個人開発プロジェクト

Online Chat Service

使用技術	フロントエンド: Streamlit バックエンド: Python インフラ: Docker Compose その他: PyCryptodome, GitHub Actions
GitHub	`github.com/BackendExplorer/Online-Chat-Service`
期間	2025 年 1 月〜5 月（約 5 カ月）

概要

独自ソケット通信と RSA ＋ AES 暗号を用いたリアルタイムチャットサービス。ホストがルームを作成し、ゲストが参加。マルチスレッドで複数ルームに対応し、無操作 5 分で自動退出。Docker ＋ GitHub Actions で安定運用。

機能と技術のこだわり

- TCP ソケットでルーム制御、UDP でメッセージ送信
- RSA ＋ AES による暗号通信を独自実装
- マルチスレッドによる複数接続処理
- タイムアウト機能による自動退出
- Docker Compose 構築、GitHub Actions でビルドテスト

工夫した点

- 固定長ヘッダーでプロトコルを柔軟・拡張可能に設計
- TCP/UDP サーバを並列起動し、リアルタイム性と信頼性を両立
- GitHub Actions による Docker のビルドテストを実装

個人開発プロジェクト

Video Processor

使用技術	フロントエンド: Streamlit バックエンド: Python データベース: SQLite インフラ: Docker Compose その他: FFmpeg, GitHub Actions
GitHub	`github.com/BackendExplorer/Video-Processor`
期間	2025 年 1 月〜5 月（約 5 カ月）

概要

Python と FFmpeg で開発した動画変換サービス。動画アップロード後、RSA ＋ AES による暗号通信を通じて安全に受信し、圧縮・リサイズ・音声抽出・GIF 生成などを高速変換。プロトコル設計から変換制御までを独自に実装。

機能と技術のこだわり

- FFmpeg で動画圧縮・解像度変換・音声抽出・GIF 作成に対応
- TCP ベースの独自プロトコル（MMP）で動画・音声・JSON を安全に送受信
- ハイブリッド暗号（RSA ＋ AES）による安全な通信経路の確立
- SQLite を用いたログ記録と、Streamlit ベース UI による結果の可視化
- Docker Compose 構成と GitHub Actions によるビルドテスト

工夫した点

- 固定長ヘッダーによるパケット解析の容易化と異種メディア対応の柔軟性
- 通信・暗号・動画処理を責務ごとにクラス分離し、保守性を向上
- 暗号鍵交換に RSA、ファイル転送に AES-CFB を用いてセキュア転送を実現

個人開発プロジェクト

Portfolio Site

使用技術	フロントエンド: HTML / CSS / Vanilla JavaScript Web サーバ: Nginx HTTPS: Let's Encrypt（Certbot） デプロイ: rsync + SSH（deploy.sh） その他: UFW, fail2ban, unattended-upgrades（運用設定）
GitHub	`github.com/BackendExplorer/portfolio-site`
期間	2025 年 11 月〜12 月（約 1 カ月）

概要

自分のレジュメと制作物を公開する静的ポートフォリオサイト。共通ヘッダー/フッターをビルド無しで差し込み、作品一覧と詳細は JSON（projects.json）から自動レンダリング。Nginx で静的配信し、Let's Encrypt で HTTPS 化、rsync+SSH による 1 コマンドデプロイまで含めて「開発→公開→運用」を一通り構築。

機能と技術のこだわり

- 共通ヘッダー/フッターを include.js（fetch）で差し込み、全ページの統一レイアウトを実現
- projects.json を編集するだけで作品一覧/詳細ページが更新される JSON 駆動レンダリング
- Resume ページと PDF 配布（/assets/resume.pdf）により、閲覧・DL 導線を整理
- Nginx による静的配信と、Certbot による HTTPS 証明書の自動発行・自動更新（certbot.timer）
- deploy.sh による rsync –delete 同期と nginx -t → reload のデプロイ導線を整備

工夫した点

- public/ をドキュメントルートに固定し、ローカル/本番で同一パス体系を維持（404 の手戻りを防止）
- rsync 後に nginx -t を挟むことで、設定ミスによる停止事故を予防
- SSH 鍵ログイン・UFW・fail2ban・自動更新など、最小限のサーバセキュリティを運用に組み込み