Gabriel Schütz de Souza

Senior Software Engineer — Airport Revenue & Payment Systems

contact@gabrielschutz.de · Remote (Brazil, GMT-3, 1–2h ahead of US Eastern; full US-hours overlap) · gabrielschutz.de · linkedin.com/in/gabriel-schuetz
English (fluent) · Portuguese (native) · German (fluent, TestDaF 5/5/5/4) · French (B2)

Six years building and running **approach-fee billing, collection, and payment infrastructure** in production for a German aviation SaaS serving 350+ airfields: fee calculation from flight movements, customer matching, invoice generation, overdue collection, and end-to-end card/SEPA payment pipelines. One of three core developers; most of those years went into the airport management system, a large part of which I built module by module, and I went deepest on the money path, from a flight movement to cash in the bank. The systems where a mistake costs money or breaks the law. The billing engine I bootstrapped has **invoiced and collected €62.9M in approach fees**.

EXPERIENCE

Senior Software Engineer — Payments & Financial Systems

aerops GmbH (German aviation SaaS, 350+ airfields) · Jul 2020 – Jul 2026 · Remote contract (via own consultancy)

Approach-fee billing & collection:

- Built **AFS, the approach-fee billing service** for a national air-navigation services provider, in Node.js (v1 solo, 2021): it ingests flight movements, calculates MTOW-based **approach fees**, and issues the fee notices. It has **invoiced and collected €62.9M in approach fees** all-time, mostly bank/SEPA-settled. Peak month (July 2022): 10,856 movements and 363 fee notices.
- Ran the monthly billing rounds by hand early on, then automated the **customer matching**: **~90% of movements are assigned to the right account automatically**, which is why a single operator can run the whole billing operation. Built the tooling for the remaining ~10%: a hand-matching front end, aircraft-to-operator ownership rules, and a script that derives ownerships from six months of payment history.
- Built the **dunning / collections microservice** (sole author, from scratch) for overdue fee notices: escalation levels on a schedule, letters, fees, debtor blocking, and a person approving every batch before anything sends; the ledger is the single source of truth, so a posted payment closes the case on its own. Built the per-airport **document-dispatch platform** that delivers them: SMTP and OAuth mailers with multi-tier fallback and per-mailer token-bucket rate limiting.

Airport management system (PHP), a large part of it built module by module:

- Billing: **landing, parking, approach, and emission fees** for the airports themselves from the same flight-movement data (worked on); **recurring articles** for rent and other scheduled fees; **quotes** and **credit notes** modules; month-end collective invoicing runs; per-airport custom invoice templates with article translations.
- Operators and aircraft: the **aircraft ownership** feature (aircraft ↔ operator, with history); the movement-entry front end that sets the operator from the tail number automatically; the **debtor module** with account linking between local airport accounts and the global app account; **airport-sent registration links** so operators register, manage their account and add aircraft that are then matched automatically.
- Onboarding and support: importers for fee schedules and debtor accounts; a per-airport view of every billing run and audit fields recording which user changed what; the **fuel card** mechanism (an RFID card activates the pump, the fueling is matched to the operator, so shared cards still bill the right party).
- **Operator portal**, the pilot-facing side, architected and led from 2020: card and SEPA payments with 3D Secure, SEPA self-onboarding without a login.

Payments & reconciliation:

- Re-architected the platform's **Stripe payment infrastructure**, inherited as a Shopware plugin, into a central payment service in the core backend with an event-driven spine: **Stripe Connect** destination charges with automated fee-splitting and partner payouts, webhooks, **SEPA** mandate lifecycle, 3D Secure, multi-currency (EUR/CHF), handling **€8.47M** in gross volume across the platform.
- Built the **bank-statement import + payment-matching** system (auto-reconciles by invoice number + amount): ~80% of incoming customer payments match exactly, ~97% at least get an automatic candidate. Plus a refund/storno pipeline with retry and **idempotency** handling that eliminated duplicate billing, and SEPA exports in both directions.

Accounting & compliance (revenue you can defend in an audit):

- Built the German/EU **financial-compliance export suite** end-to-end: automated **DATEV** accounting exports, **XRechnung / EN 16931** e-invoicing built from the ground up with a validator, and a **SAP FI (RFBIBL) interface taken solo to two customer production go-lives** (with formal written customer sign-offs), plus **BMD** (Austria) and **CEGID** (France) exports, and the outgoing invoice book (Rechnungsausgangsbuch).

Platform engineering:

- Drove the migration of the billing domain out of the legacy shop system into the core backend with old-and-new parity as the measure of done; removed ~14k net lines of core-backend source (about 466 files) by moving it onto container autowiring and constructor promotion, behind a zero-diff harness, which became the team's coding standard.
- Built the test and CI platform (in-process transactional acceptance tests, self-hosted GitHub Actions runners, Playwright end-to-end suites, pre-commit static analysis); deployed and hotfixed production routinely and was the escalation point for billing and accounting questions, diagnosed from the database and the logs.
- **AI-augmented workflow in production**, tooling I built: a multi-agent orchestrator that specs, builds and tests features, and an AI Playwright pipeline that derives end-to-end suites from real use; run with verification at every step: plan first, a second model cross-checks, tests, then human review on every pull request.

Earlier

- **Java developer internship — SystemHaus** (Brazil, 2016), during electronics-technician training
- **Research internship — Weizmann Institute of Science** (Israel, Jul 2016): SVM-based image recognition

TECHNICAL STACK

Backend: Node.js/Express, PHP/Laravel · **Data:** MySQL/MariaDB (schema design, indexing, locking on billing tables), MongoDB · **Payments:** Stripe (Connect, Terminal, webhooks), SEPA · **Compliance:** DATEV, XRechnung / EN 16931, SAP FI (RFBIBL) · **Messaging:** RabbitMQ, DB-backed job queues · **Infra/CI:** Docker, GitHub Actions (self-hosted runners) · **Testing:** PHPUnit, Codeception, Jest, PHPStan, Playwright · **APIs:** REST, OpenAPI

EDUCATION

- **B.Sc. Computer Science — Uniritter (Brazil)**, in progress, expected 2027
- Prior CS coursework: **TU Dresden** (2019–2021); **UNISINOS** Computer Engineering (2017–2019)
- **Electronics technician** — Fundação Liberato technical school (Brazil), 2012–2017: Assembly, C, microcontrollers

AWARDS

ICYS Gold Medal, Engineering — Stuttgart (2017) · FEBRACE 2nd Place — São Paulo (2017) · Weizmann Institute International Summer Science Institute scholarship (2016)

The medals were for **Haptikos** (2015–16), a wearable for deafblind users: facial-expression recognition on a Raspberry Pi driving a vibration-motor matrix.