

ОТЧЕТ № GB-Django-30042022

по выполнению исследовательской работы на тему:

«Кэширование в Django»

Преамбула

Django является современным фреймворком для веб-приложений на языке Python, который предлагает готовые шаблоны для проектирования. Благодаря использованию данного фреймворка повышается скорость разработки, соблюдается принцип «DRY», увеличивается качество проекта, а также появляется возможность переноса отдельных частей проекта в другие приложения.

Проект, выполненный с использованием данного фреймворка изначально может быть развернут на локальном хосте для выполнения отладки. Чтобы проект стал доступен пользователям сети интернет его необходимо разместить на публичном веб-сервере.

Для оптимизации работы приложений с целью повышения скорости отклика используется механизм кэширования, который позволяет сохранять повторно запрашиваемый контент и использовать его для последующих запросов.

Цель работы

Провести исследовательскую работу по оценке влияния механизмов кэширования на производительность существующего проекта сайта «Geekshop»

Задачи

- 1) Сравнить производительность до и после добавления декоратора «@cached_property»;
- 2) Сравнить производительность до и после добавления тега «with»;
- 3) Сравнить производительность до и после использования низкоуровневого кэширования «low_cache»;
- 4) Сравнить производительность до и после применения кэширования шаблона;
- 5) Сравнить производительность до и после добавления кэширования контроллера.

Выводы

- 1) Инструменты кэширования помогают снизить нагрузку, тем самым повышается скорость работы с приложением.
- 2) Результаты проведенной оптимизации приведены в таблице 1. Желтым отмечено незначительное изменение в результатах, зеленым – существенное.
- 3) В разрабатываемых проектах следует использовать механизмы кэширования.

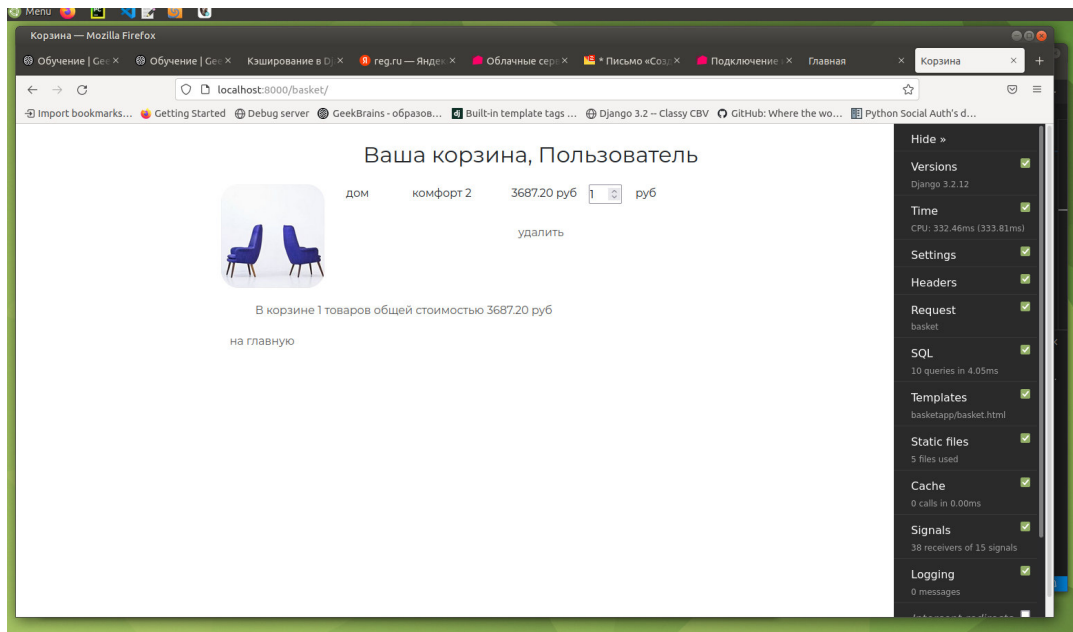
Таблица 1

№	Применяемый метод	До оптимизации		После оптимизации	
		Время, мсек	Кол-во запросов и время (мсек)	Время, мсек	Кол-во запросов / время (мсек)
1	Декоратор «@cached_property»	332.46	10 / 4.05	215.40	10 / 3.51
	Тег «with»	940.46	7 / 3.54	683.23	7 / 3.79
	Низкоуровневое кэширование «low cache»	371.64	10 / 4.14	316.01	6 / 2.83
	Кэширование шаблона	395.99	9 / 4.74	489.13	7 / 5.8
	Кэширование контроллера	181.65	7 / 3.47	32.61	0 / 0

Содержание

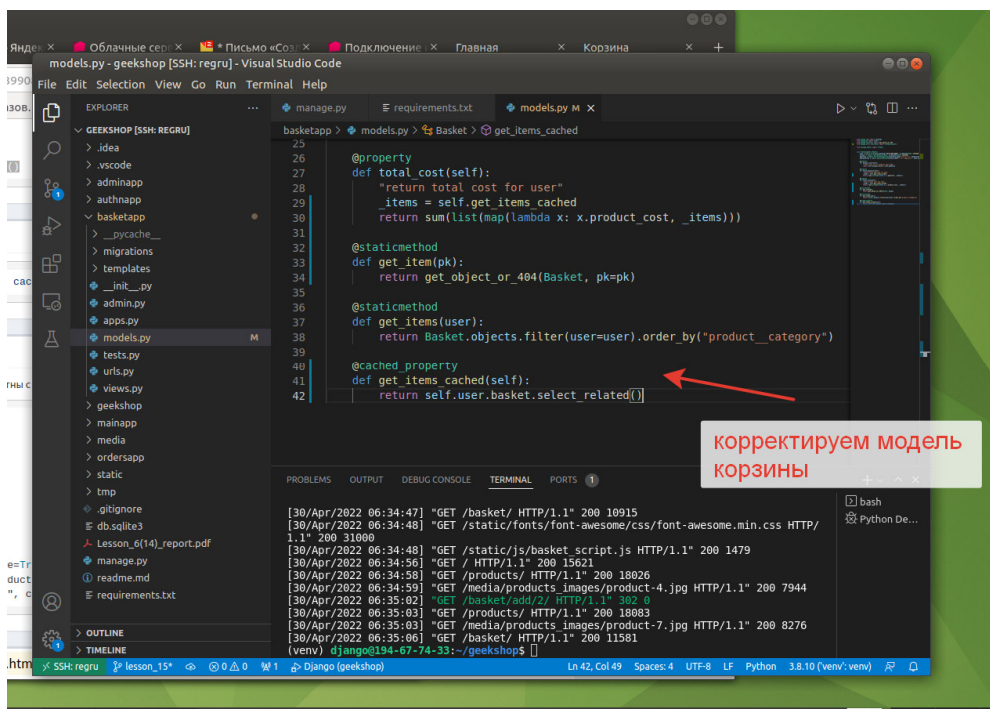
- 1) Сравнение производительности до и после добавления декоратора «@cached_property»

Работу декоратора «@cached_property» оценим на примере работы корзины. Первоначально проводим оценку работы до внесения изменений. Получаем следующие значения:

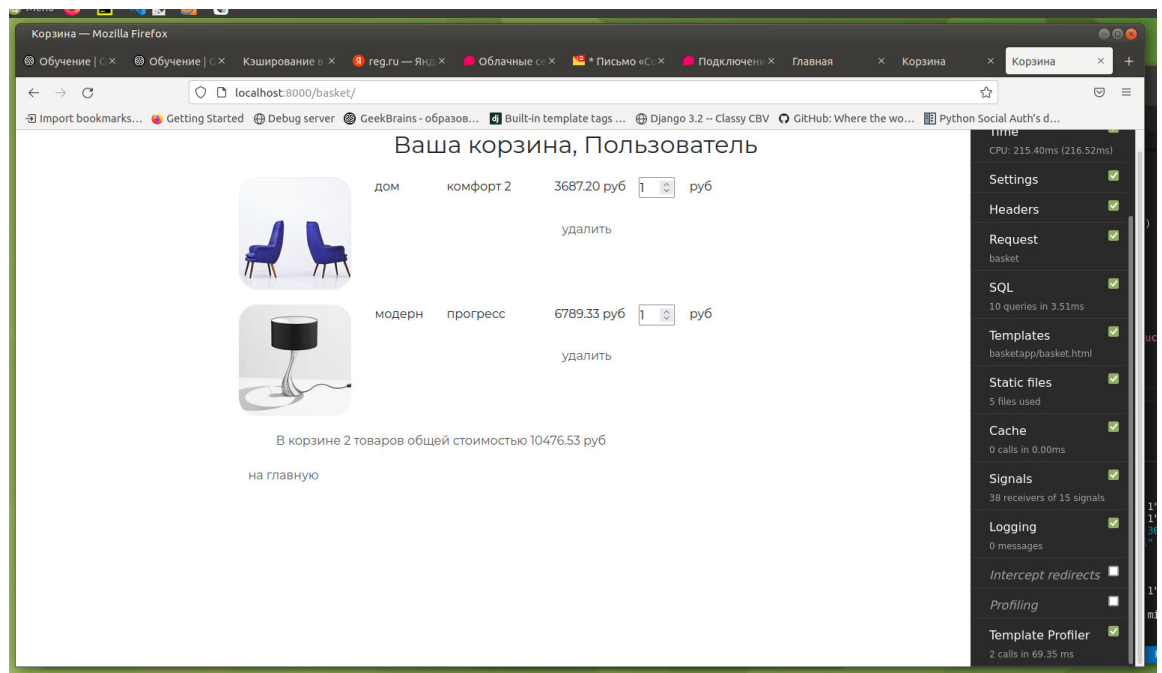


До оптимизации время загрузки составило 332.46 мсек, выполнено 10 запросов за 4.05 мсек.

Корректируем модель корзины путем добавления декоратора «@cached_property»:



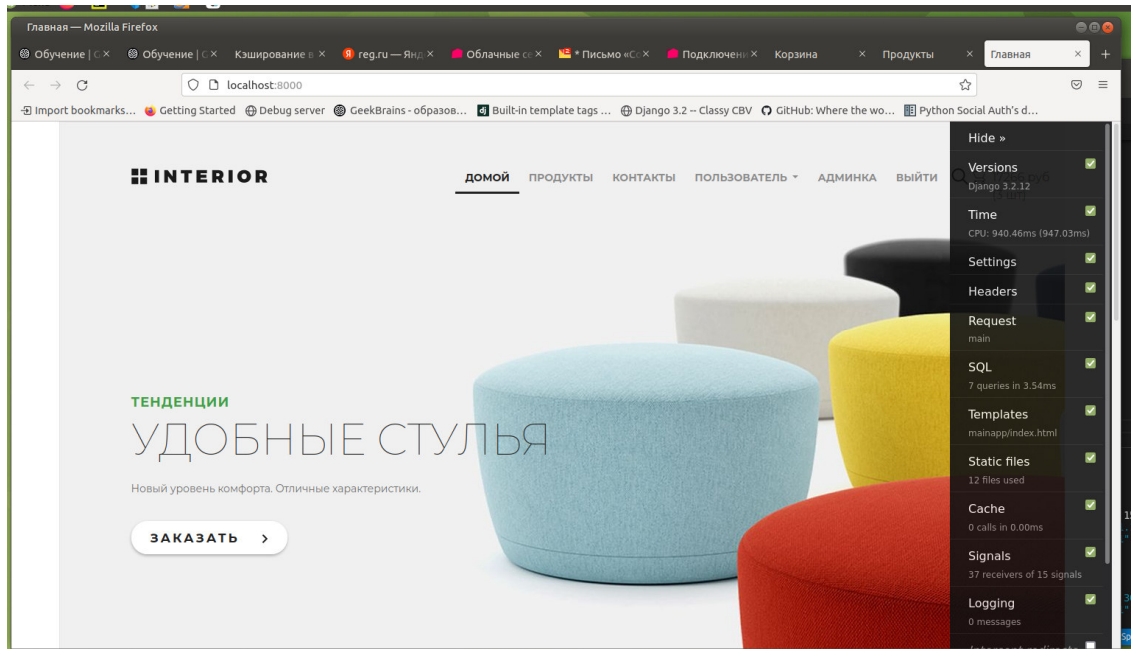
Проверяем после внесения изменений:



После оптимизации время загрузки составило 215.40 мсек, выполнено 10 запросов за 3.51 мсек.

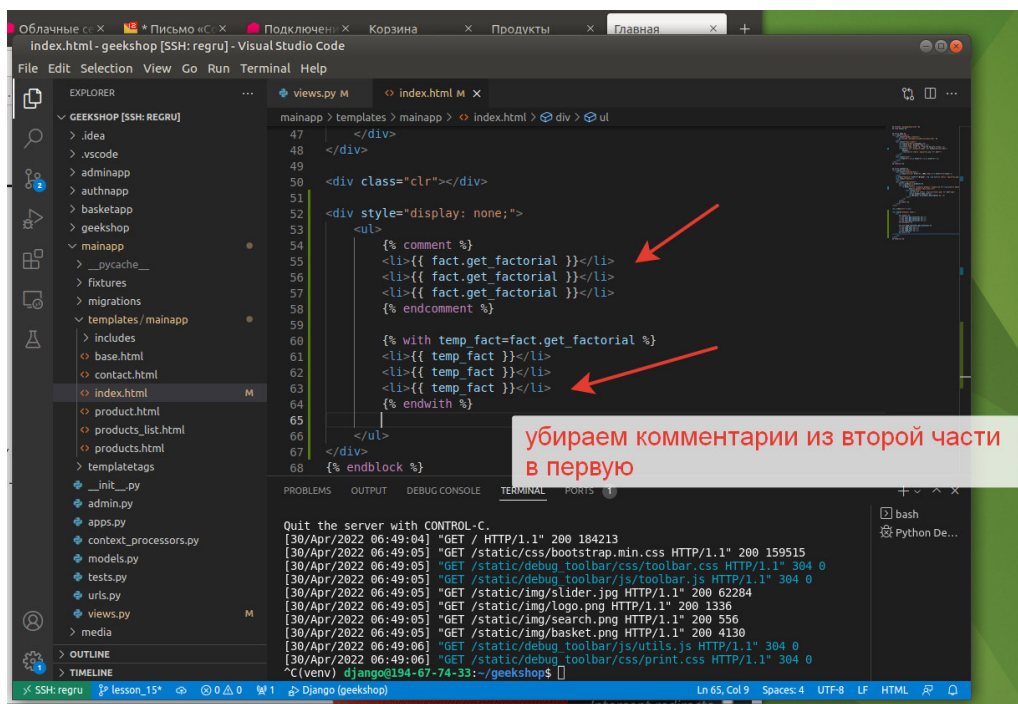
2) Сравнение производительности до и после добавления тега «with»

Работу декоратора тега «with» оценим на примере загрузки главной страницы сайта. Используем для оценки функцию факториала (изменения вносятся в `mainapp/views.py`). Первоначально проводим оценку работы до внесения изменений. Получаем следующие значения:

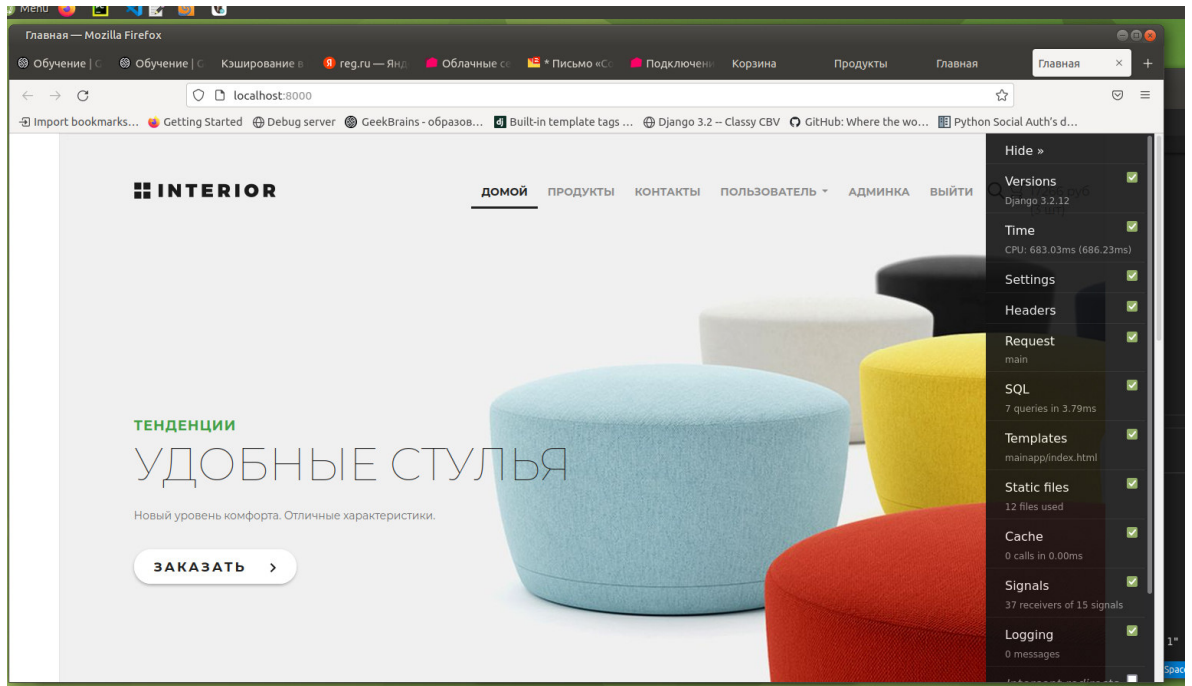


До оптимизации время загрузки составило 940.46 мсек, выполнено 7 запросов за 3.54 мсек.

Корректируем `mainapp/views.py` (изменяем комментируемое место):



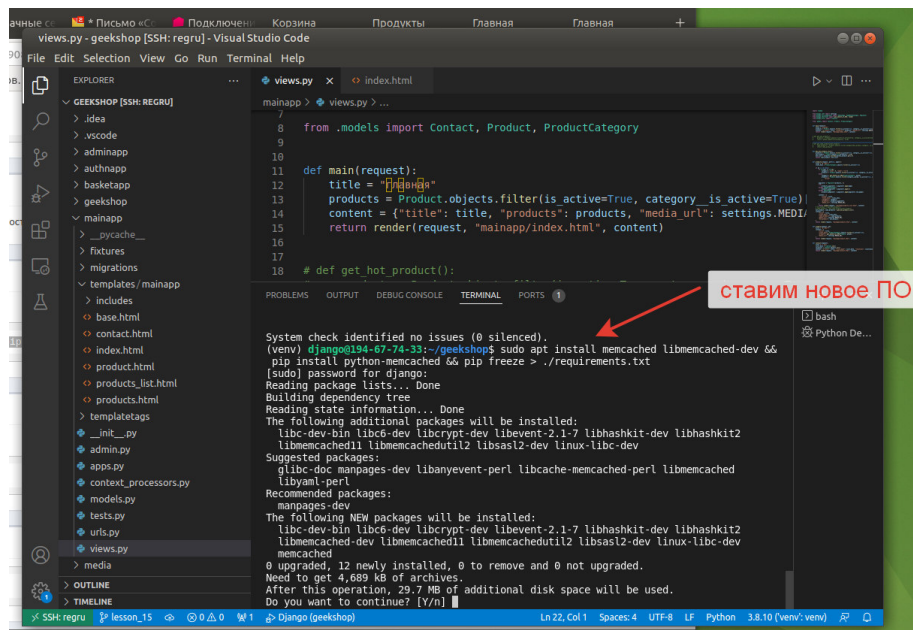
Проверяем после внесения изменений:



После оптимизации время загрузки составило 683.23 мсек, выполнено 7 запросов за 3.79 мсек.

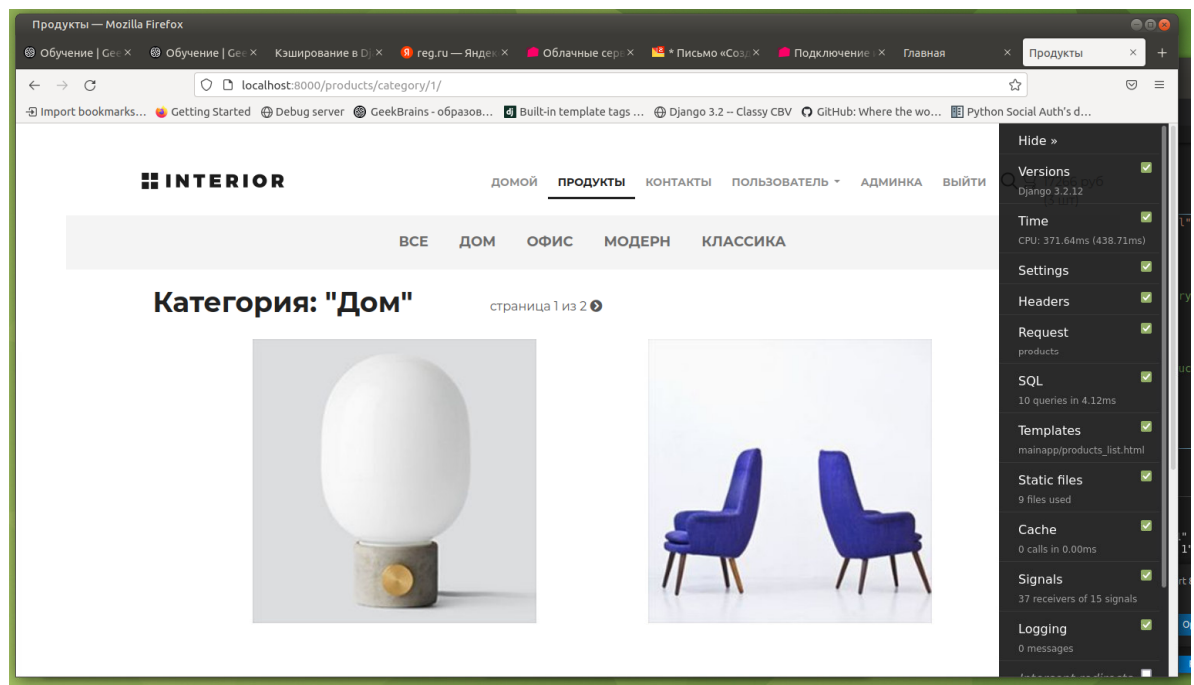
3) Сравнение производительности до и после использования низкоуровневого кэширования «low_cache»

Устанавливаем новое ПО (memcached):



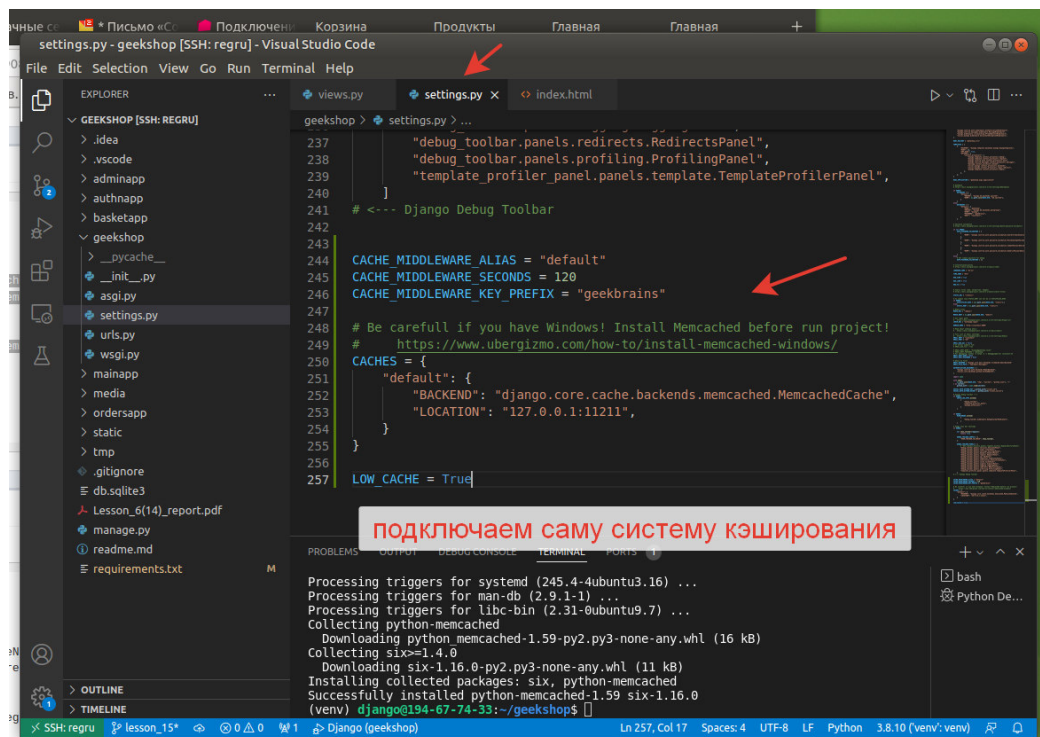
```
System check identified no issues (0 silenced).
(venv) django@194-67-74-33:~/geekshop$ sudo apt install memcached libmemcached-dev &&
pip install python-memcached && pip freeze > ./requirements.txt
[sudo] password for django:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
libc-dev-bin libc6-dev libcrypt-dev libevent-2.1-7 libhashkit-dev libhashkit2
libmemcached11 libmemcachedutil2 libsasl2-dev linux-libc-dev
Suggested packages:
glibc-doc manpages-dev libanyevent-perl libcache-memcached-perl libmemcached
libyaml-perl
Recommended packages:
manpages-dev
The following NEW packages will be installed:
libc-dev-bin libc6-dev libcrypt-dev libevent-2.1-7 libhashkit-dev libhashkit2
libmemcached-dev libmemcached11 libmemcachedutil2 libsasl2-dev linux-libc-dev
memcached
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,689 kB of archives.
After this operation, 29.7 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Работу механизма низкоуровневого кэширования «low_cache» оценим на примере загрузки страницы продуктов. Первоначально проводим оценку работы до внесения изменений. Получаем следующие значения:



До оптимизации время загрузки составило 371.64 мсек, выполнено 10 запросов за 4.12 мсек.

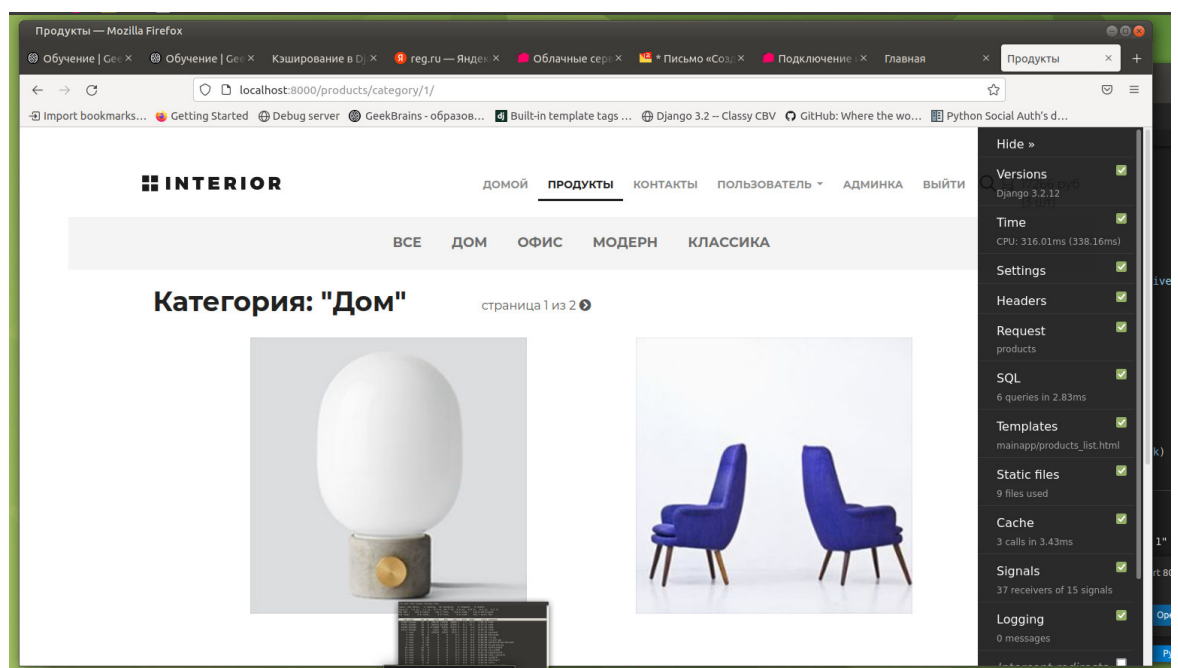
Корректируем `geekshop/settings.py` (добавляем новый фрагмент в конец кода):



```
237     "debug_toolbar.panels.redirects.RedirectsPanel",
238     "debug_toolbar.panels.profiling.ProfilingPanel",
239     "template_profiler_panel.panels.template.TemplateProfilerPanel",
240 ]
241 # <--- Django Debug Toolbar
242
243
244 CACHE_MIDDLEWARE_ALIAS = "default"
245 CACHE_MIDDLEWARE_SECONDS = 120
246 CACHE_MIDDLEWARE_KEY_PREFIX = "geekbrains"
247
248 # Be carefull if you have Windows! Install Memcached before run project!
249 # https://www.ubergizmo.com/how-to/install-memcached-windows/
250 CACHES = {
251     "default": {
252         "BACKEND": "django.core.cache.backends.memcached.MemcachedCache",
253         "LOCATION": "127.0.0.1:11211",
254     }
255 }
256
257 LOW_CACHE = True
```

подключаем саму систему кэширования

Проверяем результаты после внесения изменений:



После оптимизации время загрузки составило 316.01 мсек, выполнено 6 запросов за 2.83 мсек.

4) Сравнение производительности до и после применения кэширования шаблона

Работу механизма кэширования шаблона оценим на примере загрузки страницы заказов. Первоначально проводим оценку работы до внесения изменений. Получаем следующие значения:

Характеристика	Значение
Описание заказа	Заказ №4 от 2022-04-25 18:17:44
Заказчик	Пользователь
Время обновления заказа	2022-04-25 18:18:02
Текущий статус	Формируется
Общее количество товаров	1
Общая стоимость	4147.51 руб.

Продукт	Количество	Цена	Delete
венеция (классика)	1	4147.51 руб	удалить
-----	0		удалить

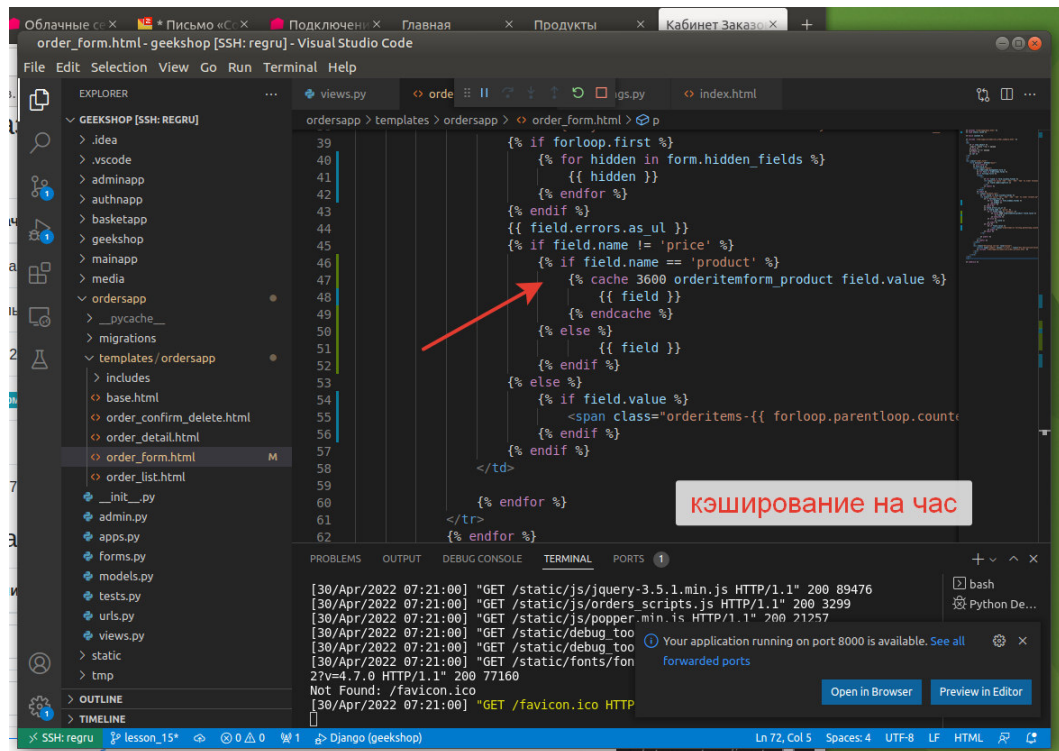
До оптимизации время загрузки составило 395.99 мсек, выполнено 9 запросов за 4.74 мсек.

Обращаем внимание на наличие дубликатов среди SQL-запросов (отрицательно сказывается на работе страницы):

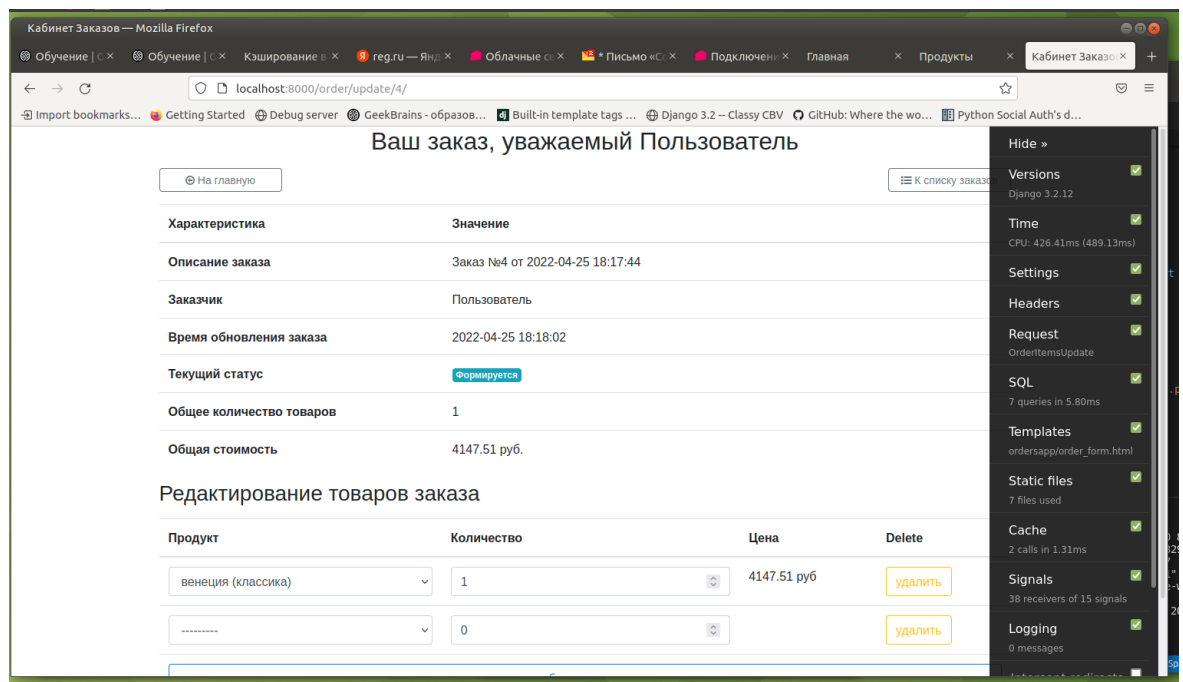
SQL queries from 1 connection

Query	Time
SELECT ... FROM "ordersapp_orderitem" INNER JOIN "ordersapp_order" ON ("ordersapp_orderitem"."order_id" = "ordersapp_order"."id") INNER JOIN "authnapp_shopuser" ON ("ordersapp_orderitem"."user_id" = "authnapp_shopuser"."id") INNER JOIN "mainapp_product" ON ("ordersapp_orderitem"."product_id" = "mainapp_product"."id") INNER JOIN "mainapp_productcategory" ON ("mainapp_product"."category_id" = "mainapp_productcategory"."id") WHERE ("ordersapp_orderitem"."order_id" = 4) AND "ordersapp_orderitem"."order_id" = 4) ORDER BY "ordersapp_orderitem"."id" ASC	0.37
SELECT ... FROM "django_session" WHERE ("django_session"."expires_date" > "2022-04-30 07:10:36.973697" AND "django_session"."session_key" = "mmad8t4kqydy5jwfnbyfx4ppm1n7") LIMIT 21	0.49
SELECT ... FROM "authnapp_shopuser" WHERE "authnapp_shopuser"."id" = 1 LIMIT 21	0.36
SELECT ... FROM "social_auth_usersocialauth" WHERE "social_auth_usersocialauth"."user_id" = 1	0.60
SELECT ... FROM "ordersapp_orderitem" INNER JOIN "ordersapp_order" ON ("ordersapp_orderitem"."order_id" = "ordersapp_order"."id") INNER JOIN "authnapp_shopuser" ON ("ordersapp_orderitem"."user_id" = "authnapp_shopuser"."id") INNER JOIN "mainapp_product" ON ("ordersapp_orderitem"."product_id" = "mainapp_product"."id") INNER JOIN "mainapp_productcategory" ON ("mainapp_product"."category_id" = "mainapp_productcategory"."id") WHERE "ordersapp_orderitem"."order_id" = 4	0.40
2 similar queries. Duplicated 2 times.	
SELECT ... FROM "ordersapp_orderitem" INNER JOIN "ordersapp_order" ON ("ordersapp_orderitem"."order_id" = "ordersapp_order"."id") INNER JOIN "authnapp_shopuser" ON ("ordersapp_orderitem"."user_id" = "authnapp_shopuser"."id") INNER JOIN "mainapp_product" ON ("ordersapp_orderitem"."product_id" = "mainapp_product"."id") INNER JOIN "mainapp_productcategory" ON ("mainapp_product"."category_id" = "mainapp_productcategory"."id") WHERE "ordersapp_orderitem"."order_id" = 4	0.39
2 similar queries. Duplicated 2 times.	
SELECT ... FROM "mainapp_product" INNER JOIN "mainapp_productcategory" ON ("mainapp_product"."category_id" = "mainapp_productcategory"."id") WHERE "mainapp_product"."is_active"	0.20
2 similar queries. Duplicated 2 times.	
SELECT ... FROM "mainapp_product" INNER JOIN "mainapp_productcategory" ON ("mainapp_product"."category_id" = "mainapp_productcategory"."id") WHERE "mainapp_product"."is_active"	0.20
2 similar queries. Duplicated 2 times.	

Проводим изменения в шаблоне ordersapp/templates/ordersapp/order_form.html:



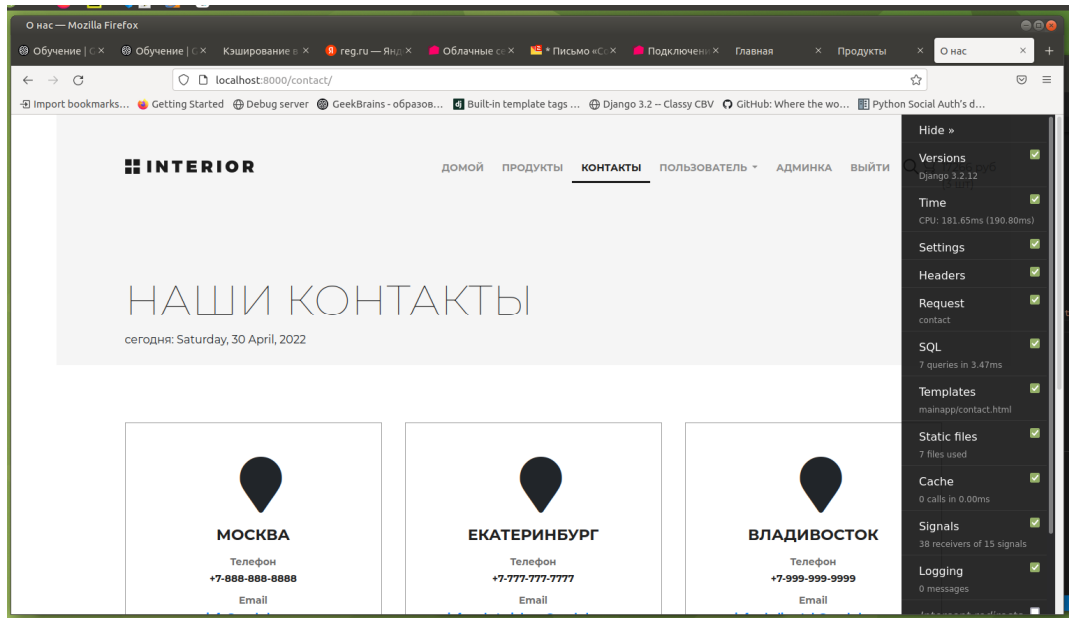
Проверяем результаты после внесения изменений:



После оптимизации время загрузки составило 489.13 мсек, выполнено 7 запросов за 5.8 мсек.

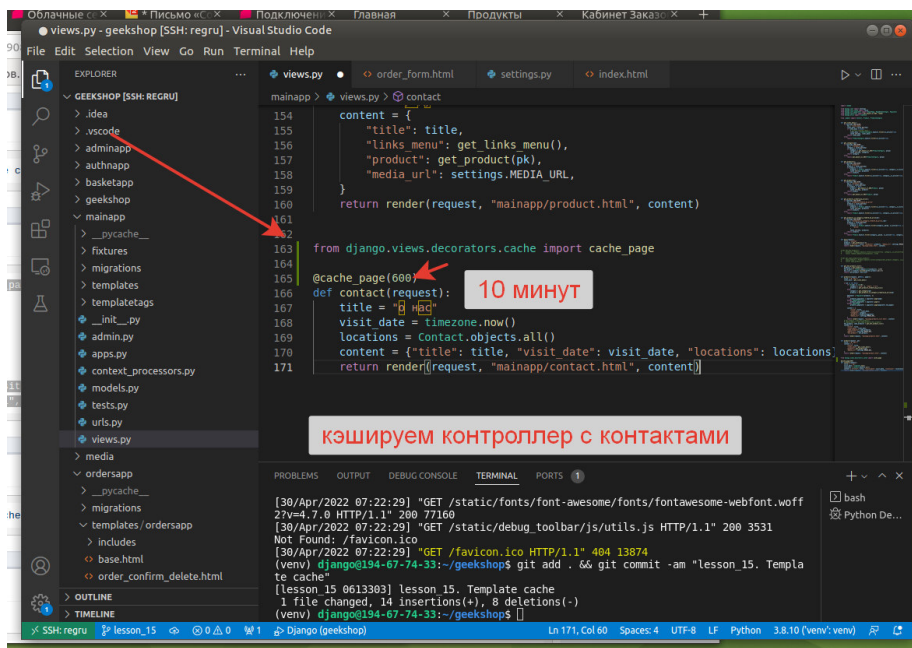
5) Сравнение производительности до и после добавления кэширования контроллера

Работу механизма кэширования шаблона оценим на примере загрузки страницы контактов. Первоначально проводим оценку работы до внесения изменений. Получаем следующие значения:

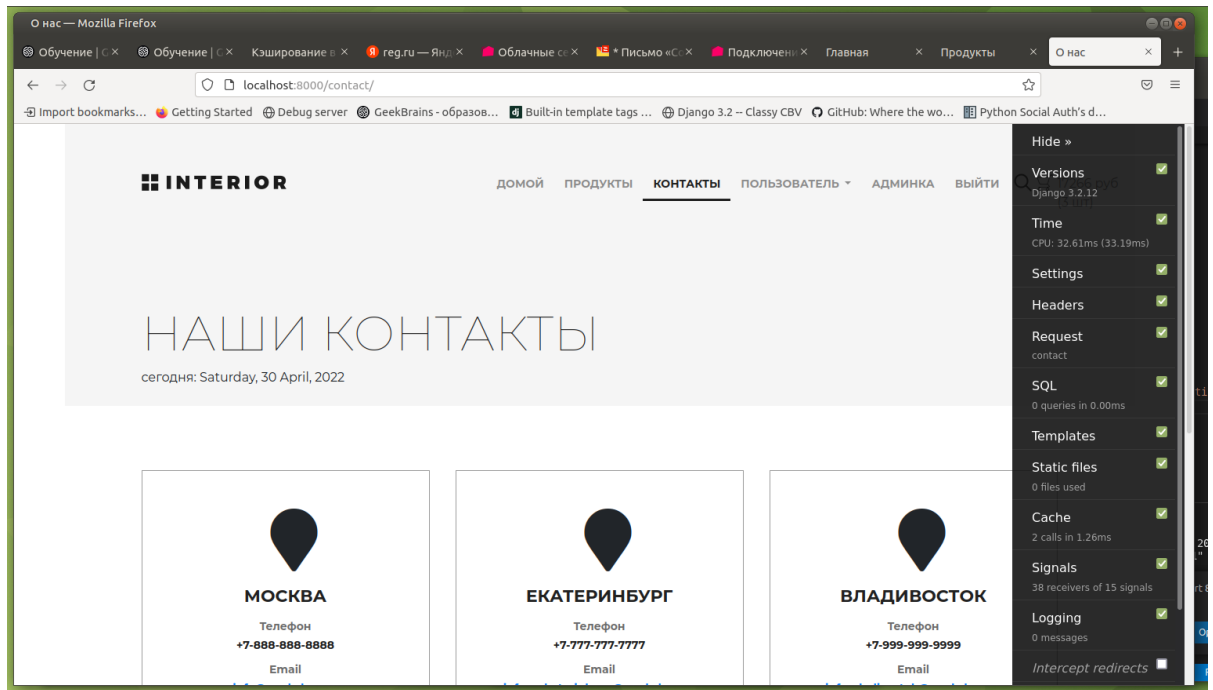


До оптимизации время загрузки составило 181.65 мсек, выполнено 7 запросов за 3.47 мсек.

Проводим изменения в файле `mainapp/views.py`, кэшируем контроллер с контактами:



Проверяем результаты после внесения изменений:



После оптимизации время загрузки составило 32.61 мсек, выполнено 0 запросов за 0 мсек.