

Отчёт по реализации индивидуального проекта. Часть 6

Дисциплина: Архитектура ЭВМ

Перегудов Александр Вадимович

Содержание

1	Цель работы	6
2	Задание	7
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
5	Выводы	28
	Список литературы	29

Список иллюстраций

4.1	Папка i18n	9
4.2	Языковые файлы в папке i18n	9
4.3	Файл menus.ru.yaml	10
4.4	Перевод меню	10
4.5	Добавил русский язык и указал папку с контентом для этого языка	11
4.6	Директории en и ru	11
4.7	Пост previous week 4	12
4.8	Файл index.md	12
4.9	Пост на английском языке	13
4.10	Пост Programming languages	13
4.11	Часть содержимого файла index.md	14
4.12	Часть содержимого файла index.md	14
4.13	Часть содержимого файла index.md	15
4.14	Пост на английском языке	15
4.15	Директория en/	16
4.16	Часть содержимого файла _index.md	16
4.17	Часть содержимого файла _index.md	17
4.18	Часть содержимого файла _index.md	17
4.19	Часть содержимого файла _index.md	18
4.20	Часть содержимого файла _index.md	18
4.21	Часть содержимого файла _index.md	19
4.22	Часть содержимого файла _index.md	19
4.23	Часть содержимого файла _index.md	20
4.24	Команда для сборки сайта	20
4.25	Команда для сборки сайта	21
4.26	Сайт на английском языке	21
4.27	Сайт на русском языке	22
4.28	Директория post	22
4.29	Директория previous week 5	23
4.30	Изменённый файл index.md	23
4.31	Директория Software architecture	24
4.32	Новая картинка	24
4.33	Часть содержимого файла index.md	25
4.34	Директория Software architecture	25
4.35	Часть содержимого файла index.md	26
4.36	Часть поста на русском языке	26

4.37 Часть поста на английском языке	27
--	----

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . .	8
-----	---	---

1 Цель работы

Приобрести навыки настройки нескольких языков на примере конструктора Hugo.

2 Задание

Сделать поддержку английского и русского языков.

3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

4 Выполнение лабораторной работы

Перешёл в корневую папку с сайтом и создал директорию i18n (рис. 4.1).

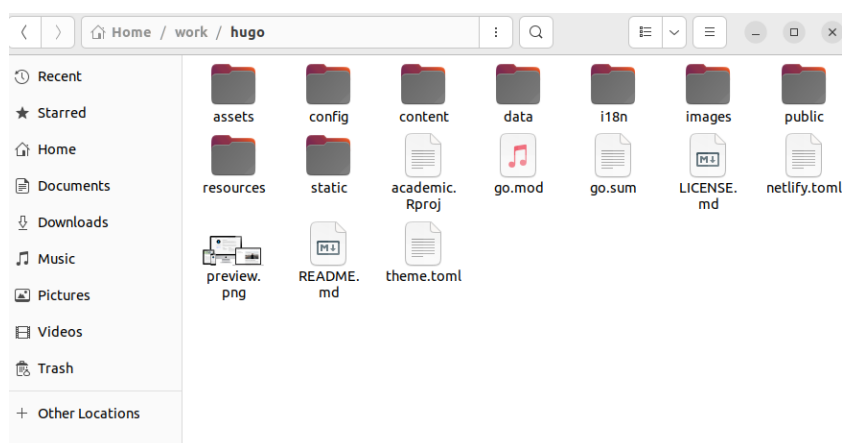


Рис. 4.1: Папка i18n

Скачал языковые файлы для русского и английского языков и скопировал их в директорию i18n (рис. 4.2).

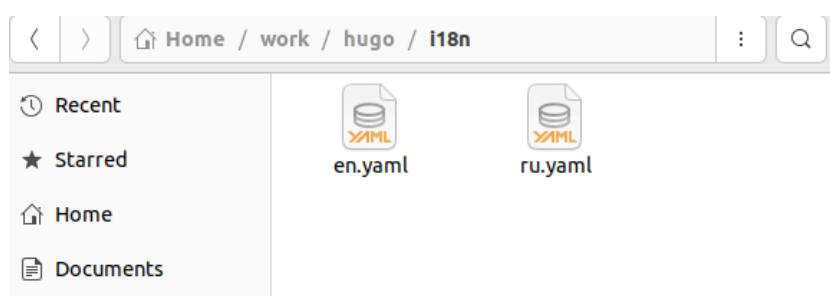


Рис. 4.2: Языковые файлы в папке i18n

Перешёл в директорию _default, скопировал файл menus.yaml и переименовал его в menus.ru.yaml (рис. 4.3).

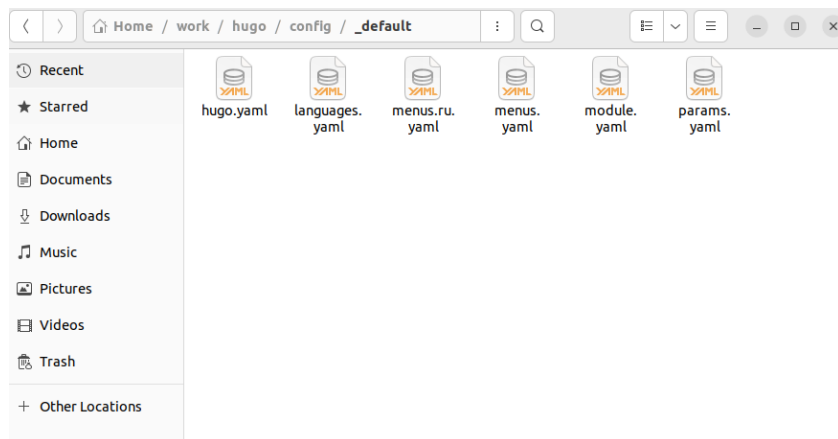


Рис. 4.3: Файл menus.ru.yaml

Изменил файл menus.ru.yaml (рис. 4.4).

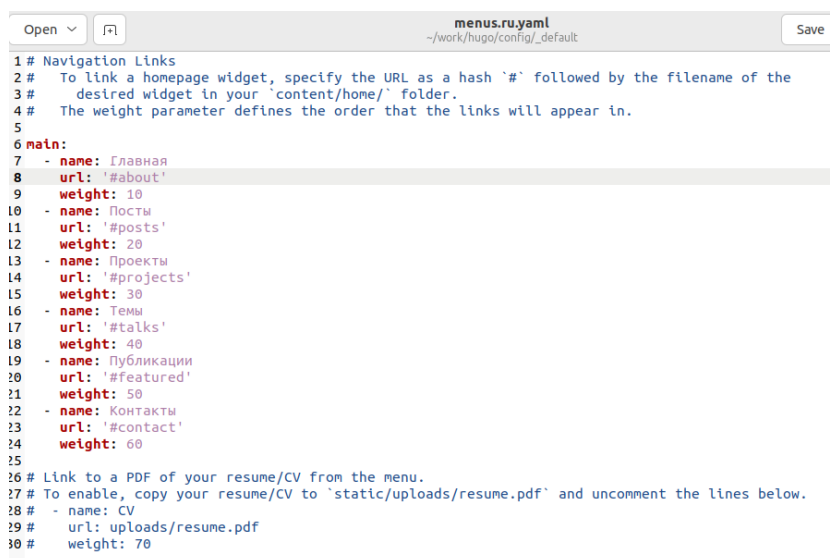
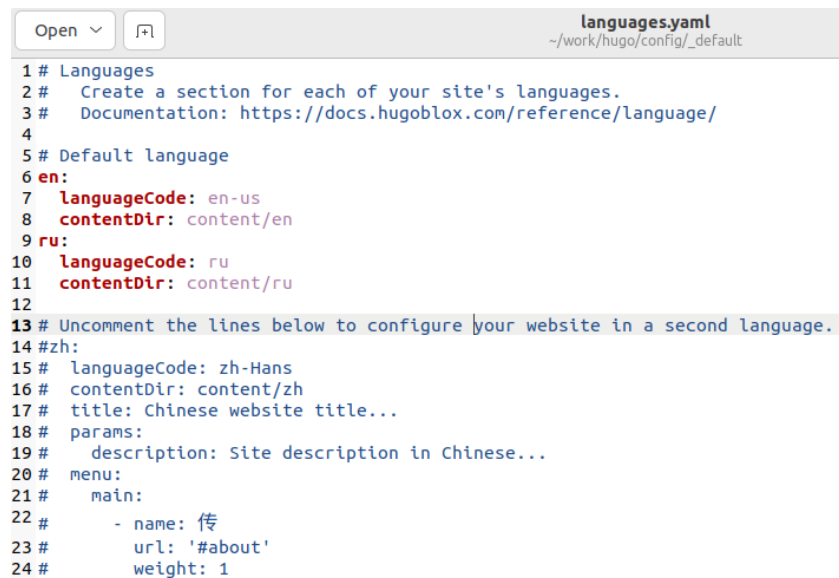


Рис. 4.4: Перевод меню

В той же директории изменил файл languages.yaml (рис. 4.5).



```

1 # Languages
2 #   Create a section for each of your site's languages.
3 #   Documentation: https://docs.hugoblox.com/reference/language/
4
5 # Default language
6 en:
7   LanguageCode: en-us
8   contentDir: content/en
9 ru:
10  LanguageCode: ru
11  contentDir: content/ru
12
13 # Uncomment the lines below to configure your website in a second language.
14 #zh:
15 #   LanguageCode: zh-Hans
16 #   contentDir: content/zh
17 #   title: Chinese website title...
18 #   params:
19 #     description: Site description in Chinese...
20 #   menu:
21 #     main:
22 #       - name: 传
23 #         url: '#about'
24 #         weight: 1

```

Рис. 4.5: Добавил русский язык и указал папку с контентом для этого языка

Перешёл в директорию content, создал ru и en директории и скопировал все файлы и директории в content в эти директории (рис. 4.6).

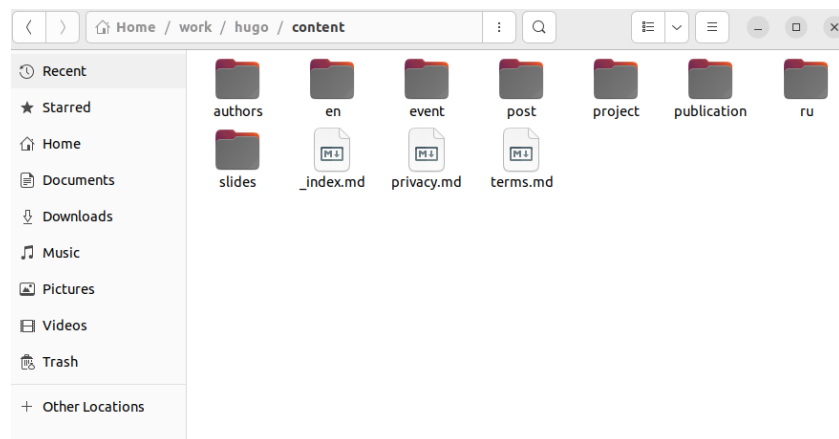


Рис. 4.6: Директории en и ru

Перешёл в директорию en/post/previous week 4 (рис. 4.7).

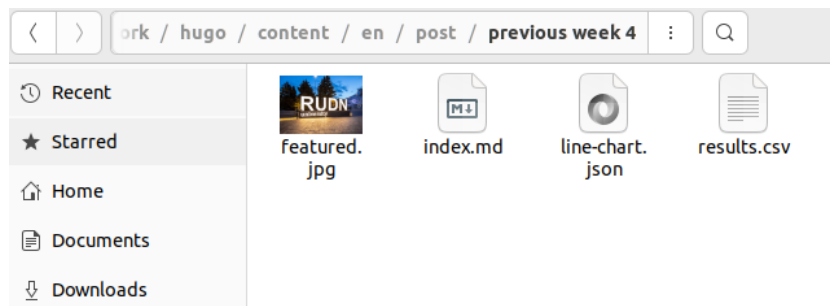


Рис. 4.7: Пост previous week 4

Изменил файл index.md для английской версии сайта (рис. 4.8).

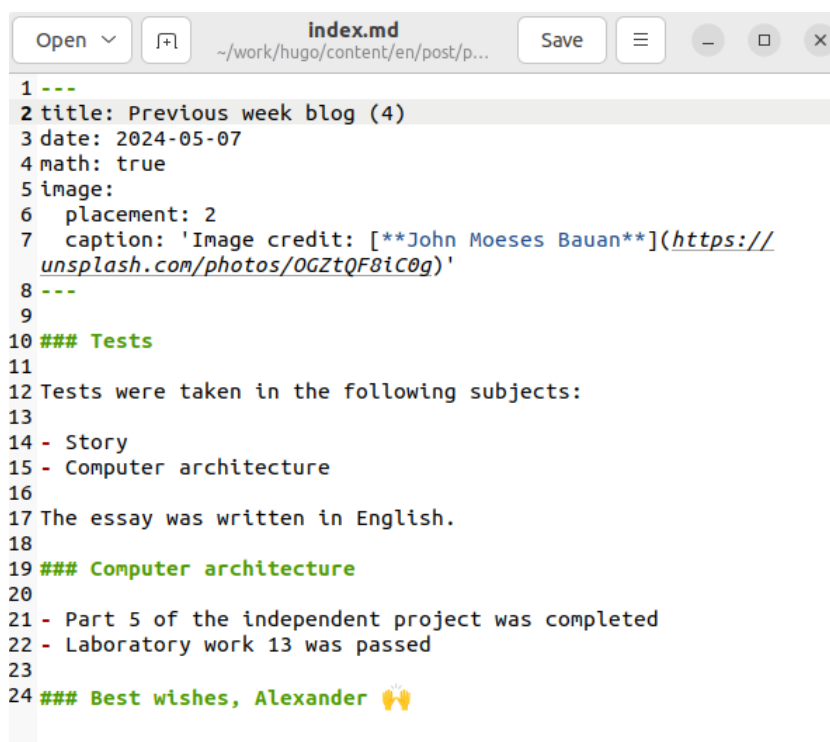


Рис. 4.8: Файл index.md

Открыл сайт и проверил изменения (рис. 4.9).

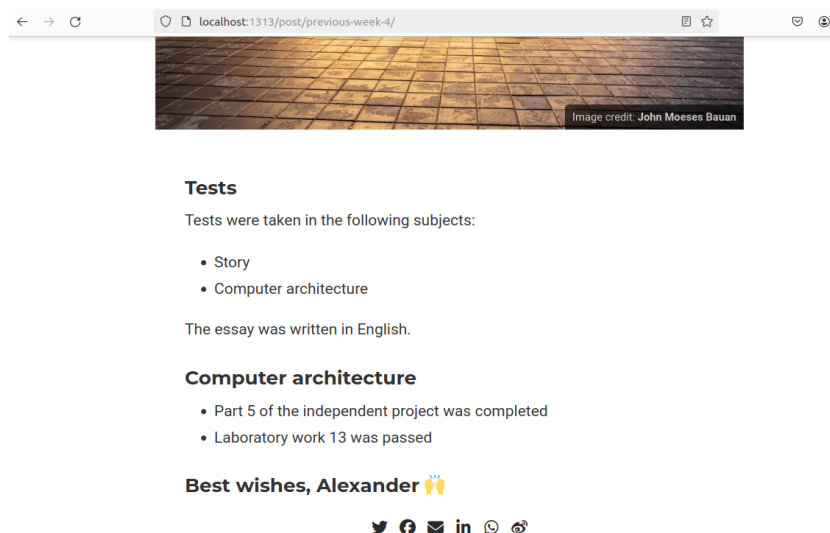


Рис. 4.9: Пост на английском языке

Перешёл в директорию en/post/Scientific Programming languages (рис. 4.10).

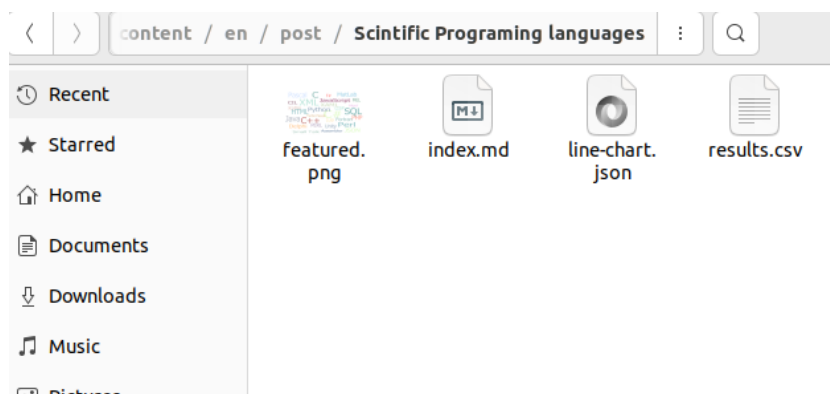
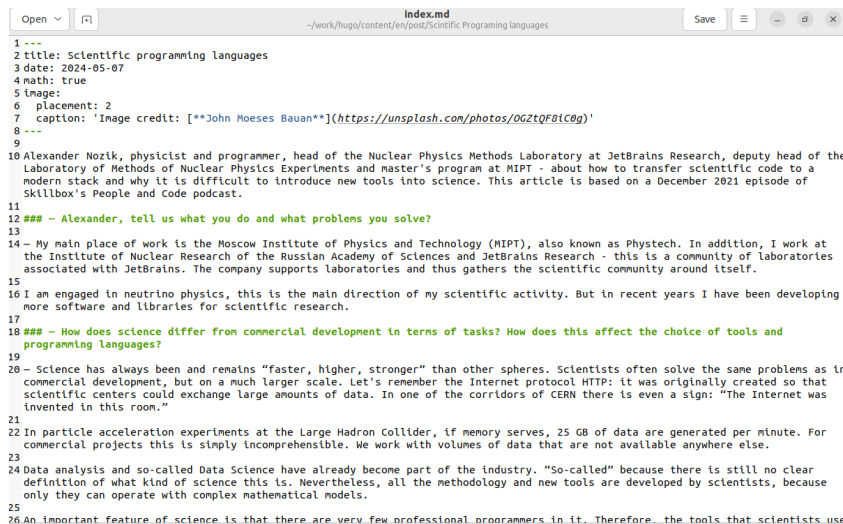


Рис. 4.10: Пост Programming languages

Изменил файл index.md для английской версии сайта (рис. 4.11, 4.12, 4.13).

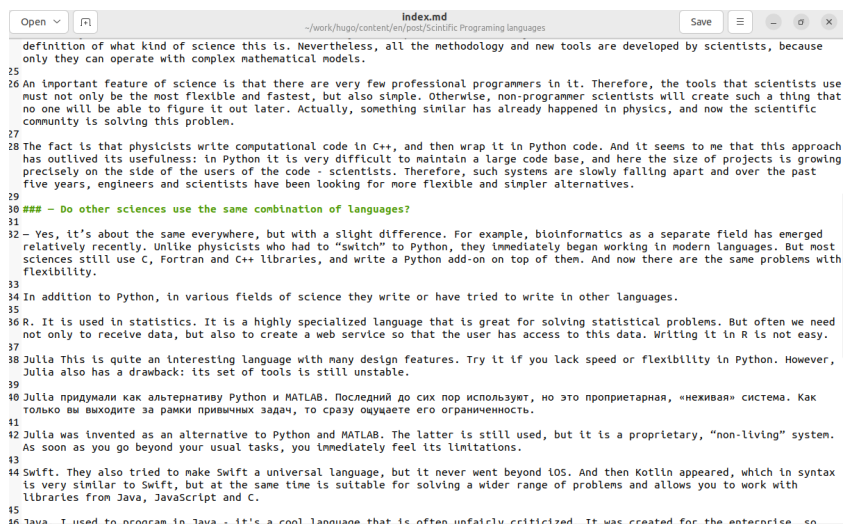


```

1 ---
2 title: Scientific programming languages
3 date: 2024-05-07
4 math: true
5 images:
6   placement: 2
7   caption: 'Image credit: [**John Moeses Bauan**](https://unsplash.com/photos/OGZtQF0iC0g)'
8 ---
9
10 Alexander Nozik, physicist and programmer, head of the Nuclear Physics Methods Laboratory at JetBrains Research, deputy head of the
11   Laboratory of Methods of Nuclear Physics Experiments and master's program at MIPT - about how to transfer scientific code to a
12   modern stack and why it is difficult to introduce new tools into science. This article is based on a December 2021 episode of
13   Skillbox's People and Code podcast.
14
15 ## - Alexander, tell us what you do and what problems you solve?
16
17 - My main place of work is the Moscow Institute of Physics and Technology (MIPT), also known as Phystech. In addition, I work at
18   the Institute of Nuclear Research of the Russian Academy of Sciences and JetBrains Research - this is a community of laboratories
19   associated with JetBrains. The company supports laboratories and thus gathers the scientific community around itself.
20
21 I am engaged in neutrino physics, this is the main direction of my scientific activity. But in recent years I have been developing
22   more software and libraries for scientific research.
23
24 ## - How does science differ from commercial development in terms of tasks? How does this affect the choice of tools and
25   programming languages?
26
27 - Science has always been and remains "faster, higher, stronger" than other spheres. Scientists often solve the same problems as in
28   commercial development, but on a much larger scale. Let's remember the Internet protocol HTTP: it was originally created so that
29   scientific centers could exchange large amounts of data. In one of the corridors of CERN there is even a sign: "The Internet was
30   invented in this room."
31
32 In particle acceleration experiments at the Large Hadron Collider, if memory serves, 25 GB of data are generated per minute. For
33   commercial projects this is simply incomprehensible. We work with volumes of data that are not available anywhere else.
34
35 Data analysis and so-called Data Science have already become part of the industry. "So-called" because there is still no clear
36   definition of what kind of science this is. Nevertheless, all the methodology and new tools are developed by scientists, because
37   only they can operate with complex mathematical models.
38
39 An important feature of science is that there are very few professional programmers in it. Therefore, the tools that scientists use

```

Рис. 4.11: Часть содержимого файла index.md



```

25 definition of what kind of science this is. Nevertheless, all the methodology and new tools are developed by scientists, because
26   only they can operate with complex mathematical models.
27
28 An important feature of science is that there are very few professional programmers in it. Therefore, the tools that scientists use
29   must not only be the most flexible and fastest, but also simple. Otherwise, non-programmer scientists will create such a thing that
30   no one will be able to figure it out later. Actually, something similar has already happened in physics, and now the scientific
31   community is solving this problem.
32
33 The fact is that physicists write computational code in C++, and then wrap it in Python code. And it seems to me that this approach
34   has outlived its usefulness: in Python it is very difficult to maintain a large code base, and here the size of projects is growing
35   precisely on the side of the users of the code - scientists. Therefore, such systems are slowly falling apart and over the past
36   five years, engineers and scientists have been looking for more flexible and simpler alternatives.
37
38 ## - Do other sciences use the same combination of languages?
39
40 - Yes, it's about the same everywhere, but with a slight difference. For example, bioinformatics as a separate field has emerged
41   relatively recently. Unlike physicists who had to "switch" to Python, they immediately began working in modern languages. But most
42   sciences still use C, Fortran and C++ libraries, and write a Python add-on on top of them. And now there are the same problems with
43   flexibility.
44
45 In addition to Python, in various fields of science they write or have tried to write in other languages.
46
47 R. It is used in statistics. It is a highly specialized language that is great for solving statistical problems. But often we need
48   not only to receive data, but also to create a web service so that the user has access to this data. Writing it in R is not easy.
49
50 Julia This is quite an interesting language with many design features. Try it if you lack speed or flexibility in Python. However,
51   Julia also has a drawback: its set of tools is still unstable.
52
53 Julia придумали как альтернативу Python и MATLAB. Последний до сих пор используют, но это проприетарная, «неживая» система. Как
54   только вы выходите за рамки привычных задач, то сразу ощущаете его ограниченность.
55
56 Julia was invented as an alternative to Python and MATLAB. The latter is still used, but it is a proprietary, "non-living" system.
57   As soon as you go beyond your usual tasks, you immediately feel its limitations.
58
59 Swift. They also tried to make Swift a universal language, but it never went beyond iOS. And then Kotlin appeared, which in syntax
60   is very similar to Swift, but at the same time is suitable for solving a wider range of problems and allows you to work with
61   libraries from Java, JavaScript and C.
62
63 Java - I used to program in Java - it's a cool language that is often unfairly criticized. It was created for the enterprise so

```

Рис. 4.12: Часть содержимого файла index.md

```

45
46 Java. I used to program in Java - it's a cool language that is often unfairly criticized. It was created for the enterprise, so
47 there is an overly drawn-out "ceremony": to build a simple application, you need to write a lot of additional code. Yes, this
48 simplifies support and increases the stability of the application, but it greatly complicates the programming process itself.
49
50 Kotlin. Now I write a lot in Kotlin. It has the advantages of Java, but relieves the programmer of a good half of the "ceremonies",
51 and therefore, in my opinion, it has great prospects.
52 What are the main problems in scientific IT and how are they trying to solve them?
53
54 ### - You mentioned that there was a lot of fancy stuff in science. I wonder if this only applies to Fortran and C++ libraries or
55 something else? And what are the main problems there?
56
57 This is a very big problem that we will have to sort out for a long time." The fact is that one day development turned into a
58 separate professional field. 40 years ago, when programming was just in its infancy, physicists were the coolest programmers. They
59 worked on huge machines the size of a closet, and everyone understood everything. But 15-20 years ago it turned out that software
60 engineering is an independent engineering field.
61
62 Scientists fell far behind because they continued to develop software in the same style in which they did it on the antediluvian
63 mainframes. They came up with wonderful algorithms, but did not think at all about code support, testing, CI/CD, and especially
64 about architecture.
65
66 For example, data collection systems are completely unable to interact with each other. They were written by different programmers,
67 who sometimes specifically made it so that the systems could not communicate with each other. If two experiments are performed on
68 different systems, then the scientist will have to process their results separately. Moreover, it can take several years to study
69 each system - they are so complex.
70
71 Due to the fact that science did not implement modern approaches to software design and did not use new tools, we have accumulated
72 millions of legacy lines. There you can find good and even brilliant algorithms, but it will be almost impossible to understand
73 them. To feel the pain, find a Fortran 77 program and try to understand what's going on there.
74
75 ### For reference: in Fortran 77, variable name length is limited to eight characters. This means that when all matching variable
76 names run out, the program becomes unreadable. Number-letter identifiers are used, with which the code only becomes more confusing.
77 But most programs in science are still written in Fortran.
78
79 Relatively recently, scientists realized that they need to invest in software engineering, pay attention to architecture and master
80 modern tools, and not fence their bicycles. To my surprise, over the past two years, many groups of engineers have emerged that are
81 purposefully promoting the modern approach in science.
82
83
84
85 ### Best wishes, Alexander 🙏

```

Рис. 4.13: Часть содержимого файла index.md

Открыл сайт и проверил изменения (рис. 4.14).

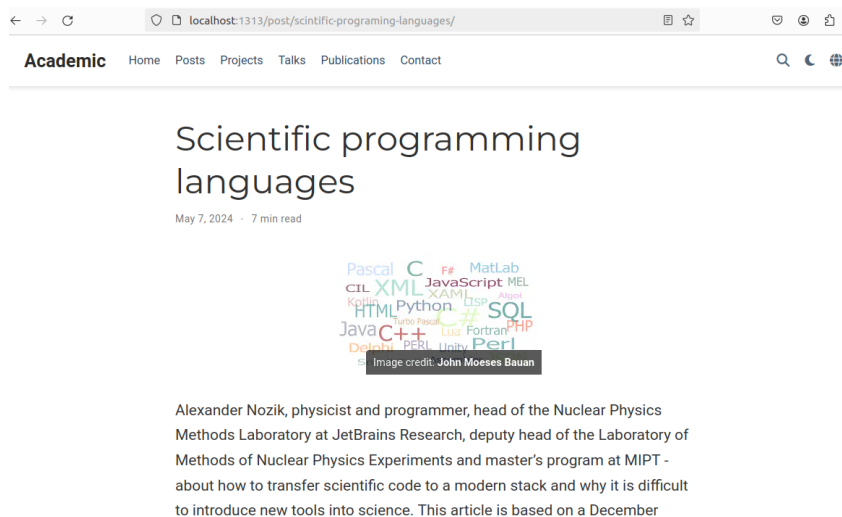


Рис. 4.14: Пост на английском языке

После я изменил все посты, публикации и события. Процесс аналогичен двум выше а постов много. Для примера я описал изменения двух постов, но на самом деле изменил все. Для русской версии сайта тоже самое.

Перешёл в директорию en/ (рис. 4.15).

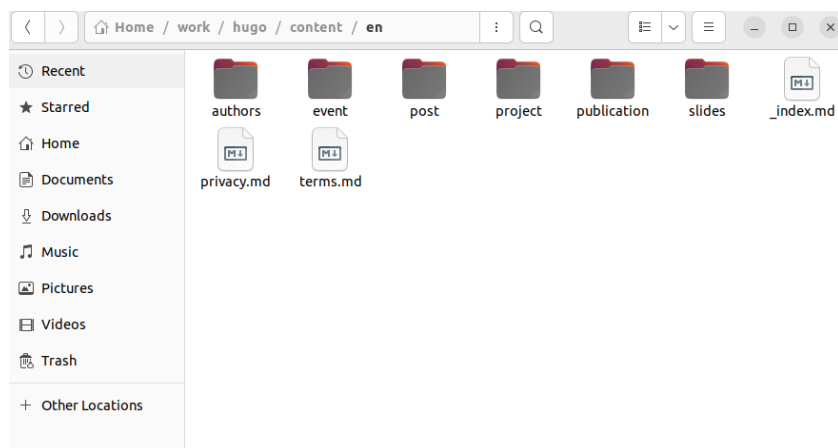


Рис. 4.15: Директория en/

Открыл и изменил файл `_index.md` (рис. 4.16, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.23).

```

1 |---
2 # Leave the homepage title empty to use the site title
3 title: ''
4 date: 2022-10-24
5 type: landing
6
7 sections:
8   - block: hero
9     demo: true # Only display this section in the Hugo Blox Builder demo site
10    content:
11      title: Hugo Academic Theme
12      image:
13        filename: hero-academic.png
14      cta:
15        label: '**Get Started**'
16        url: https://hugoblox.com/templates/
17      cta_alt:
18        label: Ask a question
19        url: https://discord.gg/z8wNVzb
20      cta_note:
21        label: >-
22      <div style="text-shadow: none;"><a class="github-button" href="https://github.com/HugoBlox/hugo-blox-builder" data-
23        icon="octicon-star" data-size="large" data-show-count="true" aria-label="Star">Star Hugo Blox Builder</a></div><div style="text-
24        shadow: none;"><a class="github-button" href="https://github.com/HugoBlox/theme-academic-cv" data-icon="octicon-star" data-
25        size="large" data-show-count="true" aria-label="Star">Star the Academic template</a></div>
26      text: |
27        **Generated by Hugo Blox Builder - the FREE, Hugo-based open source website builder trusted by 500,000+ sites.**
28
29        **Easily build anything with blocks - no-code required!**
30
31        From landing pages, second brains, and courses to academic resumés, conferences, and tech blogs.
32
33      <!-- Custom spacing -->
34      <div class="mb-3"></div>
35      <!-- GitHub Button JS -->
36      <script async defer src="https://buttons.github.io/buttons.js"></script>
37    design:
38      background:
39        gradient: end: '#1976d2'

```

Рис. 4.16: Часть содержимого файла `_index.md`


```
Open ▾ [ ] _index.md ~/work/hugo/content/en Save [≡] [⏮] [⏭] [X]
33 <script async defer src="https://buttons.github.io/buttons.js"></script>
34 design:
35   background:
36     gradient_end: '#1976d2'
37     gradient_start: '#004ba0'
38     text_color_light: true
39 - block: about.biography
40 id: about
41 content:
42   title: @biography
43   # Choose a user profile to display (a folder name within 'content/authors/')
44   username: admin
45 - block: skills
46 content:
47   title: Skills
48   text: ''
49   # Choose a user to display skills from (a folder name within 'content/authors/')
50   username: admin
51 design:
52   columns: '1'
53 - block: experience
54 content:
55   title: Experience
56   # Date format for experience
57   # Refer to https://docs.hugoblox.com/customization/#date-format
58   date_format: Jan 2006
59   # Experiences.
60   # Add/remove as many 'experience' items below as you like.
61   # Required fields are 'title', 'company', and 'date_start'.
62   # Leave 'date_end' empty if it's your current employer.
63   # Begin multi-line descriptions with YAML's '|2-' multi-line prefix.
64   items:
65     - title: Worker
66       company: VingeComp
67       company_url: ''
68       company_logo: suitcase
69       location: Russia
70       date_start: '2021-06-23'
71       date_end: '2023-10-10'
```

Рис. 4.17: Часть содержимого файла _index.md

```
Open ▾ [ ] _index.md ~/work/hugo/content/en Save [≡] [⏮] [⏭] [X]
68   company_logo: suitcase
69   location: Russia
70   date_start: '2021-06-23'
71   date_end: '2023-10-10'
72   description: |2-
73     Responsibilities include:
74
75     * Analysing
76     * Construct
77     * Deploying
78     * Translation
79 - title: Student
80   company: University RUDN
81   company_url: ''
82   company_logo: school
83   location: Russia
84   date_start: '2023-09-01'
85   date_end: ''
86   description: Programming and softarchtecture. Implenenting several python projects.
87 design:
88   columns: '2'
89 - block: accomplishments
90 content:
91   # Note: 'ashy:' is used to add a 'soft' hyphen in a long heading.
92   title: 'Accomplish&ashy;ments'
93   subtitle:
94   # Date format: https://docs.hugoblox.com/customization/#date-format
95   date_format: Jan 2006
96   # Accomplishments.
97   # Add/remove as many 'item' blocks below as you like.
98   # 'title', 'organization', and 'date_start' are the required parameters.
99   # Leave other parameters empty if not required.
100  # Begin multi-line descriptions with YAML's '|2-' multi-line prefix.
101  items:
102    - certificate_url: https://github.com/magister6239/NltsheGame
103      date_end: '2025-10-10'
104      date_start: '2021-01-25'
105      description: ''
106  ...
```

Рис. 4.18: Часть содержимого файла _index.md

```
Open  ~index.md  Save  [Icons]
~/work/hugo/content/en

103   date_end: '2025-10-10'
104   date_start: '2021-01-25'
105   description: ''
106   icon: python
107   organization: Github
108   organization_url: https://github.com/magister6239
109   title: Python project
110   url: ''
111   - certificate_url: https://github.com/magister6239/study_2023-2024_os-intro
112   date_end: ''
113   date_start: '2021-01-01'
114   description: Whole work
115   icon: sitenap
116   organization: Github
117   organization_url: https://github.com/magister6239
118   title: Computer architecture
119   url: ''
120
121   design:
122     columns: '2'
123   - block: collection
124     id: posts
125     content:
126       title: Recent Posts
127       subtitle: ''
128       text: ''
129       # Choose how many pages you would like to display (0 = all pages)
130       count: 5
131       # Filter on criteria
132       filters:
133         folders:
134           - post
135         authors: ''
136         category: ''
137         tag: ''
138         exclude_featured: false
139         exclude_future: false
140         exclude_past: false
141         publication_type: ''
```

Рис. 4.19: Часть содержимого файла _index.md

```
Open  ~index.md  Save  [Icons]
~/work/hugo/content/en

138   exclude_featured: false
139   exclude_future: false
140   exclude_past: false
141   publication_type: ''
142   # Choose how many pages you would like to offset by
143   offset: 0
144   # Page order: descending (desc) or ascending (asc) date.
145   order: desc
146   design:
147     # Choose a layout view
148     view: compact
149     columns: '2'
150   - block: portfolio
151     id: projects
152     content:
153       title: Projects
154       filters:
155         folders:
156           - project
157       # Default filter index (e.g. 0 corresponds to the first 'filter_button' instance below).
158       default_button_index: 0
159       # Filter toolbar (optional).
160       # Add or remove as many filters ('filter_button' instances) as you like.
161       # To show all items, set 'tag' to ''.
162       # To filter by a specific tag, set 'tag' to an existing tag name.
163       # To remove the toolbar, delete the entire 'filter_button' block.
164       buttons:
165         - name: All
166           tag: ''
167         - name: Deep Learning
168           tag: Deep Learning
169         - name: Other
170           tag: Demo
171
172   design:
173     # Choose how many columns the section has. Valid values: '1' or '2'.
174     columns: '1'
175     view: showcase
176     # For Showcase view, flip alternate rows?
```

Рис. 4.20: Часть содержимого файла _index.md

```

173 columns: '1'
174 view: showcase
175 # For Showcase view, flip alternate rows?
176 flip_alt_rows: false
177 - block: markdown
178 content:
179   title: Gallery
180   subtitle: ''
181   text: |-
182     {{< gallery album="demo" >}}
183 design:
184   columns: '1'
185 - block: collection
186 id: featured
187 content:
188   title: Featured Publications
189   filters:
190     folders:
191       - publication
192     featured_only: true
193 design:
194   columns: '2'
195   view: card
196 - block: collection
197 content:
198   title: Recent Publications
199   text: |-
200     {{% callout note %}}
201     Quickly discover relevant content by [filtering publications](./publication/).
202     {{% /callout %}}
203   filters:
204     folders:
205       - publication
206     exclude_featured: true
207 design:
208   columns: '2'
209   view: citation
210 - block: collection
211 ...

```

Рис. 4.21: Часть содержимого файла _index.md

```

208 columns: '2'
209 view: citation
210 - block: collection
211 id: talks
212 content:
213   title: Recent & Upcoming Talks
214   filters:
215     folders:
216       - event
217 design:
218   columns: '2'
219   view: compact
220 - block: tag_cloud
221 content:
222   title: Popular Topics
223 design:
224   columns: '2'
225 - block: contact
226 id: contact
227 content:
228   title: Contact
229   subtitle:
230   text: |-
231     Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam mi diam, venenatis ut magna et, vehicula efficitur enim.
232     # Contact (add or remove contact options as necessary)
233     email: test@example.org
234     phone: 888 888 88 88
235     appointment_url: 'https://calendly.com'
236     address:
237       street: 450 Serra Mall
238       city: Stanford
239       region: CA
240       postcode: '94305'
241       country: United States
242       country_code: US
243     directions: Enter Building 1 and take the stairs to Office 200 on Floor 2
244     office_hours:
245       - 'Monday 10:00 to 13:00'
246       - 'Tuesday 09:00 to 18:00'
247       - 'Wednesday 09:00 to 18:00'
248       - 'Thursday 09:00 to 18:00'
249       - 'Friday 09:00 to 18:00'
250       - 'Saturday 09:00 to 18:00'
251       - 'Sunday 09:00 to 18:00'

```

Рис. 4.22: Часть содержимого файла _index.md

```

238 city: Stanford
239 region: CA
240 postcode: '94305'
241 country: United States
242 country_code: US
243 directions: Enter Building 1 and take the stairs to Office 200 on Floor 2
244 office_hours:
245   - 'Monday 10:00 to 13:00'
246   - 'Wednesday 09:00 to 10:00'
247 # Choose a map provider in 'params.yaml' to show a map from these coordinates
248 coordinates:
249   latitude: '37.4275'
250   longitude: '-122.1697'
251 contact_links:
252   - icon: twitter
253     icon_pack: fab
254     name: DM Me
255     link: 'https://twitter.com/Twitter'
256   - icon: skype
257     icon_pack: fab
258     name: Skype Me
259     link: 'skype:echo123?call'
260   - icon: video
261     icon_pack: fas
262     name: Zoom Me
263     link: 'https://zoom.com'
264 # Automatically link email and phone or display as text?
265 autolink: true
266 # Email form provider
267 form:
268   provider: netlify
269   formspree:
270     id:
271     netlify:
272     # Enable CAPTCHA challenge to reduce spam?
273     captcha: false
274 design:
275   columns: '2'
276 ---

```

Рис. 4.23: Часть содержимого файла `_index.md`

Тоже самое я сделал и для русской версии сайта. Процесс аналогичен этому. Русская версия сайта находится в директории `content/ru`. Пересобрал сайт и Запустил процесс для работы сайта (рис. 4.24, 4.25).

```

avperegudov@avperegudov: ~/work/hugo
Environment: "development"
Serving pages from disk
Running in Fast Render Mode. For full rebuilds on change: hugo server --disableFastRender
Web Server is available at http://localhost:1313/ (bind address 127.0.0.1)
Press Ctrl+C to stop
avperegudov@avperegudov:~/work/hugo$ ~/bin/hugo
Start building sites ...
hugo v0.124.1-d8083b05f16c945fec04f745f0ca8640560cf1ec+extended linux/amd64 BuildDate=2024-03-20T11:40:10Z VendorInfo=gohugoio

| EN | RU |
-----+-----+-----
Pages | 62 | 61 |
Paginator pages | 1 | 1 |
Non-page files | 43 | 46 |
Static files | 9 | 9 |
Processed images | 91 | 84 |
Aliases | 15 | 14 |
Cleaned | 0 | 0 |

Total in 687 ms
avperegudov@avperegudov:~/work/hugo$

```

Рис. 4.24: Команда для сборки сайта

```
avperegudov@avperegudov:~/work/hugo$ ~/bin/hugo server
Watching for changes in /home/avperegudov/{.cache,work}
Watching for config changes in /home/avperegudov/work/hugo/config/_default, /home/avperegudov/.cache/hugo_cache/modules/filecache/modules/pkg/mod/github.com/hugo!blox/hugo-blox-builder/modules/blox-plugin-netlify@v1.1.2-0.20231108141515-0478cf6921f9/config.yaml, /home/avperegudov/.cache/hugo_cache/modules/filecache/modules/pkg/mod/github.com/hugo!blox/hugo-blox-builder/modules/blox-plugin-reveal@v1.1.2/config.yaml, /home/avperegudov/.cache/hugo_cache/modules/filecache/modules/pkg/mod/github.com/hugo!blox/hugo-blox-builder/modules/blox-bootstrap/v5@v5.9.7/hugo.yaml, /home/avperegudov/work/hugo/go.mod
Start building sites ...
hugo v0.124.1-d8083b05f16c945fec04f745f0ca8640560cf1ec+extended linux/amd64 BuildDate=2024-03-20T11:40:10Z VendorInfo=gohugoio
```

	EN	RU
Pages	62	61
Paginator pages	1	1
Non-page files	43	46
Static files	9	9
Processed images	91	84
Aliases	15	14
Cleaned	0	0

Рис. 4.25: Команда для сборки сайта

Открыл сайт и проверил изменения (рис. 4.26, 4.27).

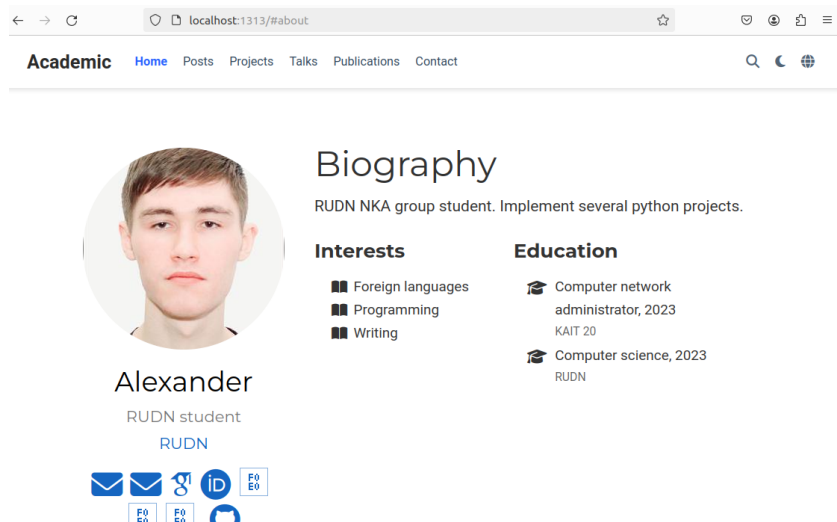


Рис. 4.26: Сайт на английском языке

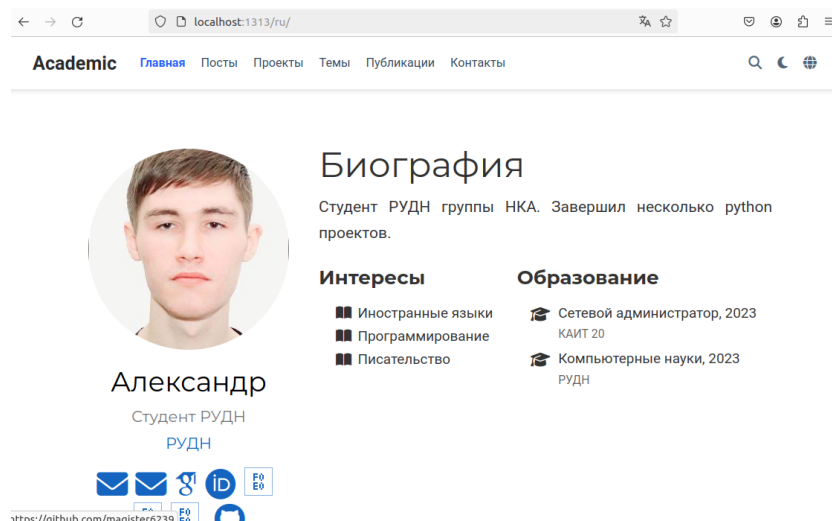


Рис. 4.27: Сайт на русском языке

Перешёл в директорию `content/ru/post` (рис. 4.28).

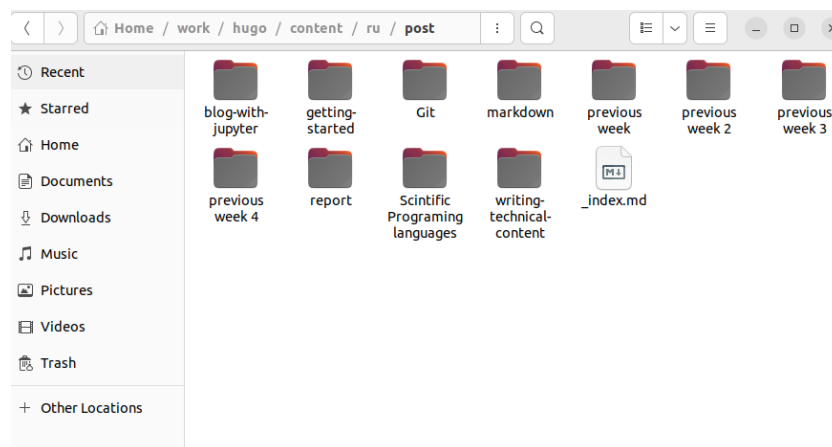


Рис. 4.28: Директория post

Скопировал директорию `previous week 4` и переименовал её в `previous week 5` (рис. 4.29).

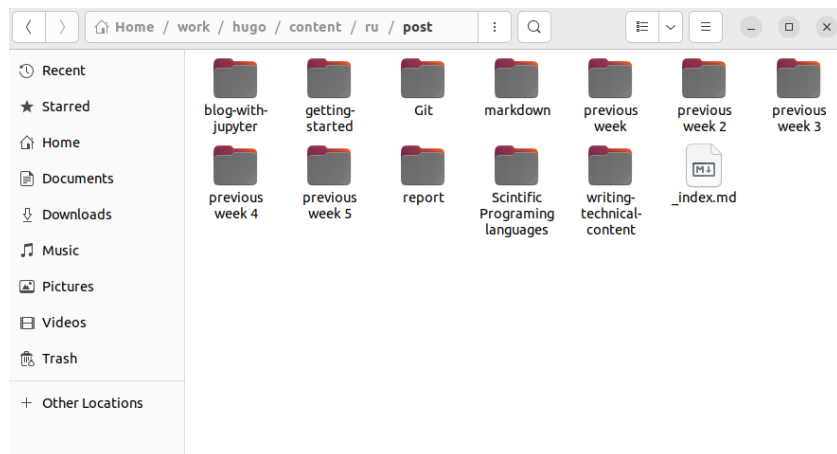


Рис. 4.29: Директория previous week 5

Изменил файл index.md в директории previous week 5 (рис. 4.30).

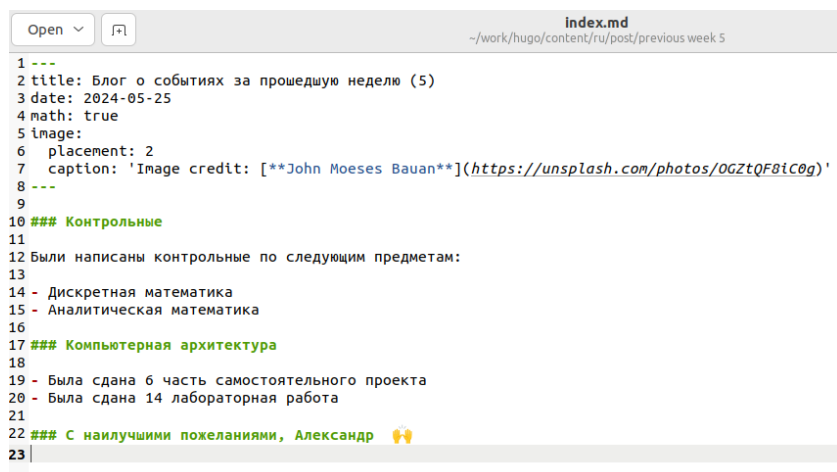


Рис. 4.30: Измененённый файл index.md

В той же директории скопировал директорию Git и переименовал её в Software architecture. (рис. 4.31).

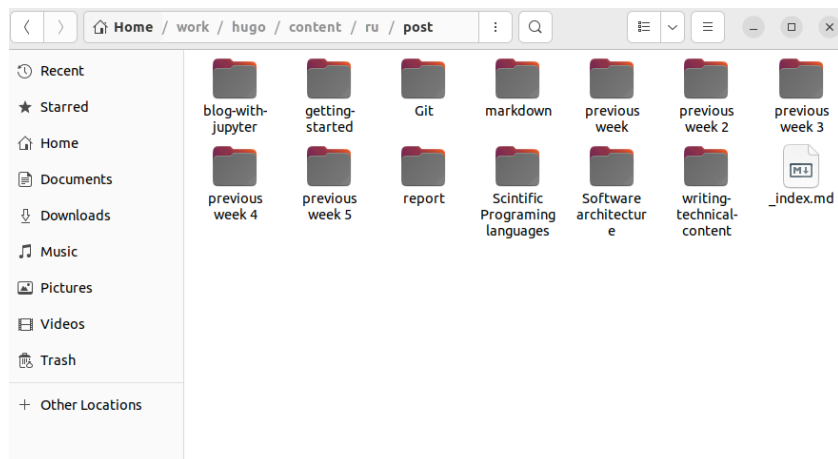


Рис. 4.31: Директория Software architecture

Перешёл в директорию Software architecture и заменил картинку. (рис. 4.32).

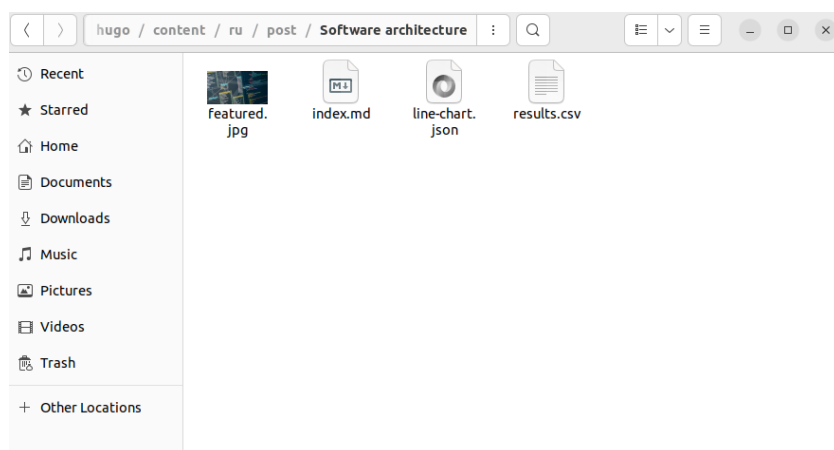
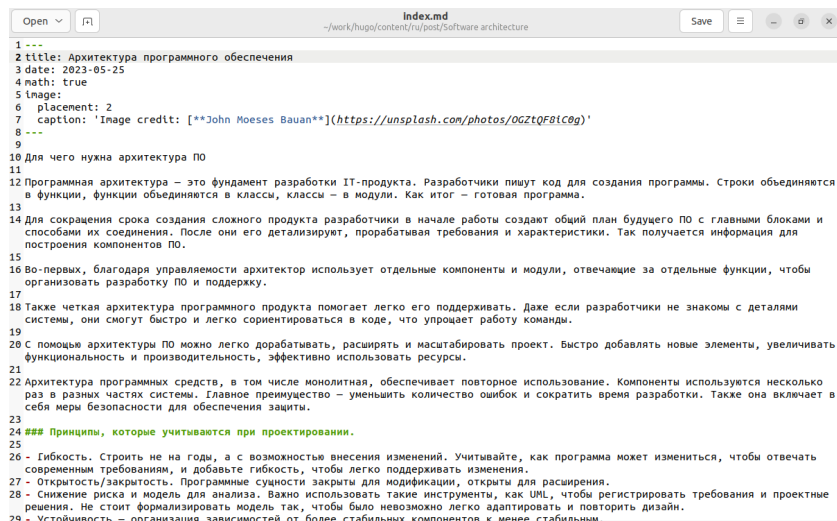


Рис. 4.32: Новая картинка

Открыл и изменил файл index.md (рис. 4.33).



```
1 ---
2 title: Архитектура программного обеспечения
3 date: 2023-05-25
4 math: true
5 image:
6   placement: 2
7   caption: 'Image credit: [**John Moeses Bauan**](https://unsplash.com/photos/OGZtQf8tC0g)'
8 ---
9
10 Для чего нужна архитектура ПО
11
12 Программная архитектура – это фундамент разработки IT-продукта. Разработчики пишут код для создания программы. Строки объединяются
13 в функции, функции объединяются в классы, классы – в модули. Как итог – готовая программа.
14
15 Для сокращения срока создания сложного продукта разработчики в начале работы создают общий план будущего ПО с главными блоками и
16 способами их соединения. После они его детализируют, прорабатывая требования и характеристики. Так получается информация для
17 построения компонентов ПО.
18
19 Во-первых, благодаря управляемости архитектор использует отдельные компоненты и модули, отвечающие за отдельные функции, чтобы
20 организовать разработку ПО и поддержку.
21
22 Также четкая архитектура программного продукта помогает легко его поддерживать. Даже если разработчики не знакомы с деталями
23 системы, они смогут быстро и легко сориентироваться в коде, что упрощает работу команды.
24
25 С помощью архитектуры ПО можно легко дорабатывать, расширять и масштабировать проект. Быстро добавлять новые элементы, увеличивать
26 функциональность и производительность, эффективно использовать ресурсы.
27
28 Архитектура программных средств, в том числе монолитная, обеспечивает повторное использование. Компоненты используются несколько
29 раз в разных частях системы. Главное преимущество – уменьшить количество ошибок и сократить время разработки. Также она включает в
30 себя меры безопасности для обеспечения защиты.
31
32 ### Принципы, которые учитываются при проектировании.
33
34 - Гибкость. Строить не на годы, а с возможностью внесения изменений. Учитывайте, как программа может измениться, чтобы отвечать
35 современным требованиям, и добавляйте гибкости, чтобы легко поддерживать изменения.
36
37 - Открытость/закрытость. Программные сущности закрыты для модификации, открыты для расширения.
38
39 - Снижение риска и модель для анализа. Важно использовать такие инструменты, как UML, чтобы регистрировать требования и проектные
40 решения. Не стоит формализовать модель так, чтобы было невозможно легко адаптировать и повторить дизайн.
41
42 - Устойчивость – организация зависимостей от более стабильных компонентов к менее стабильным.
```

Рис. 4.33: Часть содержимого файла index.md

Скопировал директорию Software architecture в директорию content/en/post (рис. 4.34).

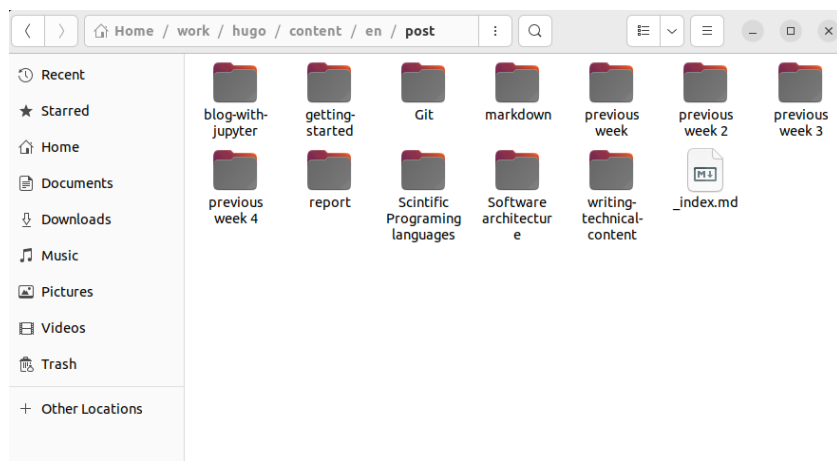


Рис. 4.34: Директория Software architecture

Открыл и изменил файл index.md (рис. 4.35).

```
Open ▾ ↵ Index.md
-work/jugo/content/en/post/Software architecture
Save - - - - -

1: ---
2: title: Software architecture
3: date: 2023-25-05
4: math: true
5: Image:
6: placement: 2
7: caption: 'Image credit: [**John Moeses Bauman**](https://unsplash.com/photos/0GZtQF8iCag)'
8: ---
9
10 ### What is software architecture?
11
12 Software architecture is the foundation of IT product development. Developers write code to create a program. Strings are combined
13 into functions, functions are combined into classes, classes are combined into modules. The result is a finished program.
14
15 To reduce the time required to create a complex product, developers at the beginning of work create a general plan for the future
16 software with the main blocks and ways to connect them. Then they detail it, working out the requirements and characteristics. This
17 is how information is obtained to build software components.
18
19 16 First, due to manageability, the architect uses individual components and modules responsible for individual functions to organize
20 software development and support.
21
22 17
23
24 18 Also, the clear architecture of the software product makes it easy to support. Even if developers are not familiar with the details
25 of the system, they will be able to quickly and easily navigate the code, making the team's work easier.
26
27 19
28 20 With the help of software architecture, you can easily modify, expand and scale the project. Quickly add new elements, increase
29 functionality and productivity, and use resources efficiently.
30
31 21
32 22 Software architecture, including monolithic architecture, allows for reuse. Components are used multiple times in different parts
33 of the system. The main benefit is to reduce the number of errors and reduce development time. It also includes safety measures to
34 ensure protection.
35
36 23
37 24 Basic principles of software architecture
38
39 25
40 26 ### Principles that are taken into account during design.
41
42 27
43 28 - Flexibility. Build not for years, but with the possibility of making changes. Consider how the program may change to meet modern
44 needs, and add flexibility to easily support changes.
45
46 29 - Openness/closedness. Software entities are closed for modification and open for expansion.
47
48 30 - Risk reduction and analysis model. It is important to use tools such as UML to capture requirements and design decisions. You
```

Рис. 4.35: Часть создаваемого файла index.md

Открыл сайт и проверил изменения (рис. 4.36, 4.37).

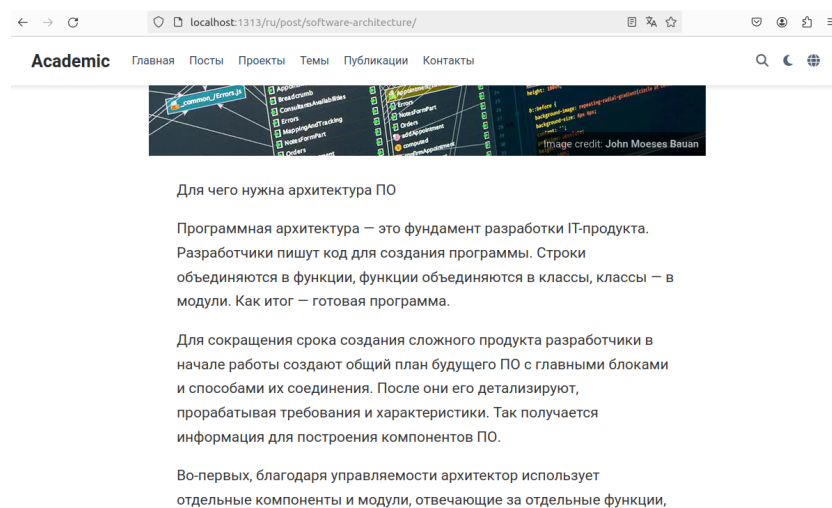


Рис. 4.36: Часть поста на русском языке

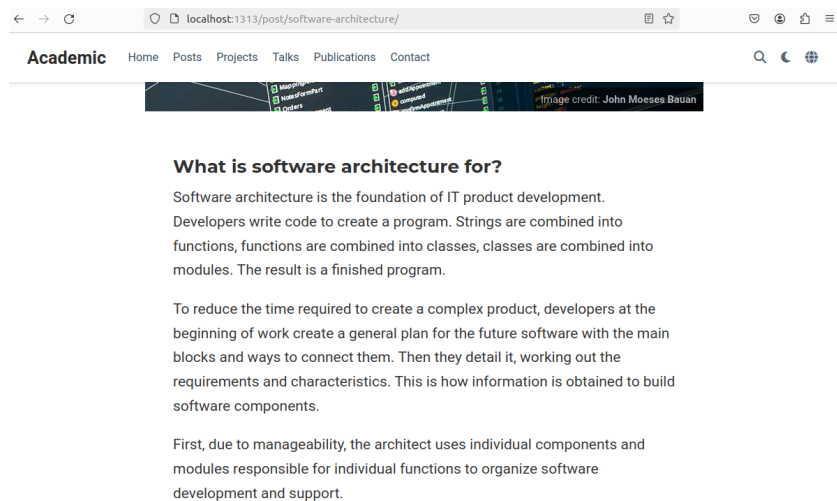


Рис. 4.37: Часть поста на английском языке

5 Выводы

Были приобретены навыки настройки языков на сайте.

Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.