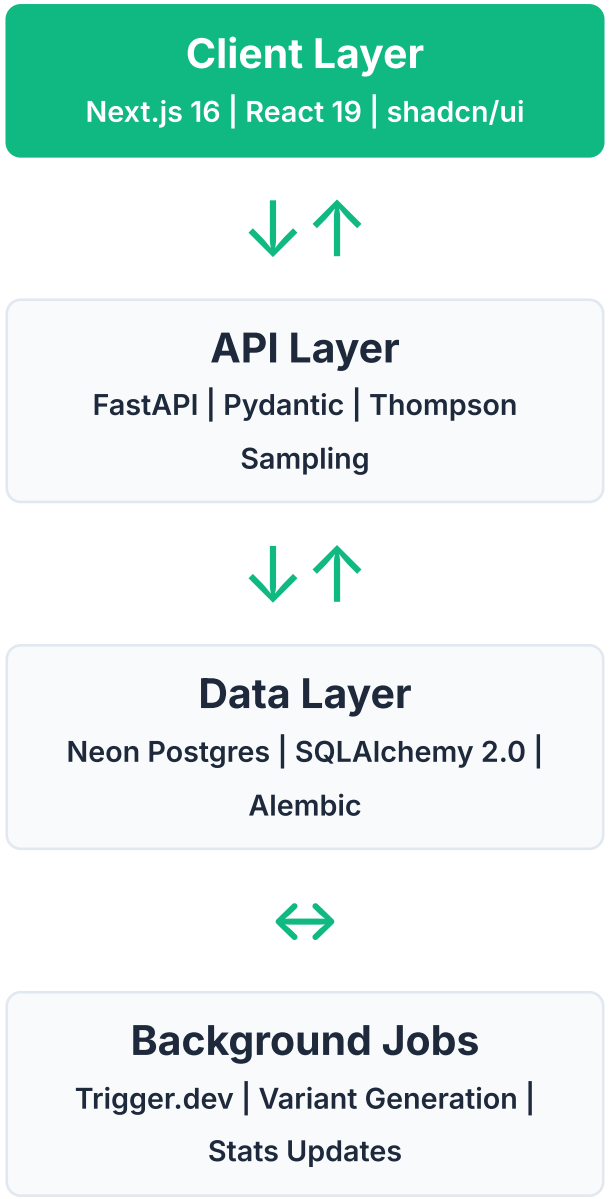
Evoloop

Technical Deep-Dive

Architecture | Testing | Algorithms | Infrastructure

Architecture Overview



Tech Stack

Frontend

Framework	**Next.js 16.0.10**
UI Library	**React 19.2.1**
Components	**shadcn/ui + Radix**
Styling	**Tailwind CSS 4**
Icons	**Lucide React**
Auth	**Stack Auth**

Backend

Framework	**FastAPI**
Validation	**Pydantic 2.0**
ORM	**SQLAlchemy 2.0**
Database	**Neon Postgres**
Migrations	**Alembic**
Jobs	**Trigger.dev**

Database Schema

Users

```
id: UUID (PK)
email: String(255)
password_hash: String
plan: "free" | "pro"
settings: JSON
```

Sites

```
id: UUID (PK)
user_id: FK → Users
url: String(2048)
status: analyzing|running
autonomy_mode: String
brand_constraints: JSON
```

Variants

```
id: UUID (PK)
site_id: FK → Sites
parent_variant_id: FK
patch: JSON
status: active|killed
generation_reasoning: Text
```

ExperimentStats

```
variant_id: UUID (PK)
visitors: Integer
conversions: Integer
alpha: Float (beta param)
beta: Float (beta param)
prob_best: Float
```

Events

```
id: UUID (PK)
site_id: FK (indexed)
variant_id: FK
visitor_id: String (indexed)
event_type: impression|conversion
created_at: DateTime (indexed)
```

ConversionGoals

```
id: UUID (PK)
site_id: FK → Sites
type: String(50)
config: JSON
created_at: DateTime
```

Thompson Sampling

Multi-armed bandit for optimal traffic allocation

```python
class ThompsonSampler:
    @staticmethod
    def select_variant(variants: List[Tuple[str, float, float]],
                       visitor_id: str = None) -> str:
        """Select variant using beta distribution sampling"""
        if visitor_id:
            # Deterministic assignment per visitor
            seed = int(hashlib.md5(visitor_id.encode()).hexdigest()[:8], 16)
            random.seed(seed)

        # Sample from beta distribution for each variant
        samples = [(v[0], random.betavariate(v[1], v[2])) for v in variants]

        # Return variant with highest sample
        return max(samples, key=lambda x: x[1])[0]
```

API Design

15+ RESTful endpoints with Pydantic validation

Protected Routes /api

POST	/auth/signup	User registration
POST	/auth/login	Authentication
GET	/sites	List user sites
POST	/sites	Create site
PATCH	/sites/:id	Update site
DELETE	/sites/:id	Delete (cascade)
GET	/variants	List variants
PATCH	/variants/:id	Update status
GET	/stats/site/:id	Aggregated stats

Public Routes /v1

GET	/assign	Get variant assignment
POST	/event	Track impression/conversion

Request Validation

```python
class EventRequest(BaseModel):
    site_id: str
    variant_id: str
    visitor_id: str
    type: Literal["impression", "conversion"]
    metadata: Optional[Dict] = None
```

Test Coverage

90%

Minimum Required

--cov-fail-under=90

Test Framework: pytest + pytest-cov

Async Support: pytest-asyncio

Test DB: SQLite in-memory

Test Suite (26 tests)

test_auth.py	4 tests	Signup/Login
test_sites.py	5 tests	CRUD ops
test_variants.py	5 tests	Lifecycle
test_runtime.py	4 tests	Assignment
test_thompson.py	9 tests	Algorithm
test_health.py	1 test	Health check

Frontend Architecture

Component Structure

```
/components
├── ui/              # 16 shadcn primitives
│   ├── button.tsx
│   ├── card.tsx
│   ├── dialog.tsx
│   └── ...
├── dashboard/
│   ├── header.tsx
│   └── sidebar.tsx
└── landing/
    ├── animated-section.tsx
    └── variant-visualization.tsx
```

Key Patterns

App Router

Next.js 16 with React Server Components

Type Safety

Full TypeScript, strict mode enabled

Auth Integration

Stack Auth with StackProvider wrapper

Testing

Vitest + React Testing Library

Background Jobs

Trigger.dev for async variant generation and stats updates

1

Page Analysis

Scrape URL, extract brand constraints, identify mutable elements

2

Variant Generation

LLM-powered copy, optional image generation via OpenRouter

3

Stats Update

Aggregate events, recalculate beta params, update prob_best

```
// trigger/generate-variants.ts
export const generateVariants = task({
  id: "generate-variants",
  run: async (payload: { siteId: string }) => {
    const site = await getSite(payload.siteId);
    const constraints = site.brand_constraints;

    const variants = await openrouter.generate({
      model: "anthropic/claude-sonnet",
      prompt: buildVariantPrompt(site, constraints)
    });

    await saveVariants(site.id, variants);
```

Security & Auth

Authentication Flow

Frontend: Stack Auth

OAuth providers, session management, protected routes

Backend: Custom + Salt

```python
def hash_password(password: str) -> str:
    salt = os.environ.get("PASSWORD_SALT")
    return hashlib.sha256(
        f"{password}{salt}".encode()
    ).hexdigest()
```

Security Measures

- **UUID Primary Keys** - Non-sequential, unpredictable
- **User Isolation** - All queries filtered by user_id
- **Cascade Deletes** - No orphaned data
- **Input Validation** - Pydantic on all endpoints
- **Email Validation** - EmailStr type enforcement
- **Environment Secrets** - No hardcoded credentials

Deployment Architecture

Vercel

- Next.js frontend (Edge Runtime)
- Python API (Serverless Functions)
- Automatic preview deployments
- Global CDN distribution

```
// vercel.json
{
  "rewrites": [
    { "source": "/api/:path*",
      "destination": "/api/index.py" },
    { "source": "/v1/:path*",
      "destination": "/api/index.py" }
  ]
}
```

Neon Postgres

- Serverless PostgreSQL
- Auto-scaling to zero
- Database branching for previews
- Connection pooling built-in

```
# Connection with health check
engine = create_engine(
    DATABASE_URL,
    pool_pre_ping=True
)
```

By The Numbers

90%
Test Coverage

26
Test Cases

15+
API Endpoints

6
Database Tables

20+
UI Components

10K
Monte Carlo Sims

Production-ready. Type-safe. Statistically rigorous.