README

プロジェクト名: (例) TechFeed Hub - テックブログ情報収集効率化サービス

1. 概要 (Overview)

1.1. プロジェクトの目的

各社から発信される技術ブログの情報を効率的に収集・閲覧できるWebサービスを開発する。情報の洪水の中から、ユーザーが必要とする知識やトレンドを素早くキャッチアップできる環境を提供し、エンジニアの学習・情報収集体験を向上させる。

1.2. 解決する課題

- 購読したいテックブログが多すぎて、個別にチェックするのが大変
- 興味のない記事（例: イベントレポート、採用情報）がノイズになる
- 後で読みたい記事が溜まっていき、管理しきれない
- 複数のブログを横断して特定の技術について検索したい

1.3. ターゲットユーザー

- 最新の技術動向や他社の事例に関心が高いWebエンジニア
- 効率的な情報収集方法を模索している全てのIT技術者

ターゲットセグメントの精緻化

プライマリターゲット：成長期のミドルエンジニア（経験3-7年）

- **ペインポイント**：技術選定の意思決定に必要な実践的知見が散在している
- **ジョブ**：チームの技術的課題を解決するための事例・ベストプラクティスを探す
- **成功指標**：収集した情報から実際にプロジェクトに適用できた施策数

セカンダリターゲット：テックリード・エンジニアリングマネージャー

- **ペインポイント**：チーム全体の技術力底上げに必要な学習リソースの選定が困難
- **ジョブ**：チームメンバーに共有すべき重要な技術トレンドを把握する
- **成功指標**：チーム内での技術知識共有の活性化度

2. コンセプト

「AI要約付きの、超パーソナライズ可能なテックブログリーダー」

- ユーザーが本当に読みたい記事だけを、最適な形で届ける。

- AIの力で「読む」手間を「把握する」手間に変える。
- 将来的には「聞く」という選択肢も提供する。

3. 機能要件 (Functional Requirements)

3.1. MVP (Minimum Viable Product) の機能

> まずはコアとなる価値を最速で提供するための最低限の機能群。

- **F-01: RSS/Atomフィードの定期収集**
  - 事前に登録されたテックブログのRSS/Atomを定期的に巡回し、新規記事をDBに保存する。
- **F-02: 記事一覧表示**
  - 収集した記事を時系列（新着順）で一覧表示する。
  - 記事タイトル、ブログ名、発行日、要約（あれば）を表示する。
- **F-03: 全文検索**
  - 記事のタイトルや本文を対象にキーワード検索ができる。
- **F-04: レスポンシブデザイン**
  - PC、タブレット、スマートフォンなど、どのデバイスでも快適に閲覧できる。

3.2. 拡張機能 (Future Extensions)

> MVPリリース後の差別化・価値向上を目指す機能群。

Phase 2: パーソナライズ

- **F-11: 購読ブログの選択機能**
  - ユーザーが一覧から購読したいブログを自由に選択・解除できる。
- **F-12: キーワードフィルタリング機能**
  - `React`, `Go`, `AWS` など、登録したキーワードが含まれる記事をハイライト表示する。
- **F-13: ミュート機能**
  - 興味のない企業や `登壇レポート`, `入社エントリー` といった特定のキーワードを含む記事を非表示にする。

Phase 3: AI活用

- **F-21: AIによる自動要約**
  - 記事本文をAIで要約し、一覧画面や詳細画面に表示する。

- **F-22: AIによる自動タギング**
  - 記事内容を解析し、`フロントエンド`, `インフラ` などの技術タグを自動付与する。
- **F-23: 関連記事レコメンド**
  - 記事内容の類似度に基づき、関連性の高い他の記事を推薦する。

Phase 4: 音声配信

- **F-31: 記事の音声化**
  - AI要約または記事全文をText-to-Speechで音声化し、再生できるようにする。

4. 技術スタック (Technology Stack)

領域	技術/サービス	目的/理由
フロントエンド	Next.js or React	モダンなUI/UXと開発体験。静的サイト生成(SSG)
バックエンド	Java (Spring Boot)	堅牢なAPI開発。今後の業務への応用。
データベース	Amazon RDS (PostgreSQL)	高機能なリレーショナルDB。管理の容易さ。
API実行環境	AWS App Runner	コンテナベースのフルマネージドサービス。デプロイの簡素化。
バッチ処理	AWS Lambda (Java)	RSS収集処理。サーバーレスでコスト効率が良い。
バッチトリガー	Amazon EventBridge	Lambdaを定期実行するためのスケジューラ。
静的ファイル	AWS S3 + CloudFront	フロントエンドのホスティング。高速・高耐久・低コスト。

AI (将来)	Amazon Bedrock / Polly	AI要約 / Text-to-Speech

5. アーキテクチャ (Architecture)

5.1. システム構成図（テキスト表現）

```

```

```
+----------------------+
|   Users (Browser)    |
+-----------+----------+
            |
+-----------v----------+
|   Amazon CloudFront  | (CDN, HTTPS)
+-----------+----------+
  /                    \\
```

```

+---------v---------+ +---v-----------------+

| AWS S3 | | AWS App Runner |

| (Frontend Files) | | (Spring Boot API) |

+-------------------+ +----------+----------+

|

+----------v----------+

| Amazon RDS (DB) |

+---------------------+

^

|

+----------------------+ |

| Amazon EventBridge |------+

| (Scheduler) |

+-----------+----------+

| Triggers

+-----------v----------+

| AWS Lambda (Java) |

| (RSS Fetcher) |

```

```
+----------------------+

```

5.2. データフロー

1. **定期バッチ処理:** EventBridgeがLambdaを定刻に起動 → Lambdaが各ブログのRSSを取得し、新規記事をRDSに保存。
2. **ユーザーアクセス:** ユーザーはCloudFront経由でS3上のWebサイトにアクセス。
3. **API通信:** Webサイト(Next.js)は、記事データなどをApp Runner上で動くAPIにリクエスト → APIはRDSからデータを取得して応答。

6. データベース設計（簡易版）

- **blogs** テーブル
  - `id` (PK), `name`, `site_url`, `rss_url`, `created_at`, `updated_at`
- **articles** テーブル
  - `id` (PK), `blog_id` (FK), `title`, `url`, `summary`, `published_at`, `ai_summary`, `created_at`, `updated_at`
- **users** テーブル (※パーソナライズ機能実装時)
  - `id` (PK), `username`, `email`, `password_hash`, `created_at`
- **subscriptions** テーブル (※パーソナライズ機能実装時)
  - `user_id` (FK), `blog_id` (FK)

7. 開発マイルストーン (Milestones)

1. **M1: 環境構築 (1週目)**
   - AWSアカウントセットアップ、IAMユーザー作成。
   - GitHubリポジトリ作成。
   - Spring Boot / Next.js のプロジェクト雛形作成。
2. **M2: バッチ処理の実装 (2週目)**
   - Lambda関数（Java）でRSSを取得し、RDS(PostgreSQL)に保存する処理を実装。
   - EventBridgeで定期実行設定。
3. **M3: MVPの完成 (3-4週目)**
   - App RunnerでAPIをデプロイ。
   - S3+CloudFrontでフロントエンドをデプロイ。
   - 記事一覧表示と検索機能を実装。
4. **M4以降: 拡張機能開発**

- パーソナライズ機能（Phase 2）から順次実装。
- 各機能はGitHubのIssueでタスク管理を行う。