe was the choice.
* **FastText:** Another great embedding that can integrate both Skip-Gram and CBOW features. It uses negative sampling to show the words that are not related to the initial sentence. It is also extremely fast, works well with compound words, which are somewhat frequent in Russian language. Furthermore, the implementation using the FastText library was very effective (fastest and easiest). FastText Skipgram implementation was initially used for the main approach, however it appeared to have major inadequacies.

The flaws appeared vivid because of **multilingual** problem. Not only the most models are pretrained in English language, but the project names were frequently in different languages. It typically had some framework or program names, for instance, “Python”, “Tableu” and so on. It is obvious, that embedding an english word and putting it in the same vector space with Russian words is not an option, as the whole space would not be representative (Most embedders are aimed only for single languages). Obviously, using some API translator was not enough to turn the whole sentence into one language. It would presumably lead in some cases of “Python”, becoming a “Питон”, which could be interpreted as a snake sometimes. However, some projects contained a lot of words that could be translated to Russian without loss of information. Google translate API was already dealing with this task great, as all the “household” words were not translated, while the API knew it importance and did not translate.

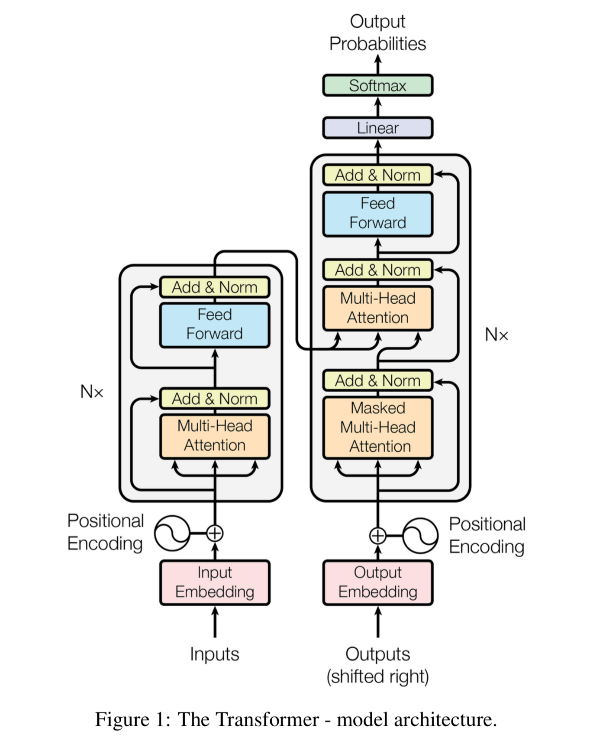
This hassle led to using a DeepPavlov **BERT** multilingual model for embedding the words. That way we were able to get rid of noisy words that are better translated and keep important foreign words in the same vector space.

## 5**.2 BERT**

In order to understand BERT (Bidirectional Encoder Representations from Transformers), we need to investigate transformers, first. Transformers are a relatively new invention in the world of NLP. Transformer – framework that is based on self-attention, which essentially weights the significance of each of the words (To which “selves” does it need to pay more attention in order to understand the sentence better?).

Transformers are very similar to RNNs, which have been used back in the days for NLP and CV (Computre vision). LSTM (Long-Short Term Memory model), for example, is only able to process one word at a time, while transformers process the input as a whole at one, decreasing runtime.

The transformer has the following scheme (Figure 5.2.1): It consist of 2 parts: Encoder (left hand side) and decoder (right hand side). Positional encoding in both parts is responsible for numbering the words in the text in the order they appear. The encoder then processes the inputs and simultaneously generates embeddings for all words in the sentence. The decoder, on the other hand processes the previously occurred words and their meanings alongside with the embedded words, choosing the variant with the highest probability. Basically, Encoder in that case decides with the context of the word, while decoder decides what should be done to fit in this context. Such structure allows the transformers to carry inside them a certain level of language understanding, both for the decoder and encoder. Stacking the encoders results in BERT (24 encoders for large BERT and 12 encoders for base BERT), while stacking decoders in GPT (Generative Pre-trained Transformer).

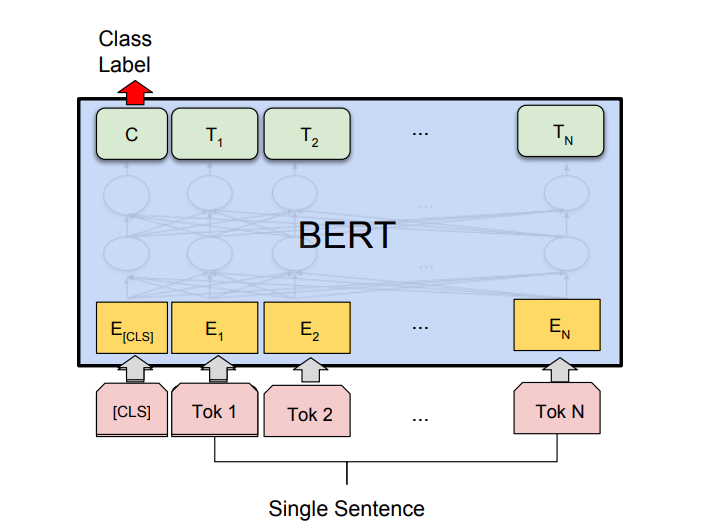
Figure 5.2.1 Transformers Pipeline

BERT achieved state-of-art score in NLP tasks such as:

1. GLUE (General language understanding Evaluation)
2. SWAG (Stanford Question answering dataset)
3. SQuAD (Situations with adversarial generations)

Because of high language understanding, BERT can be used for any NLP task. It is practically a “Pocket knife” for NLP. However, the best results are definitely shown with Masked Language and next sentence prediction tasks because the framework is actually pre-trained on those tasks, predominantly.

Obviously, we could be using the BERT for the classification we need by finetuning it and training it on needed data, using the following pipeline (Figure 5.2.2), which is essentially very similar. The only difference is that the layers are not fully connected with each other, rather they are connected with a fresh set of desired outputs (classes and labels).

Figure 5.2.2 BERT classification

## 

## 5**.**3 NLP Classification and Cosine similarity

A vital thing to consider is the training sets, on which the classifier was initially trained. Typically, any NLP classification task is either:

1. Sentiment analysis, which gets the intended feeling of the text. For example, it predicts the movie viewer’s rating (good, bad, very bad, etc) by their comments about the film.
2. Insult detection, which is straightforward: Is the text insulting and toxic? Used by many companies to find toxic behaviour and ban users automatically without moderation.
3. Intent analysis, which captures the intention of the text. Does the customer want to buy a product? May be complain? Refund?

Topic analysis is present in an extremely low proportion of datasets that can be used to train. And, of course, none of them are Russian.

This led to me using the DeepPavlov basic classifier tasked for sentiment analysis and trained on the Twitter Mokoron (Sentiment unfortunately) dataset, as an experiment. The hopes for feasible results were very low.

For the other approach (tags-to-name) one more thing needs to be done:

Creating embeddings is only the part of the work. We have to be able to compare the words effectively in the obtained vector spaces. Cosine Similarity (or Orchini similarity and Tucker coefficient of congruence) is the mathematical interpretation of words’ similarity. It is defined as the cosine of the angle between the vectors. Specifically, it is calculated by dividing product of the vectors by the product of their length.

# 6. Recommender system implementation

## 6.1 Functional and Non-Functional Criteria

Functional criteria:

● The program and each function execution have a reasonable runtime.

● The program outputs reasonable results and can be applied to solve our problem.

● The program handles the problem of multilingual word embeddings.

Non-Functional criteria:

● The system is scalable. New functionality can be added and the models can be adjusted and trained on new data.

● All the work is done using Python and such libraries as: Pandas, Numpy, Deeppavlov, nltk, gensim.

## 6.2 Initial “Name-to-Tag” Approach

Initially a very straightforward algorithm was introduced to create some sort of categorization.

The algorithm was as follows:

1. Get a set of tags. For each of the tags, the model would output a set of 30-50 most similar tags using fasttext skipgram model with lemmatization on, big corpus: GeoWAC: Population-balanced Russian Gigaword Corpus. Loaded using gensim library.

Table 6.2.1 Most similar wordsИзображение выглядит как текст

Автоматически созданное описание

1. The new collection of most similar words was averaged up and embedded as a single vector respective to the initial tag. That way we could figuratively increase the borders of a “cluster”, by adjusting the tag to a better position, where it is surrounded by the most informative words.
2. Finally, the project title is embedded also as a vector and compared via the cosine similarity with the set of possible tags. That way we are able to assign with a certain threshold probability the best tags to a project.

There are definitely many drawback for such scheme. Firstly, we are not provided with a set of actual tags. Given we decided to work with folksonomy approach, the tags were supposed to be typed by the student with their own preferences. Otherwise, the set of tags must be continually updated, each time adding new tags to the vector space. Secondly, such approach would need the next iteration of the same steps to compare the students’ tags with the projects’ tags. Repetition and more time needed. Moreover, setting the threshold is also very tricky, because the probabilities would differ by very small fractions, assuming we have 100+ possible tags.

This approach was the starting point, however realizing its drawbacks it never appeared to be fully implemented.

## 6.3 DeepPavlov Classifier

The most time was spent trying to create a clear and technically correct model using DeepPavlov models. The goal was also like the first approach: assign the tags to a paper. A lot of different frameworks were tested from DeepPavlov library.

Even though I first wanted to use a BERT approach here again, I decided to use Keras classifier instead, because the benchmarks’ estimates stated it to be even better than BERT. Also there appeared some problem with the BERT implementation, making the outputs not usable.

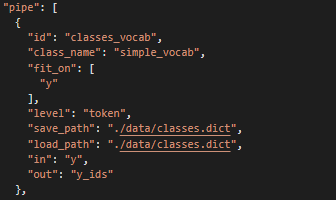
The new algorithm was very straightforward: Keras classifier is simply a CNN built on Keras with 22 layers and 560k parameters.

In order for the model to actually run properly, we need to adjust the config. The config consist of he following parts:

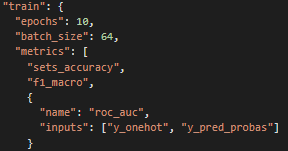
1. Dataset\_reader and Dataset iterator. Defines the data we need to use to train. I was training it on the HabraHabr dataset, its “x” and “y” columns and split the set into 0.9 for train and 0.1 for test.

Изображение выглядит как текст

Автоматически созданное описание

1. Pipe (part of chainer) that specifies the dictionary of classes in the training set and their cooccurance. The pipe also contains the embedder. In our case the embedder was GloVe.
2. Main (part of Pipe) specifies the “guts” of the model: number of filters, optimizer, dropout rate and etc. The activation function is sigmoid, as we need multilabel classification.
3. Train component is responsible for setting the number of epochs and batches used. It also incorporates the metrics for assessment, which however do not work with multilabel classification.Изображение выглядит как текст

   Автоматически созданное описание



The results obtained from the model were horrible. Just as with the BERT model, the effectiveness of assigning the tags is very bad. May be slightly better than random. Throughout manually checking I could not find just 1 correctly assigned tag. Presumably this happens because of very limited data we have.

## 6.4 “Tags-to-Name” Approach

After all the struggling, a new architecture was implemented, which appeared to be the easiest and most performative. This framework is almost the same as the first approach, however the logic is twisted around: instead of assigning a set of tags to a sentence, the model assigned a sentence to a set of tags.

The tags are simply what the student wrote down at the website. The model gets it in a list like format and outputs, for instance, 2 topics (Table 6.4.1).

This topics are the closest to the list of tags presented by the students, which also means that these projects would definitely be interesting to the researcher.

Table 6.4.1 Predicted Project recommendations

Изображение выглядит как текст

Автоматически созданное описание

There is also a need to calculate and assess the model.

Assessment is the problem obviously. In this case I would be using what is called “ручная сверка”. I checked how many of these project’s allocations ended up recommending the original project, this rate is: 23% (10 of 43 were correct)

I also calculated manually, with my own common sense the rate of projects that most certainly wouldn’t be interesting to the student. This rate is: 37% (92 of 249 recommended topics)

The results are far from perfect, however very plausible.

The problems nevertheless may also appear if the students put in too many interests that are little interconnected. For example, it is easy for the model to propose some programming project if the student wrote in: “машинное обучение, программирование”, however if his other interests are also there and these mix up with “политика, история” the intended vector representation may shift drastically, ending up recommending hardly suitable topics.

The theoretical fix that has not yet been tested and implemented is to compare the words between each other using the discussed “most simila” function. Then for example, “машинное обучение, программирование” would be fairly similar to each other and would recommend something based solely on them. “политика, история” would end up in the other subset and make recommendations without programming themes affecting. Although, the model would not be able to find great intersecting projects, for instance, I assume that “Машинное обучение” and “биология” would not be seen as similar and thus not come up with some interesting biotechnological intersection.

Possible solution is going through all possible combinations of tags and looking at the highest probabilities. Given some threshold is carefully calculated it could possibly give relatable recommendations.

# 7. Website Implementation

## 7.1 Basic resources for creating a website:

Flask - A framework for creating web applications in the Python programming language, using the Werkzeug toolset, as well as the Jinja2 template engine. It belongs to the category of so-called microframes - minimalistic frameworks of web applications that consciously provide only the most basic features. The data base for a wep app were created with a help of SQLAlchemy. This is a python package that allows you to process sql queries in the python development environment.

All the work is done using the Python and the Flask library and Bootstrap for frontend development.

Our project greatly simplifies the choice of a project for students, as well as

interaction of teachers and students

## 7.2 My goals

1. Create an authentication system

2. Create a database of users and projects

3. Create a profile for a student and a teacher

4. Allow teachers to add projects and students to apply for them

5. Frontend design for a web application

6. Provide an extension option to use models of my teammates

## 7.3 Technical basis

It was decided to use the Flask framework to develop a web application. And to use SQL lite as a database. Web pages are implemented in HTML.

## 

## 7.4 Functional and Non-functional requirements

**Non-Functional requirements:**

1. Helping students choose a project using deep networks

2. Organize all projects

3. Help students interact with teachers

**Functional requirements:**

1. The system sends a request with data to the database when registering new users as well as adding projects.

2. Checks the profile type and based on this gives permission to publish the project

3. Creates a database for projects and users, and links them together

4. Performs full-fledged search using specialized search engines

5. Recommends projects based on the data that the student indicates about himself

## 7.5 Implementation

I chose Pycharm as the IDE.

Implemented authorization for users with the separation of profile types into student and teacher.

Added an opportunity for students to fill out data about themselves and use this data for the recommendation system in the future

Added an opportunity for the teacher to add a project, all projects created by the teacher are displayed in his profile. The main page shows all the projects

Tables have been created in the database for User and Project

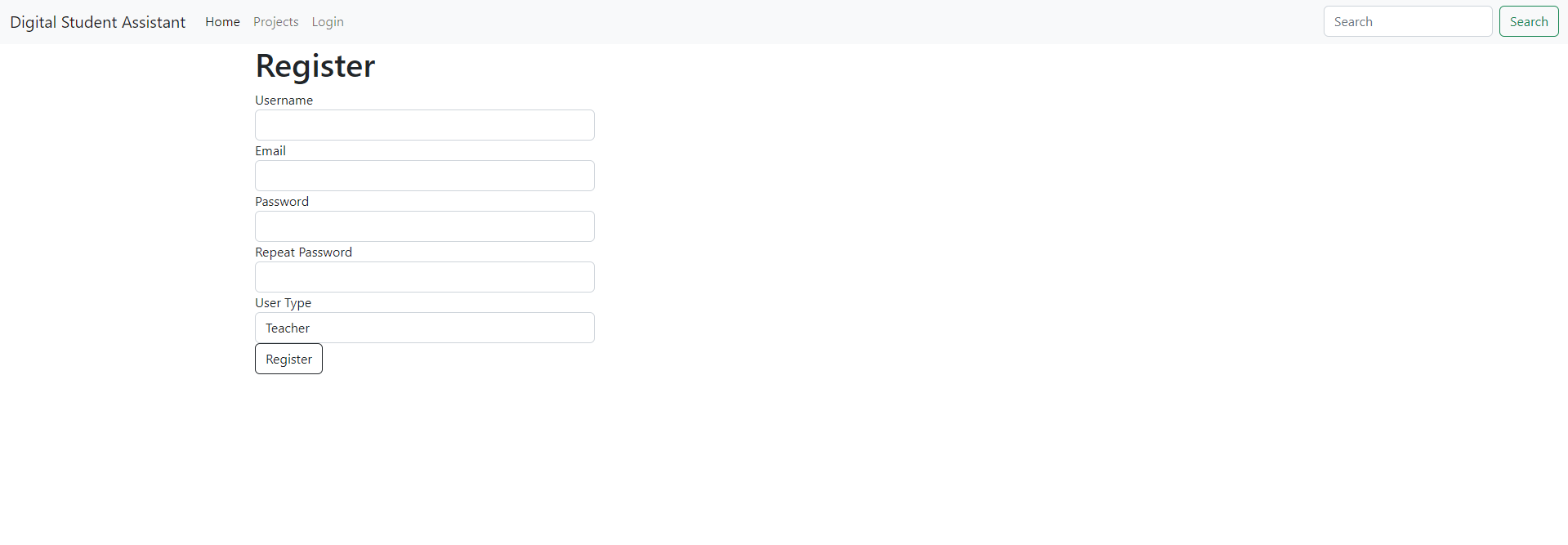
Delving into the details, I created an authorization page for web application users, on the backend side there is a check for all possible errors from the client, that is, there is no way to enter anything in the email field, for example. All web site pages are not available, while the user is not logged in.

Figure 7.5.1

After the user has registered, he has the opportunity to log in using his email and password, all passwords are stored in encrypted form, for security reasons

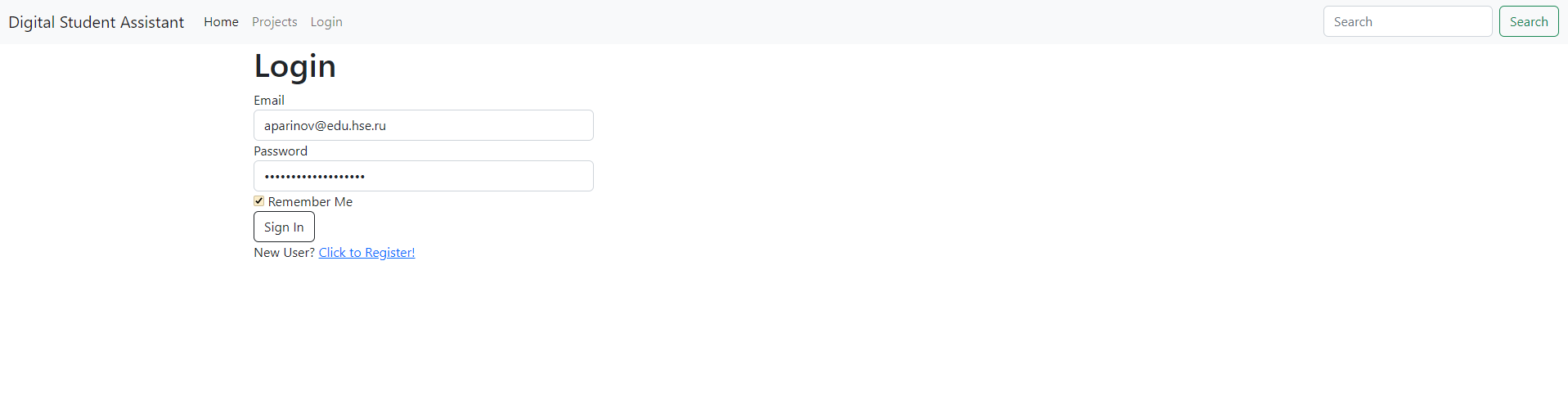


Figure 7.5.1

Then the user gets to the main page where all projects are displayed

Here the student can sign up for the project or unsubscribe from the project if he changed his mind or received confirmation from the teacher for another project

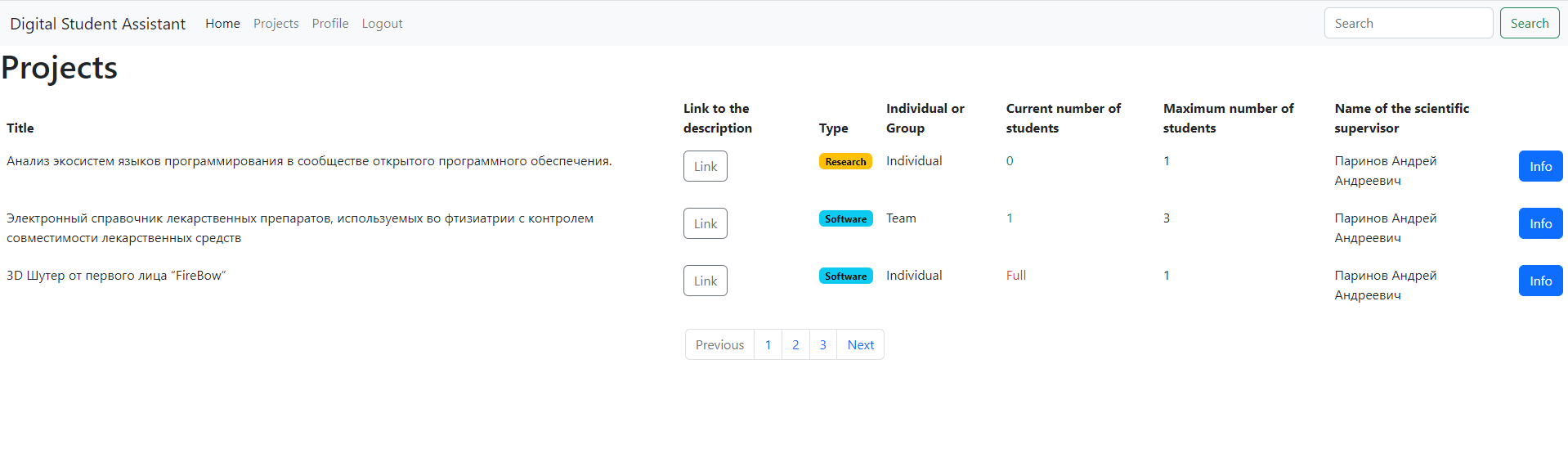


Figure 7.5.3

Here is the table with all projects, if the number of students are equals to the maximum, then the full sign is displayed and student are not avaliable to make request for this project. Students can click on Info button and get to the project page.

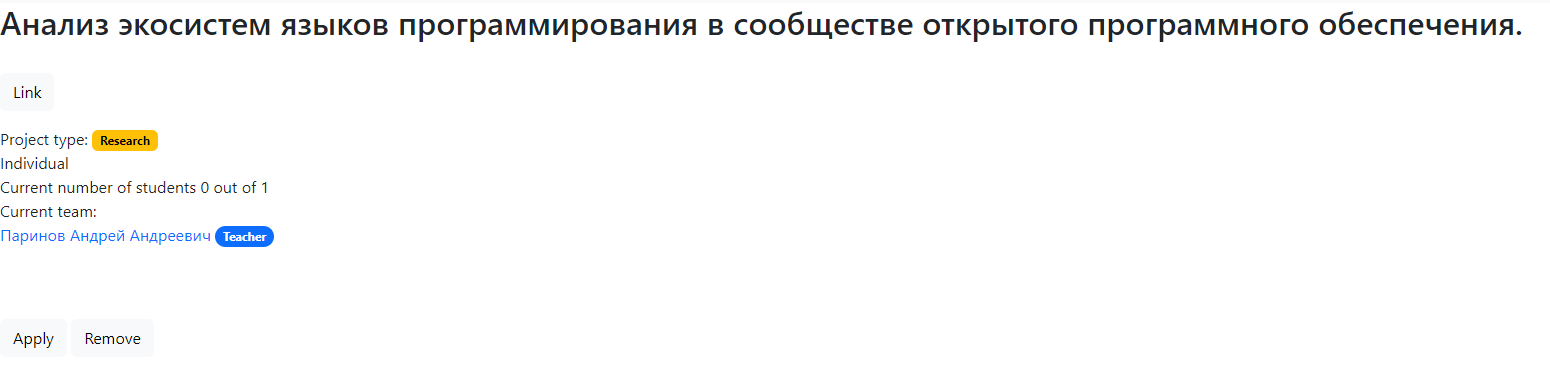


Figure 7.5.4



Figure 7.5.5

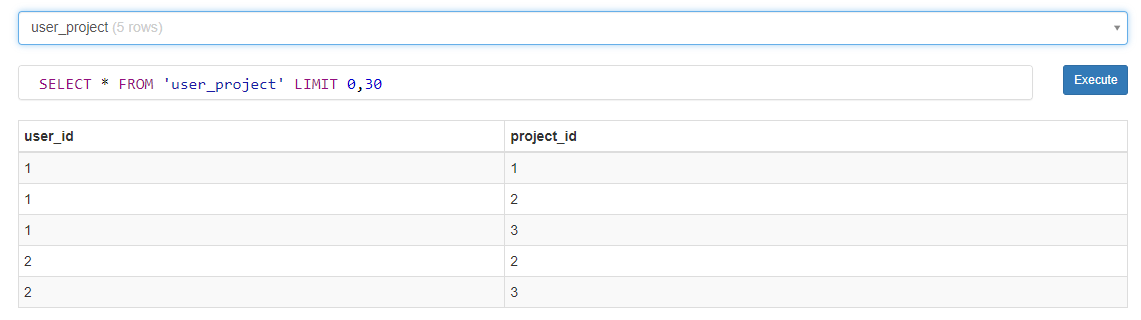
When the student click on apply button the new ralation in the database is creating, and now this student is related with with project and added to the Current team. All possible errors have been checked, and only accounts with the student type can apply for a project, it will be impossible for a teacher to do this. This is how it looks like in the database

Figure 7.5.6

The user can view the full information on the project by clicking on the link button, it will forward it to a Google document.

The user can see all his projects for which he has applied in his profile. For a user with a teacher account type, it is possible to add a project, and this project will be added to the database

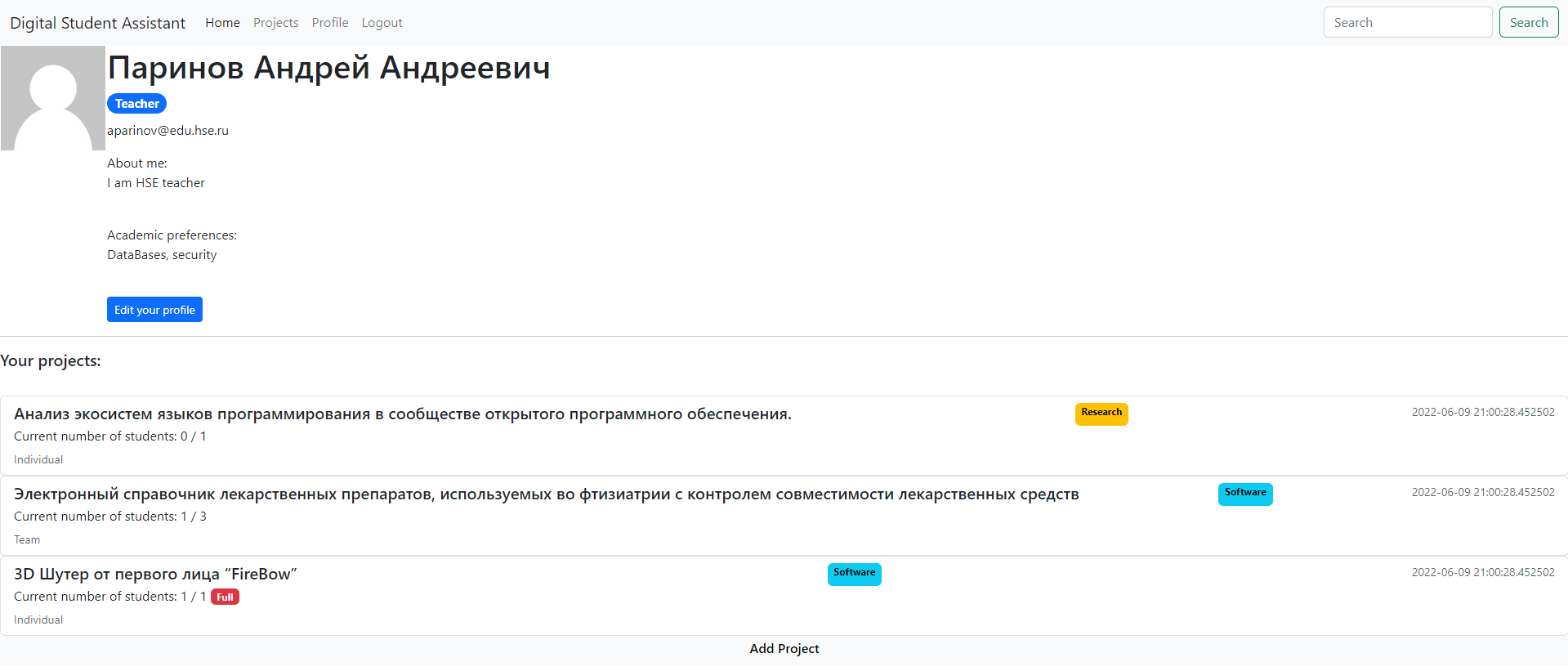


Figure 7.5.7

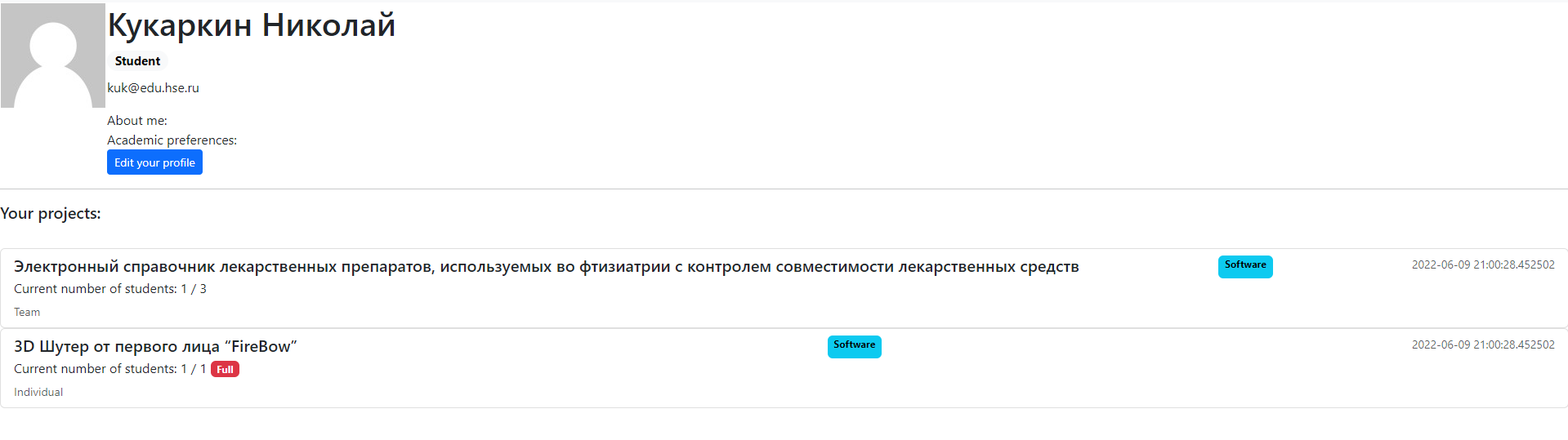
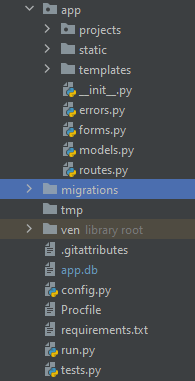
This is how profile page looks like. Photos for avatars will be uploaded from a third-party Gravatar service to save space on the server. From profile page students and teachers can control their projects.

Figure 7.5.8

Users are able to edit their profiles, information about a student is taken from his profile, for example academic preferences. And in the future, the recommendation system, based on this information, recommends projects for students.

In addition to everything, there is an input and search window at the top right. This is done so that students can search by title for projects that interest them. This is implemented using existing search solutions.

Now we can talk about the backend component

This is an application structure

Projects: projects blueprint with all routes and templates(html files) that are needed for projects

Static: all files that won’t change(images for example or css files)

Templates: in this folder all html pages are stored

\_\_init\_\_.py where all nedeed for applications attributes are added to the app.

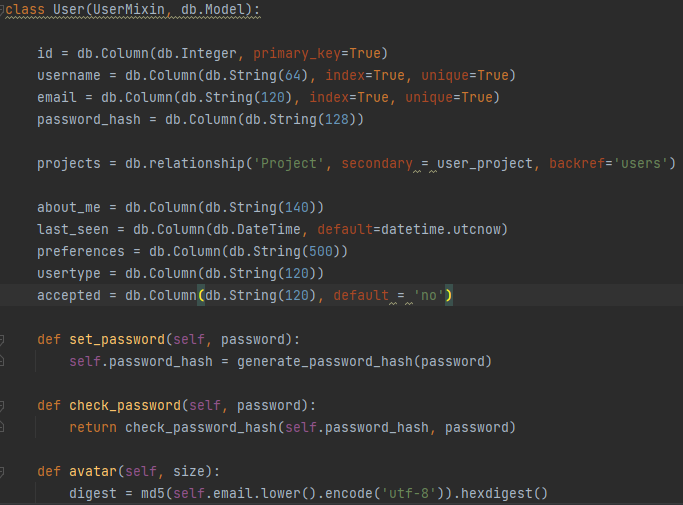
errors.py i decided to make a separate file for all errors that are possible during the user's interaction with the site(404 or 505 erros)

Figure 7.5.9

forms.py this is where all forms are located, when user type something on the web site this is all handled here. Registration for example

The web page has two post requests and a get request from the server, and taking information from the database.From handling from python package Flask\_wtf were used.

The next module is models.py. In this file database is initialized and all relations are stored.



# 8. Visual Analytics

## 8.1 Comparative analysis of Business Intelligence Tools

There are tons of different options on the market. Business Intelligence tools are presented from various companies, leaders in the IT technology market. All of them vary in reliability, integration capabilities, ease-of-use and of course, pricing. Here is a comparative analysis of BI tools:

*Microsoft Power BI*

Microsoft Power Business Intelligence is a BI platform that specializes in visualizing data. It contains well developed connectors which gives you an opportunity to improve your work in campaigns and allows users to discover real-time trends. Power BI may be accessible from almost anywhere using any device, as it is web-based. This software allows users to integrate your own apps and solutions and offer reports and real-time dashboards.

There is a free version available, which does not have any time constraints, however it is limited to 1GB of data per user. Otherwise, 10GB of storage costs $10 per user per month.

*SAS Business Intelligence*

SAS BI is most known for its superior predictive analytics, however it also has a strong business intelligence solution. This well-established self-service technology helps users to make meaningful business conclusions by leveraging databases and different metrics. SAS provides users with a variety of customization choices through their APIs. It considers providing both powerful analytics and reporting and high-level data integration. They also have a fantastic text analytics option that provides you with extra context information about the data.

SAS BI is a cloud-based enterprise analysis solution that allows users to manage interactive reports and monitor metrics. The platform's features include a configurable dashboard, marketing reports, forecasting, data source connectors, ad-hoc analysis, and more. It's designed for large enterprises. The data visualization tool in SAS Business Intelligence allows users to perform automated analysis and construct interactive graphics to enhance competitive advantages. The solution's collaboration module also allows several users to collaborate on the same project using Microsoft Office products like Outlook, PowerPoint, and Excel.

There is no free version (only a free trial), and the starting fee is $8000 per year.

*Tableau*

Tableau is a BI and analytics tool that assists in the analysis of essential business data and the generation of actionable insights. The technology enables businesses to compile data from different of sources, for instance SQL databases, spreadsheets, and cloud apps into a single dataset. Tableau's live visual analytics and interactive dashboard allow you to slice and dice statistics to find new insights and possibilities. Users can construct interactive maps and analyze data from a variety of sources, including geographies, territories, demographics, and more. Tableau assists in the creation of a narrative story of data analysis through interactive graphics that can be shared with an audience. Tableau can be tailored to meet the needs of a variety of industries, including finance, communication, education, healthcare, real estate, manufacturing, and technology. The solution can be implemented on-premises or hosted as a SaaS application in the cloud. The Tableau Mobile software allows users to see and analyze data on their mobile phones with any OS.

Free version is available for students.

*Oracle BI*

Oracle Business Intelligence is a cloud-based service that uses predictive analytics to help small to large businesses obtain insights into their performance and make better management solutions. An administrative dashboard is included with the centralized platform, allowing users to aggregate input from numerous data sources and adjust the generated results. Versioning, data archiving, a self-service portal, and alerts/notifications are all essential aspects of Oracle Business Intelligence Suite. It gives businesses the tools they need to communicate key business goals across areas and track progress with scorecards. Users can utilize the solution to access existing data in the system and create financial, production, and interactive reports based on important parameters. Enterprises can use Oracle BI Suite to monitor user activity and get system warnings in order to run a multi-step analytical workflow.

Pricing is available by request.

*MicroStrategy*

MicroStrategy business intelligence program is a BI platform that enables both working with databases and construct visualization objects. Its functionality is available on-premises and in the cloud as well. The application is used both for personal needs and large enterprises deployment. MicroStrategy BI solution supports dashboards that combine functionality of different applications, such as monitoring markets and portfolios, risk analysis, searching for news and social media and matching it to real-time analysis. Clients can use the solution to offer information in the form of operating reports, ad hoc reports, bills, and statements. Drag-and-drop feature allows users to create operational reports and statements. Multiple reporting styles and devices are supported by the MicroStrategy analytics platform, allowing customers to access their data in the way that the company requires.

The product is available with a metered or named-user pricing model. There is no free version (only a free trial), and the first cost is $600.

*SAP Business Objects*

SAP Business Objects BI program allows to create comprehensive reports, analyze data, and visualize it interactively. The platform integrates customer relationship management (CRM), digital supply chain, Enterprise Resource Planning (ERP), and other areas. This platform's role-based dashboards are particularly enticing since they allow users to construct unique solutions and apps. SAP is a comprehensive software that delivers a variety of functions on a single platform and serves all sectors from IT to management. The product's complexity, on the other hand, increases the price, making it not a public software.

*Yellowfin BI*

Yellowfin BI is a business intelligence platform focuses on visualization and machine learning. Simple filtering with the use of checkboxes and radio buttons may also be used to swiftly sort through massive data sets, and dashboards are accessible in real time from almost anywhere. This platform is compatible with browser and works on both desktops and mobile devices. The great thing about this business intelligence solution is that you can easily transform dashboards and visualizations to one step up without special skills using a low code development environment.

In our particular case I would better use Tableau or Microsoft Power BI software, based on comparative analysis above and assigned goals. However, while working with Power BI I have faced a problem with importing data from PostgreSQL database, which we used for storing data. So, it was decided to provide further analytics using Tableau program.

## 8.2 Functional and Non-Functional criteria

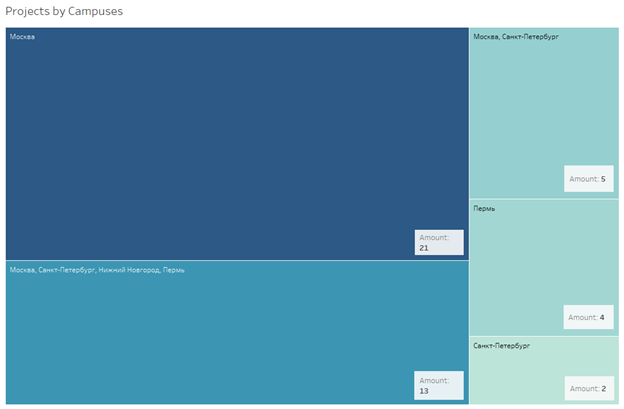
### Functional criteria

* Visual analytics of the gathered data, which provides insights on the projects, draws conclusions on plenitude of projects variety and helps to get an overview of the projects by different parameters.

### Non-functional criteria

* Perform a comparative analysis of business intelligence tools and create visual analytics using one of them.

## 8.3 Visual Analytics Implementation

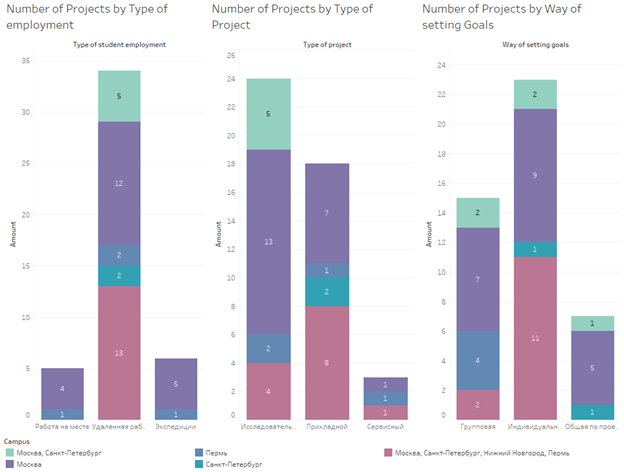


Dashboard 8.3.1

# 

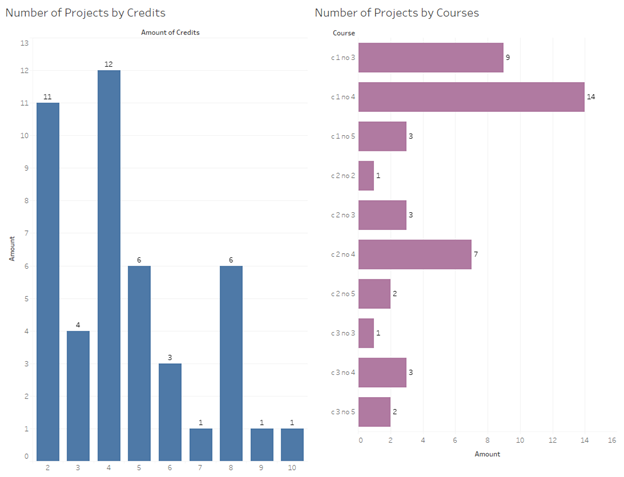
Dashboard 8.3.2

First two dashboards contain an overall analysis of number of projects and vacancies by campuses. Its quite logical that Moscow campus is the leader in both cases, as it is the biggest HSE campus. However, an interesting fact is that Saint-Petersburg campus is inferior in numbers to the Perm campus. Furthermore, we can see that some campuses are grouped, so we are not provided with exact data. Also, its clear that the data we are working with is not fulfill according to given numbers.



Dashboard 8.3.3

The third dashboard above depicts number of projects by the following categories: type of employment, type of project and way of setting goals, all subdivided by campus as well. The most left stacked bar chart shows us that the most popular type of student employment with a great advantage is “Удаленная работа”, It is present in all campuses. While types “Работа на месте” and “Экспедиции” are available only in Moscow and Perm campuses in a very limited number. The key insight is that it would be better to diversify projects by type of employment. The bar chart in the middle visualizes the same parameters, but the primary categorical variable is type of project. In this case the distribution of projects a lot better, two most popular types are “Исследовательский” and “Прикладной”. And the last bar chart illustrates the number of projects by way of setting goals. “Индивидуальная” goal setting is leading by number, followed by “Групповая” and “Общая по проекту”. After the overall analysis the main idea is that it is very important to increase the number of projects in some categories. Moreover, some types of projects are not even available in Saint-Petersburg campus for example. Its crucial to extend the variety of projects there.



Dashboard 8.3.4

The last dashboard emphasizes the distribution of projects by the amount of credits and courses. From the bar chart on the left we could conclude that most of the projects give from 2 to 5 credits. Horizontal bars on the right depict that the vast majority of the projects is available from 1st to 3rd and from 1st to 4th courses. Also, some of the bars contain 5th course, but it relates not to all studying programmes, so the numbers are rather small. The key insight is that almost all projects are expected to be done due to last courses (4th and 5th), and it is very important, because diversity and availability of the projects is vital especially for students who are doing final term work.

# 9. Key Conclusions and future plans

The overall work was extensive. The algorithms presented are simple, however considering the lack of training info produce plausible results that can be possibly applied in real life.

There are a lot of things that can be done next. Obviously, collecting and mining more data with descriptions and tags and training the intent task models to get better results. Also, connecting the back-end algorithm with the web service created by the team and putting it together to obtain a final usable product. Also, as a futher development we can use API МИЭМа to get even more projects, and to do that we need a way to authorise into it with token, which we do not have now.

We can also propose this as the final graduation project or a side project as summer practice.

# 

# 10. Bibliography

[1] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł. & Polosukhin I. (2017) Attention Is All You Need. arXiv: arXiv:1706.03762

[2] Software Advice, Software Advice Inc., 2006-2022, [www.softwareadvice.com](http://www.softwareadvice.com)

[3] Mopinion, 2022, [www.mopinion.com](http://www.mopinion.com)

[4] NLPL word embeddings repository, Nordic Language Processing Laboratory, 2022,

[vectors.nlpl.eu/repository](http://vectors.nlpl.eu/repository/)

[5] DeepPavlov’s documentation, Neural Networks and Deep Learning lab, MIPT Revision eddb4454, 2018, [docs.deeppavlov.ai/en/master/index.html](http://docs.deeppavlov.ai/en/master/index.html)

[6] The Flask Mega-Tutorial [Miguel Grinberg](https://blog.miguelgrinberg.com/author/Miguel%20Grinberg) <https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>

[7] SQLalchemy <https://flask-sqlalchemy.palletsprojects.com/en/2.x/index.html>

[8] Selenium <https://selenium-python.readthedocs.io/>

# 11. Appendix

GitHub links to the repository with the code: <https://github.com/Kukarkin/Digital-Student-Assistant-Recommender>

<https://github.com/eliray01/DigitalStudentAssistant>

<https://github.com/arted2/HSE2022>

<https://github.com/arsruts/CourseWork2022>