Petar Maric

jobs@petarmaric.com

Summary

An avid Open Source Software supporter with a PhD in Applied Computer Science. Particularly fond of Python and Django powered web development. Specializes in rejuvenating brownfield projects, by minimizing the technical debt, introducing new (automated) development/deployment processes, increasing overall code quality, seeding the quality-first development culture, and leading the development team by example. Interested in research or development with a multi-disciplinary team. Passionate about working on state-of-the-art technologies and making cool stuff. And with me, every day is load test day.

Selected Skills

Python, Django, web development, software design, managing/reducing technical debt, metaprogramming, distributed computing, *nix system administration, load testing, proof-driven performance optimization, proof-driven debugging, web spiders, parsers, scientific computing, numerical analysis, information visualization, LaTeX, Sphinx, Docker, Packer, Vagrant, Celery, FastAPI, HDF5.

Career Triangle

Independent Software Consultant

Software Developer, March 2009 – Present

Seasoned Python developer with a strong passion for web, scientific and distributed computing.

Faculty of Technical Sciences, University of Novi Sad

University Professor, December 2016 – February 2022
Teaching Assistant, October 2009 – December 2016

Responsible for teaching Programming languages and data structures, Computer architecture, Compilers, Distributed computing to thousands of future engineers, their evaluation and grading.

Open Source Software

Software Developer, August 2005 – Present

Contributing Member of the Python Software Foundation, with 40+ Open Source projects created, innumerable Open Source projects contributed to, and 28 Python packages published on the Python Package Index.
https://github.com/petarmaric and https://pypi.org/user/petar/

Created and managed the official country level mirrors for CTAN (Comprehensive TeX Archive Network), Ubuntu, Arch Linux, MariaDB, CentOS, Debian, MX Linux.
Very early (since 2005) adopter and contributor to Django.

Education

Faculty of Technical Sciences, University of Novi Sad

PhD in Electrical and Computer Engineering – Applied Computer Science, 2009 – 2016
Thesis: *A hybrid software architecture for supporting the harmonic coupled finite strip method*
Research areas: finite strip method, distributed computing, numerical analysis, metaprogramming
Research projects:

[2011 – 2022] “Computational Mechanics in Structural Engineering”, ON174027, Ministry for Science and Technology, Republic of Serbia.

Faculty of Technical Sciences, University of Novi Sad

Master of Electrical and Computer Engineering – Applied Computer Science, 2003 – 2009
Thesis: *Django based framework for online opinion research systems*

Commercial Experience

Stackry LLC

Software Architect, October 2024 – Present

While still in the process of bootstrapping and becoming familiar with the codebase, proactively began work to identify the scope of technical debt, security problems, performance/concurrency issues, dead/unused/unusual code, insufficient test code coverage, overall code quality issues, undocumented manual procedures, and other issues of concern present – while also analyzing their future risk/impact. This effort has quickly resulted in a detailed report consisting of actionable issues, most of which have been of sufficient quality/detail as to be literally copy&pasted into our product issue board - and from there on have them planned to be resolved in the coming months.

Working on managing/reducing technical debt, designing – gradually implementing and then enforcing modern development processes (automated tests, CI, 12-factor app) and Minimal Reproducible Example (MRE) based regression-handling workflows (identify, isolate, model, test before, fix, test after), seeding the quality-first and test-driven development culture, increasing test code coverage, documenting and automating development/deployment procedures, stabilizing and modernizing our core (Java based) product that hasn't been properly updated for the past 9 years, reducing the number of hours needed to fully onboard a new hire, ensuring personnel replaceability ("bus factor" > 1). Began documenting all of our existing business processes - with plans to simplify and reengineer them into more structured and robust workflows, employing new-to-the-team tools and techniques such as Finite State Machines (FSM) to better describe (behavior within) our systems.

https://www.stackry.com/

360dialog GmbH

Senior Software Engineer/Lead Product Site Reliability Engineer, July 2023 – August 2024

Have been promoted from get.chat, by the (co)owners of get.chat (the spin-off company) and 360dialog (the parent company), into my new role(s) at 360dialog.

Bootstrapped the company-wide Performance Assurance Initiative and introduced Open Source culture into the company, personally creating the first Python Open Source project of the company (please see the section titled "locust_csv2openmetrics" below for details). Each of our products has been on track to get its own load tests (sub)project, initially implemented by myself. At the same time have been mentoring fellow developers, designing a simple (yet extensive) load test (sub)project template / framework and creating extensive documentation (in form of the "Tao of load testing" document) that codifies my load testing expertise – in such manner that follow-up load tests (for other products) are meant to be (eventually) handled by our other developers with me slowly shifting away from creating load tests directly and more onto coordinating, training even more developers and establishing/updating our load testing standards. Have developed load test (sub)projects as fully automated suites of realistic / stochastic load tests; where one can choose to run a single load test, a group of load tests or all load tests within the suite - be it locally on developer machines or via CI. *Although I strongly advise against running CPU core intensive, distributed and heavily optimized load tests (and other such workloads) on your own developer laptop for a prolonged period of time as to avoid explosions (feel free to ask me for details during our interview).* Have developed a reusable/parametric CI template for standardizing and defining our load test (sub)projects specific CI pipelines. A CI cronjob has been setup to run all load tests (via concurrent CI pipelines) for each combination of our products and their respective runtime environments (i.e. staging, production) every day in early morning, as to have consistent operating conditions (due to lowest levels of background traffic) and for the load test results to be ready for review before most of the developers come to work. Have designed our load test (sub)projects to push out their load test metrics into Grafana (data source), while also creating a variety of product/test-scenario focused Grafana dashboards for analyzing our load test metrics (over time) for easier proof-driven performance optimization and to avoid unexpected performance regressions. Two notable types of

regularly scheduled load test scenarios are common to all of our products: simulating its current production traffic (for Performance Assurance) and simulating its 10x production traffic (to enable (10x) growth, without fear of breakage).
Thanks to the Performance Assurance Initiative we were able to discover major bugs in some of our products, that would lead to complete resource exhaustion and effectively denial of service. In one such complex to isolate and debug case, the underlying root cause was found to be the improper database connection (de)allocation on certain classes of application+database errors (i.e. HTTP 404 raised by a request for an nonexistent database object) where the database connection would remain to be open / busy beyond the lifetime of the service's HTTP request/response cycle, thus eventually exhausting the maximum amount of active database connections allowed by the database server - given a sufficient (yet lowish) number of HTTP 404 requests. In other cases the Performance Assurance Initiative has helped us to properly size our resources (such as the database server, k8s pod CPU/RAM request/limit settings) to yield the most suitable performance vs. OPEX ratio.

With time certain challenges, inconsistencies and deficiencies have been observed in our practices and processes. After a careful review I've determined we required a holistic approach on a company/group level to: enable change and (10x) growth, without fear of breakage; observe, manage and reduce technical debt; identify, isolate, test for and fix/prevent bugs; incrementally improve test coverage, enforcing it (via CI) and regularly increasing its minimally acceptable levels; regularly check codebase (via CI) for known vulnerabilities (insecure code/dependencies, exposed hard-coded secrets, ...) and general quality issues; encourage and enforce modern development practices; seed a test-driven and quality-first development culture; standardize and automate our development/release/deployment workflows (via CI/CD); increase deployment velocity and frequency, while minimizing the risks involved; regularly check projects for scalability, performance and concurrency issues (via load tests); establish, measure, track and optimize service level objectives for all publicly faceable projects (and their dependent projects); conduct regular proof-driven performance vs. capacity vs. OPEX optimizations (via load tests); conduct regular security audits, focusing on penetration tests and phishing attacks against our personnel; conduct regular disaster recovery procedure tests; demolish the existing company/team silo mentality, enforcing inter-team transparency; observe, manage and reduce number of hours needed to fully onboard a new hire (from-zero-to-merge); ensure personnel replaceability ("bus factor" > 1), while requiring all manual procedures to be documented.
In my experience meaningful changes aren't generally brought down from "heavens above", in form of edicts – stating the "what" without even considering "how" we expect such changes to be introduced. **Real change comes from within.** For this to happen, I have initially integrated into selected projects / teams and helped organically grow their desire-to-change, spreading it throughout the company/group. We have selected one "existing" project and one "new" project, to better understand how subsequent projects are to be bootstrapped, as based on my previous experience the optimal approach varies if the project is "existing" or "new". I have started by contributing directly to projects incrementally - help automate their development workflow, implement code quality checks, introduce the Performance Assurance Initiative, enforce performance/quality automatically via CI, shift/mold team's engineering culture, etc. During this time I've also scouted for interested/ing existing members of the selected projects, to train them in my expertise – beginning with discussions, peer reviewing mine/their pull requests, au-pair live coding sessions and growing their understanding of the above and capabilities of doing such things by themselves. I then slowly shifted away from doing things by myself (directly) and more onto coordinating, training team members and establishing/updating our standards - but would occasionally get directly involved and/or lead the team by example, if the situation required so. This way I shifted from tactical level of fighting for quality and performance onto dual-classing between managerial/developer roles. At the same time the factual responsibility of quality and performance has also shifted, for it to belong to its project's team instead of their "internal consultant" (me). All this time have been documenting my observations, making notes of what worked and what didn't (for the followup

A/B experiments). Created numerical simulations of various what-if/end-to-end failure modes of our products, for aiding VP of Engineering in making their strategic SRE decisions.

Have been working on constructing a microservice template, to speed up initial development of our FastAPI projects, cover their entire development/testing/deployment/operations lifecycle and standardize our codebases towards commonly agreed (yet continuously experimented upon) microservice best practices. This is done with the goal of encouraging and (eventually) enforcing modern development processes company-wide, by maintaining performance, code quality and test code coverage at acceptable levels (evolvable over time).
https://www.360dialog.com/

get.chat sp z o.o.
Senior Software Engineer, July 2021 – September 2023

Entered the team as an early-hire (3rd) software developer at get.chat, and have been promoted by the (co)owners of get.chat (the spin-off company) and 360dialog (the parent company) into my new role(s) at 360dialog.

On my very first week, while in the process of bootstrapping and becoming familiar with the codebase, proactively began work to identify the scope of technical debt, security problems, dead/unused/unusual code, horizontal scaling challenges, un-Pythonic/un-Djangoic practices, and other issues of concern present in the MVP – while also analyzing their future risk/impact. This document has started out as purely personal notes but has piqued interest from management upon our week-in-review meeting and has quickly resulted in a 19 page report consisting of actionable issues, most of which have been of sufficient quality/detail as to be literally copy&pasted into our product issue board - and from there on have them resolved over the coming months.

Since then have been working on managing/reducing technical debt, designing – gradually implementing and then enforcing modern development processes (automated tests, code quality checks, dependency management, CI, 12-factor app, post-deployment smoke tests) and Minimal Reproducible Example (MRE) based regression-handling workflows (identify, isolate, model, test before, fix, test after), while also managing developer team resistance to engineering culture changes, seeding the quality-first and test-driven development culture, increasing test code coverage from 0% to 70+% (personally created 1000+ unit tests), documenting and automating development/deployment procedures, creating an extensive suite of automatic parametric/multi-dimensional realistic/stochastic CI/CD-enabled load tests, proof-driven performance optimization (via load tests), heavily optimizing existing Docker workflows (applying single responsibility principle to containers, reducing Dockerfile length and complexity, optimizing image size, number of layers, build times, CI inter job image/layer caching), integrating Sentry for exception / performance monitoring/notifications and establishing Sentry issue triage duty with weekly high-level overview defect reports (for management). Decreased time of onboarding new developers (from-zero-to-productive) from weeks to days, by making short work of hidden knowledge and snowflake procedures – striving to have them extensively documented and/or automated (both locally on developer machines and via CI).

Was the other part of the split-brain development efforts to build a high performance Python server that started off as a multiplexing HTTP proxy with Python-scripted request/response-rewriting capabilities, but was quick to evolve into a highly-programmable distributed event processing platform - a universal translator of sort (thus named “Babelfish”, as proposed by me) between multiple network communication protocols and event/message formats. During development we switched between primary and secondary roles, based on our expertise, the specific topic at hand and/or our availability. At its core the Babelfish Server operates as an event transcompiler, from a native (network) protocol encoded event/message to internal standardized/neutral Babelfish Event Exchange Format (BEEF, a backronym proposed by me) and

then back onto zero or more native protocols (which may be different from the original input protocol) and/or zero or more new BEEFs (allowing for some level of recursive processing). Babelfish Application is a custom piece of software designed to work with Babelfish Server; which can choose to listen on, modify and react to specific Babelfish internal or native (protocol-specific) events. Babelfish Applications react to such events by emitting commands to the Babelfish Server, which will then process them accordingly. Created thoroughly tested reference Python implementations of the Babelfish Server Management API client and the Babelfish Applications Software Development Kit (SDK), with extensive documentation, usage examples, end-developer tutorials and effective (unit) testing utilities. Created a fully automated system of extensive parametric/multi-dimensional realistic/stochastic load/stress/torture tests for the Babelfish Server itself as well as its example Babelfish Applications (via Babelfish SDK), running automatically as part of our CI/CD workflow on every git commit for proof-driven performance optimization and to avoid unexpected performance regressions. Created a Python powered console tool for automatic KPI analysis/visualization of said parametric/multi-dimensional load/stress/torture test reports (not limited to Babelfish report formats nor use-cases).
Pioneered the integration of Babelfish Platform into get.chat and 360dialog codebases with detailed tutorials and examples for all follow-up integrations (into our other products) – which were then handled by our other developers. get.chat and 360dialog developers have successfully used the Babelfish Server features: act as a reverse HTTP proxy with request multiplexing support, transparently retry HTTP requests on intermittent service failures/downtime, process certain HTTP requests in the background (so-called CAST mode). 360dialog developers have also built their own custom Babelfish Applications (via Babelfish SDK) that were used to: offload the large media download/upload HTTP requests from python to nginx (for more efficient processing), manipulate HTTP request/response control flow (require additional authentication, rewrite URLs, check if WhatsApp Business template messaging is allowed for a particular client, etc.), manipulate HTTP request/response headers/body (redact sensitive/secret data, add customer-specific webhook headers, etc.).
https://get.chat/

DigitalMR LTD

Senior Software Engineer/Systems Architect/Enterprise Software Architect, June 2020 – September 2021

Worked on managing/reducing technical debt, designing – gradually implementing and then enforcing modern development processes (peer review via pull requests, automated tests, CI, CD) while also managing business/developer team resistance to engineering culture changes, seeding the quality-first and test-driven development culture, increasing test code coverage from 0% to 75+% (personally created 2500+ unit tests), documenting and automating development/deployment procedures, upgrading a legacy Django project from 1.11 to 3.2 (in stages), creating automated load tests, designing a new cloud-native software architecture, setting up and helping manage a new geo-distributed remote development team. All of this was done whilst implementing new features in the listening247 platform for a decades-proven global leader in the market research space.

Entered the team as a software developer and have quickly been promoted, all the way to a Enterprise Software Architect – but was still dual-classing between architect/developer roles, as I'm always leading the development team by example.
https://www.digital-mr.com/

CitiDexLI, Inc.

Senior Software Engineer, January 2010 – December 2016

Worked as a consultant with Prof. Michael Dohan and his team to improve their popular information directory business, started in 1997. As the only developer been adding new functionality and maintaining existing codebase, whilst taking on administrative roles and keeping the client's cost at minimum. Came to an environment using several different systems spread over multiple codebase and developer generations, with some data duplicated across systems causing occasional inconsistencies. Worked to unify the entire codebase into a centrally managed Django based system.

Introduced resource monitoring with improved alerts/notifications, created reliable backup policies, continue to maintain office, legacy and production servers while systematically improving their security. Planned and worked a new, modern Cloud based infrastructure, aiming to cut the OPEX in half while improving performance and resolving scalability issues.
http://www.citidex.com/

Found and reported a Horde/IMP related remote code injection vulnerability in Plesk 8.6 (not credited)
http://kb.parallels.com/en/113374

High Score Society
Software Developer, August 2009 – October 2009
Worked as a consultant to improve the Django powered gamer community and prepare it for launch. Updated project code to Django 1.1, fixed bugs, refactored and cleaned up code, implemented new features, integrated the new design, improved scalability by moving CPU and network intensive operations out from the HTTP request/response lifecycle.
http://www.highscoresociety.com/

mesawarati
Software Developer, March 2009 – April 2009
Worked as a consultant to implement a Django based photo gallery website that supports content in multiple languages. Fixed bugs, implemented i18n-ized content, refactored code, contributed patches to related Open Source projects.
http://code.google.com/p/mesawarati/

Open Source Experience

locust_csv2openmetrics
Project Founder, September 2023 – Present
Created a Python powered library/console tool for converting locust's load test CSV formatted reports into the OpenMetrics text format.
https://gitlab.com/360dialog/open-source/locust_csv2openmetrics

get.chat WhatsApp Business Team Inbox frontend
Software Developer, July 2021 – September 2023
Used GitLab CI/CD to automate the project's development/deployment procedures (automated tests, code quality checks, build static file assets, publish package and create GitLab release on git tag events), while also optimizing the CI/CD pipeline total execution time.
https://gitlab.com/get.chat/web-app

pjisp-assignment-template
Project Founder, October 2019 – Present
Created an automated workflow for creating assignments for our ACS students, their deployment and examination.
https://github.com/petarmaric/pjisp-assignment-template

acs_examine_student_assignment
Project Founder, October 2019 – Present
Created a Python powered library/console tool for automated assignment examination of ACS students.
https://github.com/petarmaric/acs_examine_student_assignment

acs_extract_student_assignments
Project Founder, October 2019 – Present
Created a Python powered library/console tool for extracting assignments from exam archives of our ACS students.
https://github.com/petarmaric/acs_extract_student_assignments

version4plos
Project Founder, August 2019 – Present
Created a Python powered library/console tool for automated tracking of PLOS LaTeX template versions.
https://github.com/petarmaric/version4plos

template4plos
Project Founder, July 2019 – Present
Created a modified version of the PLOS LaTeX template, preconfigured for automated preparation of your LaTeX paper for submission in PLOS journals.
https://github.com/petarmaric/template4plos

latex2plos
Project Founder, July 2019 – Present
Created a Python powered library/console tool for automated preparation of your LaTeX paper for submission in PLOS journals.
https://github.com/petarmaric/latex2plos

Official MX Linux mirror
Project Founder, December 2018 – February 2022
Created and managing the official MX Linux mirror for Serbia, sponsored by the Faculty of Technical Sciences, University of Novi Sad.
http://mxlinux.petarmaric.com/

smoke_test
Project Founder, November 2018 – Present
Created a Python powered library/console tool for automated smoke testing, preconfigured to make coursework easier for our ACS students.
https://github.com/petarmaric/smoke_test

Official Debian mirror
Project Founder, November 2018 – February 2022
Created and managing the official Debian mirror for Serbia, sponsored by the Faculty of Technical Sciences, University of Novi Sad.
http://debian.petarmaric.com/

Official CentOS mirror
Project Founder, November 2018 – February 2022
Created and managing the official CentOS mirror for Serbia, sponsored by the Faculty of Technical Sciences, University of Novi Sad.
http://centos.petarmaric.com/

Official MariaDB mirror
Project Founder, August 2018 – February 2022
Created and managing the official MariaDB mirror for Serbia, sponsored by the Faculty of Technical Sciences, University of Novi Sad.
http://mariadb.petarmaric.com/

fsm_strip_thickness_vibration_analysis
Project Founder, April 2018 – Present
Created a Python powered library/console tool for strip thickness-dependent vibration analysis and visualization of the parametric model of buckling and free vibration in prismatic shell structures, as computed by the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_strip_thickness_vibration_analysis

fsm_strip_length_vibration_analysis
Project Founder, April 2018 – Present
Created a Python powered library/console tool for strip length-dependent vibration analysis and visualization of the parametric model of buckling and free vibration in prismatic shell structures, as computed by the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_strip_length_vibration_analysis

fsm_strip_thickness_damage_analysis
Project Founder, April 2018 – Present
Created a Python powered library/console tool for strip thickness-dependent damage analysis and visualization of the parametric model of buckling and free vibration in prismatic shell structures, as computed by the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_strip_thickness_damage_analysis

fsm_strip_length_damage_analysis
Project Founder, April 2018 – Present
Created a Python powered library/console tool for strip length-dependent damage analysis and visualization of the parametric model of buckling and free vibration in prismatic shell structures, as computed by the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_strip_length_damage_analysis

fsm_effective_stress
Project Founder, April 2018 – Present
Created a Python powered library that uses the rheological-dynamical analogy (RDA) to compute damage and effective buckling stress in prismatic shell structures.
https://github.com/petarmaric/fsm_effective_stress

fsm_damage_analysis
Project Founder, March 2018 – Present
Created a Python powered library/console tool for damage analysis and visualization of the parametric model of buckling and free vibration in prismatic shell structures, as computed by the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_damage_analysis

fsm_strip_thickness_analysis
Project Founder, March 2018 – Present
Created a Python powered library/console tool for strip thickness-dependent visualization and modal analysis of the parametric model of buckling and free vibration in prismatic shell structures, as computed by the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_strip_thickness_analysis

fsm_eigenvalue_experiments
Project Founder, February 2018 – Present
Created a Python powered framework for computing and analyzing experiments, based on the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_eigenvalue_experiments

dynamic_pytables_where_condition
Project Founder, February 2018 – Present
Created a Python powered library that dynamically constructs a PyTables where condition from the supplied keyword arguments.
https://github.com/petarmaric/dynamic_pytables_where_condition

fsm_load_modal_composites
Project Founder, February 2018 – Present
Created a Python powered library that loads modal composites from the file containing the parametric model of buckling and free vibration in prismatic shell structures, as computed by the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_load_modal_composites

fsm_strip_length_analysis
Project Founder, February 2018 – Present
Created a Python powered library/console tool for strip length-dependent visualization and modal analysis of the parametric model of buckling and free vibration in prismatic shell structures, as computed by the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_strip_length_analysis

acs_student_mail_harvester
Project Founder, January 2018 – Present
Created a Python powered library/console tool for harvesting email addresses of our ACS students during their first week of ACS lab coursework.
https://github.com/petarmaric/acs_student_mail_harvester

fsm_modal_analysis
Project Founder, January 2018 – Present
Created a Python powered library/console tool for visualization and modal analysis of the parametric model of buckling and free vibration in prismatic shell structures, as computed by the fsm_eigenvalue project.
https://github.com/petarmaric/fsm_modal_analysis

Official Arch Linux mirror
Project Founder, December 2017 – February 2022
Created and managing the official Arch Linux mirror for Serbia, sponsored by the Faculty of Technical Sciences, University of Novi Sad.
http://arch.petarmaric.com/

physical_dualism
Project Founder, December 2017 – Present
Created a Python powered library that approximates the natural frequency from stress via physical dualism, and vice versa.
https://github.com/petarmaric/physical_dualism

fsm_eigenvalue
Project Founder, September 2017 – Present
Created a Python powered library/console tool implementing a generalization of eigenvalue problem within the harmonic coupled finite strip method, used for parametric modeling of static and dynamic inelastic buckling, free vibration, damage and failure in prismatic shell structures.
https://github.com/petarmaric/fsm_eigenvalue

MIR
Project Founder, August 2017 – Present
Created the Mesh Intermediary Representation (MIR) specification, an open standard for describing a platform-neutral data format for long-term storage and interchange of mesh-like data. It has also been designed to serve as an intermediary step when converting the mesh-like data between different computational mechanics application formats.
https://github.com/petarmaric/mir

acs_student_attendance
Project Founder, December 2016 – Present
Created a Python powered library/console tool for analyzing the lab attendance of our ACS students.
https://github.com/petarmaric/acs_student_attendance

PJISP zbirka zadataka
Project Founder, December 2016 – Present
Created a self-building PJISP course problem book that makes coursework easier for our ACS students.
https://github.com/petarmaric/pjisp-zbirka-zadataka

Official Ubuntu Archive mirror
Project Founder, November 2016 – February 2022
Created and managing the official Ubuntu package archive mirror for Serbia, sponsored by the Faculty of Technical Sciences, University of Novi Sad.
http://rs.archive.ubuntu.com/ and http://ubuntu.mirror.ftn.uns.ac.rs/archive/

packer-acs-templates
Project Founder, October 2016 – Present
Created an automated Packer based system for building VirtualBox images, which have been preconfigured to make coursework easier for our ACS students.
https://github.com/petarmaric/acs_packer_templates

export_beam_integrals
Project Founder, September 2015 – Present
Created a distributed system for computing and exporting beam integrals of all 6 supported beam types, as defined in the beam_integrals project.
https://github.com/petarmaric/export_beam_integrals

Official Ubuntu CD images mirror
Project Founder, August 2015 – February 2022
Created and managing the official Ubuntu CD images mirror for Serbia, sponsored by the Faculty of Technical Sciences, University of Novi Sad.
http://rs.releases.ubuntu.com/ and http://ubuntu.mirror.ftn.uns.ac.rs/releases/

Official CTAN mirror
Project Founder, August 2015 – February 2022
Created and managing the official CTAN (Comprehensive TeX Archive Network) mirror for Serbia, sponsored by the Faculty of Technical Sciences, University of Novi Sad.
http://ctan.mirror.ftn.uns.ac.rs/

simple_plugins
Project Founder, August 2013 – Present
Created a simple to use Python powered plugin framework, inspired by the work of Marty Alchin.
https://github.com/petarmaric/simple_plugins

friendly_name_mixin
Project Founder, August 2013 – Present
Created a Mixin class for extracting friendly names from Python classes.
https://github.com/petarmaric/friendly_name_mixin

nose_extra_tools
Project Founder, August 2013 – Present
Created more testing goodies for ‘nose.tools’. Nose extends the test loading and running features of Python ‘unittest’, making it easier to write, find and run tests.
https://github.com/petarmaric/nose_extra_tools

PJISP zbirka pitanja
Project Founder, November 2012 – Present
Created a self-building PJISP course question book that makes coursework easier for our ACS students.
https://github.com/petarmaric/pjisp-zbirka-pitanja

beam_integrals
Project Founder, June 2012 – Present
Created a Python powered library/console tool for determining beam integrals of all 6 supported beam types, as described by prof. D.D. Milašinović in “The Finite Strip Method in Computational Mechanics”.
https://github.com/petarmaric/beam_integrals

telenor_web2sms
Project Founder, October 2011 – September 2012
Created a Python powered library/console tool for sending SMSs through the Telenor WEB2SMS web app.
https://github.com/petarmaric/telenor_web2sms

mdm_compare
Project Founder, February 2011 – Present
Created a Python powered library/console tool for comparing 2 experiment results stored in the MDM file format. MDM is a readable text format suitable for storing and exchanging experiment results, used for years by our university research team and most of our software.
https://github.com/petarmaric/mdm_compare

metaTED
Project Founder, May 2009 – August 2019
Created a Python powered tool for easier downloading of all TED talks. metaTED screen-scrapes TED HTML pages and creates several metalinks of all talks, varying in both the quality levels and possible talk groupings by directory.
https://github.com/petarmaric/metated

dreamy-trac
Project Founder, July 2008 – August 2011
Developed a system to automatically install and manage Trac on DreamHost shared hosting.
http://code.google.com/p/dreamy-trac/

opinion-extractor
Project Founder, March 2008 – May 2009
My master thesis project.
http://code.google.com/p/opinion-extractor/

Feedjack
Software Developer, October 2006 – July 2007
Feed aggregator Django app. Fixed bugs, implemented i18n support, contributed Serbian localization.
http://www.feedjack.org/

Django
Software Developer, August 2005 - Present
Very early adopter and contributor to Django. Fixed bugs, helped resolve security issues with Hudson continuous integration server in late October 2010, implemented and improved i18n support, improved Windows support, contributed Serbian localization, managed the Serbian localization team.
http://www.djangoproject.com/

Manager of the Django LinkedIn group since October 2009.
http://www.linkedin.com/groups?gid=50788

Created a “Django syncmedia” proposal for Google Summer of Code 2009. Even though the proposal hasn’t been accepted then, over the next few years most of the ideas have been implemented in Django. http://gsoc2009.petarmaric.com/

Languages

Native Serbian speaker.
Fluent in English.
Basic French.
Basic German.

Hobbies

Serbian and Japanese culture, photography, hiking, home repairs, cooking, (food) chemistry, espresso brewing.

Mindset

Principles of Effective Software Development

- start smart, but also fast
- start with features/use-cases and not solutions/technologies
- think/work in terms of Minimal Viable Changes, via feature-centric pull/merge requests
- try to be incremental in your changes, rather than iterative
- discuss, prototype, optimize, GOTO 1
- git main should always be stable and working, and preferably immediately deployable to production
- try to maintain a test suite with a decent level of code coverage
- unit tests matter as much as integration tests
- refactor internals as you go, making sure any part can be replaced
- acquiring technical debt is fine at the start but it needs to be paid off in time, or your future development/refactors will suffer from it
- every day is load test day
- do not assume performance/features/capabilities, always (load) test
- optimize only after measuring / load testing
- documenting development/deployment procedures is essential, but having them automated in the form of “executable documentation” (via Makefiles) is even better
- automate as much as possible, using CI/CD to your advantage
- be transparent about the entire process: feature planning, development, testing, optimizations, release plans, deployments, ...
- try to keep the public API as stable as possible, even when at 0.* versions
- when exposing your public API always think about developers, developers, developers, …

The Why and How of Proof-driven Debugging

When building regression tests try to aim for a Minimal Reproducible Example (MRE), and discarding anything (i.e. requests to API endpoints) that's not absolutely required.

Help avoid false positives and false negatives in your regression tests, or the accidental reintroduction of previously fixed regressions, by following this development/testing workflow:

- identify the regression in your codebase
- isolate the regression to a Minimal Reproducible Example (MRE)
- model your MRE into the new regression test
- run all regression tests, to verify the regression's test (no false negatives)
- implement the regression's fix
- run all regression tests, to verify the regression's fix (no false positives)