Curriculum Vitae – Daniil Gentili

Website: https://daniil.it
Email: daniil@daniil.it
GitHub: https://github.com/danog
VAT: 01647560299

Daniil Gentili, software engineer and network administrator, born on 29/08/2000, currently residing in Italy (Rovigo).

Native speaker in Italian and English (certified C1).
10+ years of professional experience.

Professional experience in: PHP, C++20, C, Golang, Rust, Bash, Javascript/Node, Lua as well as the React, Redux, Bootstrap and AMPHP frameworks; networking (AS198747), mysql/mongodb/redis/keydb, Elasticsearch, Kafka, Linux, Docker, Grafana stack (prometheus/influxdb), and much more.

From 2020, my main clients are Telegram and Nicelocal.

Some of my work with Telegram:
- In 2019-2020, I took part in several programming contests (https://contest.com), winning several prizes (1st, 2nd and 3rd places) for developing innovative and original solutions for blockchain, VoIP and data clustering/big data applications deployed at Telegram, developed in C++20.
- An automated VoIP integration testing and quality control platform for Android, requiring low-level changes to the Android Hardware Abstraction Layer driver stack (C/C++).
- Other low-level software development, using mainly C, and a bit of PHP.
- I am currently maintaining the official technical documentation for the Telegram MTProto protocol.

In Nicelocal, I have worked on the following projects:
- A high-load, efficient NFA search engine built from scratch in Go, compiling a custom regex-like language to a highly optimized NFA.
- A high-load, massively parallel, fully sharded service for statistics in Go.
- VoIP telephony applications, including a full integration testing platform for asterisk, and fixes to critical race-conditions in the C codebase of the open-source asterisk project.
- Multiple large refactorings to a 1 million SLOC PHP 5/7 codebase, migration to PHP 8, 10% => 95% type coverage.
- Various other Rust, Go, C and (async/traditional) PHP development.

In Nicelocal, I also occupy the role of code quality manager, and thanks to their generous sponsorships, I have contributed multiple large refactoring to Psalm, an advanced PHP static analysis system developed by Vimeo: https://github.com/vimeo/psalm.
Psalm is like Typescript for PHP, adding to PHP features like generics, shaped arrays and literal types, with the addition of static analysis using formal methods, to easily discover bugs before even executing the code.
Some of my work on Psalm includes:
- Full immutability for the entire codebase, fixing a large amount of issues caused by non-deterministic execution order when threading is enabled.
- List shapes, huge improvements to array shapes, including unsealed/sealed array shapes, generics improvements.
- Currently working on disjoint array shape support and a satisfies operator for improved generics, inspired by typescript.
- Creation of over 140 pull requests with other features and bug fixes.
Thanks to these contributions, I was offered a role of Psalm maintainer, which I happily accepted.

Personal/open source work:

When I was 15, I started developing an open-source client for the MTProto cryptographic protocol called MadelineProto (https://github.com/danog/MadelineProto) an open-source client for the MTProto cryptographic protocol called MadelineProto, most notably used for interacting with Telegram Messenger's low-level RPC API (https://core.telegram.org/mtproto).
MadelineProto is now approved by Telegram, and can be seen in the apps list on the official website: https://telegram.org/apps.
MadelineProto abstracts away all implementation details of the Telegram MTProto API, handling cryptography, transport protocols, the presentation layer with a dynamic parser (soon to be replaced by a static parser) for the TL Type Language and the RPC API, offering a fully asynchronous fiber-based (AMPHP v3) high-level PHP API.

I'm also a maintainer of https://github.com/davidcole1340/ext-php-rs, a library that offers Rust bindings to the C PHP API, and https://github.com/danog/php-tokio, a Rust library based on ext-php-rs that allows anyone to use any async Rust library from PHP.

I am currently developing a commercial PHP->WASM/x86/aarch64 compiler, tightly integrated with the existing C PHP Zend runtime.
Future scope for the compiler includes an eBPF backend for Linux kernel development in PHP.

Other open source work:
- Two PHP RFCs - Proposals to add/change features in the PHP language, one of which was accepted.
- Multiple improvements to the testsuite of the PHP JIT compiler, as well as a lot of QA to find and report bugs in the PHP JIT compiler.
- https://github.com/danog/madelineTon.js - Pure JS client-side implementation of the Telegram TON blockchain protocol: this library was developed during the contests, and makes use of the mathematical properties of Ed25519 and Montgomery curves to implement efficient conversion of points, to reuse the otherwise (incompatible due to non-standard TON cryptography protocols) high-performance nacl.js library for encryption.
- https://github.com/phabelio/phabel - A PHP transpiler inspired by Babel (currently not publicly maintained, but still being used internally for various projects, thanks to the very ergonomic graph-based AST transformation framework it offers internally).
- https://github.com/danog/AsyncOrm - Async ORM based on AMPHP v3 and fibers.
- I'm the main maintainer of gojekyll, a fast Go implementation of the Jekyll blogging engine.
- Many other open source libraries, listed in https://github.com/danog

I maintain my own Autonomous System, AS198747 (https://bgp.he.net/AS198747), using a redundant infrastructure with two upstreams and a custom network configuration using bird2+routinator, openwrt, wireguard and systemd-networkd.
All subnets announced by AS198747 are RPKI-validated, and any imported RPKI invalid routes are rejected.
All of my containerized infrastructure with multiple services (self-hosted CI, gitea, PHP/Go/Rust services, etc) is also hosted on AS198747 via IPv6.
The IPv4 part of my infrastructure is hosted on a commercial ISP that does not do RPKI validation, but IPv4 RPKI invalid routes are still rejected on my routers, thanks to a custom filtering setup based on bird2 and routinator.

I'm also working on various currently-private low-level projects, including a port of Linux for the Videocore IV architecture (a custom architecture used by the VPU of Raspberry PI), and a hobby OS in C.