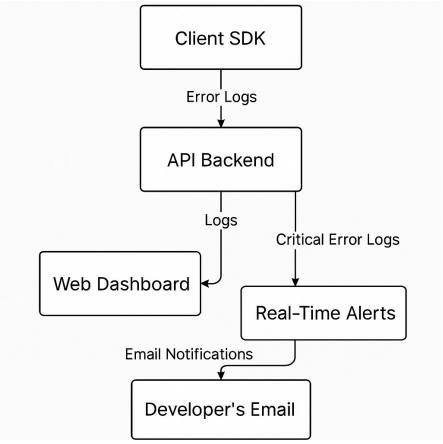
Error Logging Service Architecture Design



1. Client SDK

- **Language:** TypeScript
- **Bundler:** Vite

Justification:

- TypeScript is versatile, well-suited for both browser and mobile applications.
- Vite offers fast build times, simpler configuration, and excellent developer experience.
- SDK will capture uncaught exceptions, handled errors, and custom logs.

Alternative SDKs can be written in Python, PHP, etc., depending on client platform needs.

---

2. API Backend

- **Framework:** NestJS (TypeScript)

Justification:

- Strong TypeScript support.
- Built-in features (routing, DI, middleware) make it scalable and maintainable.

- Modular structure enables long-term project health.
- Integrated tools for validation, guards, authentication, logging.

---

3. Database

- **Primary DB:** PostgreSQL
- **ORM:** Prisma

Justification:

- PostgreSQL is ideal for structured data (user accounts, app metadata).
- JSONB column type enables storage of unstructured error logs.
- Prisma is TypeScript-native, well-supported, integrates smoothly with NestJS.

Future extension: Use ElasticSearch for high-performance log search and filtering.

---

4. Web Dashboard

- **Framework:** Next.js

Justification:

- Built-in routing and API support.
- Easy integration with Auth solutions (e.g., NextAuth).
- Easier to scale and maintain long-term
- Good TypeScript support.

---

5. Real-Time Alerts

- **Primary Service:** SendGrid
- **Alternative:** Nodemailer (SendGrid can be costly)

Justification:

- SendGrid is scalable, reliable, with good spam protection.
- Integrates easily with Node.js (official SDK).

---

6. DevOps & Deployment

- **Containerization:** Docker
- **CI/CD:** GitHub Actions (or GitLab CI)

Justification:

- Docker ensures consistency across development and production environments.
- GitHub Actions automates testing, linting, building, and deployment.

Alternatives:

- CI/CD: GitLab CI
- Monitoring: Prometheus + Grafana

Key Design Decisions

Real-time Processing

- Logs are analyzed on ingestion.
- Critical-level logs immediately trigger email alerts.
- WebSocket support (optional) for live updates on dashboard.

Log Retention Policies

- Logs older than a configurable threshold (e.g., 30 days) are archived or deleted.
- Implemented via scheduled cron jobs or background workers.

API Rate Limiting

- Per-app and per-user rate limits enforced via middleware.
- Protects against abuse and accidental flood.

Security

- Authentication: JWT (access & refresh tokens).
- Authorization: Role-based access control (admin, developer, viewer).
- All requests served over HTTPS.
- Logs sanitized to prevent injection.

DevOps Process

- `.env` files for configuration.
- Docker for containerized environments.
- CI/CD with GitHub Actions for testing, build, and deploy pipelines.
- Monitoring tools for performance and error tracking.

Developer Questions for Platform Owner

1. What platforms do clients of this new logging app use or what platforms are desirable?
2. Will users of the app have different roles?
3. Will developers be the only users of this platform?
4. How many users would have an app?
5. For how long we need to save our logs?
6. What alert types are needed (for mobile, email, some special massengers)?
7. Should we have some filters on a dashboard for logs (by platform, by user, etc.)?
8. Would be paid service or free? If paid, would it be some features or the whole system?

System Architecture Diagram