SportEase の情報の取り扱い

佐藤佑作

2026 年 2 月 9 日

1　情報の取り扱い

このアプリは Google アカウントでユーザー認証を行っている 。高専のメールアドレスでのみログインできるように、ホワイトリストで制限している。

利用する Google アカウント情報は Google メールアドレスのみである 。

- Google メールアドレス

使用用途は以下の通りである：

表 1　情報の使用用途

情報	用途
メールアドレス	ユーザー認証、アプリ内でユーザーを一意に識別するための ID

ユーザーの表示名に関しては、初回ログイン時に以下の画面で設定し、その後自由に変更可能にできる 。

図 1　初回ログイン時の表示

2　外部設計

2.1　ユーザーロール

本システムでは以下の 3 つのユーザーロールを定義し、権限管理を行っている。
必要に応じてロールを増やすことはできるが、権限管理はこの 3 つのベースロールで行っている。

表 2　ユーザーロール一覧

ロール名	概要
Student (一般学生)	主に情報の閲覧（対戦表、試合結果、クラスの進捗）および通知の受信を行う。
Admin (行事委員)	イベント全体の管理、試合結果の入力・承認、スケジュールの変更、MVP 投票の管理など、運営に必要な全機能にアクセス可能。
Root (システム管理者)	システム設定、ホワイトリスト管理、マスターデータ（スポーツ、クラス等）の管理を行う最高権限。

2.2　機能要件

各ロールにおける利用可能な機能の詳細は以下の通りである。

2.2.1　Student (一般学生)

一般的な学生ユーザーに割り当てられるロールであり、以下の機能を利用可能である。

- **アカウント管理**:
  - Google アカウントを用いたログイン・ログアウト
  - プロフィール設定（表示名、所属クラスの登録）
- **情報閲覧**:
  - 開催中のイベント情報の閲覧
  - 各競技のトーナメント表・対戦組み合わせの閲覧
  - 試合結果および勝敗のリアルタイム確認
  - クラス別の総合得点・進捗状況・順位の閲覧

  - 昼企画（リレー・綱引き等）の結果閲覧
- **チーム管理（自クラスのみ）**:
  - 所属クラスのチームメンバー編成機能（メンバーの追加・削除）
  - チームメンバーの確認
- **通知・その他**:
  - 試合開始・結果等の Web Push 通知の受信
  - MVP 投票への参加（投票権がある場合）

2.2.2 Admin (行事委員)

イベント運営の実務を担当するロールであり、Student の全機能に加え、以下の管理機能を利用可能である。

- **試合進行管理**:
  - 試合開始時間の変更・調整
  - 試合結果（スコア・勝者）の入力・確定
  - 試合ステータスの更新（試合中、終了など）
  - 昼企画（リレー・綱引き等）の予選・決勝結果の入力
- **運営管理**:
  - 前日・当日の出欠確認・登録
  - 各競技の参加定員数の調整・変更
  - ユーザーの表示名修正（不適切な名前の修正等）
  - 運営用画像・PDF 資料（要項・ルールブック等）のアップロード
  - MVP 投票状況の確認・集計
  - 行事委員権限の付与・剥奪

2.2.3 Root (システム管理者)

システムの全権限を持つロールであり、マスターデータの管理を含む以下の高度な機能を利用可能である。

- **システム・イベント設定**:
  - ログイン許可リスト（ホワイトリスト）の管理（追加・削除・一括インポート）
  - 新規イベントの作成およびアクティブイベントの切り替え
  - 雨天モード（スケジュール短縮・変更時）の切り替えおよび設定管理
- **マスターデータ管理**:

  - 競技種目の新規作成・削除
  - クラス情報の管理および在籍学生数の登録（CSV インポート対応）
  - 昼企画のグループ分け・テンプレート管理
  - 全競技のトーナメント表の一括自動生成・プレビュー
- **全体通知**:
  - 全ユーザーまたは特定のロールに向けた任意のお知らせ通知の配信

2.3 画面構成・遷移

本システムは Web ブラウザ上で動作する SPA (Single Page Application) であり、以下の画面構成を持つ。

- **ログイン画面**: Google アカウントによる認証エントリポイント。
- **ダッシュボード (ホーム)**: ログイン後の初期画面。ユーザーロールに応じたメニューを表示。
- **学生向けメニュー (Student)**:
  - **マイページ**: プロフィール設定、クラス情報確認
  - **競技・イベント情報**: トーナメント表閲覧、競技詳細、点数一覧
  - **昼企画**: 昼競技の結果確認
  - **その他**: QR コード表示、通知確認、通知申請
- **運営管理者メニュー (Admin)**:
  - **試合管理**: 試合結果入力、トーナメント進行管理、昼競技結果入力
  - **参加管理**: クラス・チーム編成管理、QR コード読み取り・参加確認、出席登録
  - **運営設定**: ロール管理、競技詳細設定、MVP 投票管理
- **システム管理者メニュー (Root)**:
  - **マスター管理**: イベント作成、競技種目設定、ホワイトリスト管理
  - **全体設定**: 雨天モード設定、トーナメント一括生成、クラス人数設定
  - **コンテンツ管理**: 競技要項アップロード、MVP 確認

2.4 動作環境

- **クライアント端末**: スマートフォン（iOS/Android）、タブレット、PC
- **推奨ブラウザ**: Google Chrome, Safari, Microsoft Edge (各最新版)
- **ネットワーク**: インターネット接続必須（学内 Wi-Fi またはモバイルネットワーク）

2.5 非機能要件

- **セキュリティ**:
  - Google OAuth2 (OpenID Connect) による堅牢な認証
  - 指定ドメイン (@sendai-nct.jp / .ac.jp) およびホワイトリストによるアクセス制限
  - 常時 SSL/TLS 化 (HTTPS) による通信の暗号化
- **パフォーマンス・可用性**:
  - WebSocket を用いたリアルタイムな試合経過・結果の配信
  - コンテナ技術 (Docker) による環境の一貫性と高い移植性

3 内部設計

3.1 システムアーキテクチャ

本システムは Docker コンテナを用いたマイクロサービス構成を採用している。

- **Reverse Proxy (Traefik)**: エントリーポイントとして HTTPS 通信の終端、Let's Encrypt による SSL 証明書の自動更新、および Frontend/Backend へのルーティングを行う。
- **Frontend (Frontapp)**: SvelteKit (Node.js) を用いた SSR/CSR ハイブリッド構成。ユーザーインターフェースを提供。
- **Backend (Backapp)**: Go 言語 (Gin Framework) による REST API サーバー。ビジネスロジックおよび WebSocket によるリアルタイム配信を担当。
- **Database (DB)**: MySQL 8.0。永続化データを管理。

3.2 技術スタック

3.3 データモデル設計

主なエンティティとその役割は以下の通りである。

- **Users**: ユーザー ID、メールアドレス、ロール、表示名を管理。
- **Events**: 球技大会等のイベント単位。アクティブなイベントの設定が可能。
- **Sports / Classes**: 競技種目および参加クラスのマスターデータ。
- **Teams / Entries**: クラス・競技ごとのチーム編成およびエントリー情報。

表 3　採用技術一覧

カテゴリ	技術
フロントエンド	SvelteKit, TailwindCSS, TypeScript
バックエンド	Go 1.24, Gin, Gorilla WebSocket
データベース	MySQL 8.0
インフラ	Docker, Docker Compose, Traefik v2
認証	Google OAuth2 (OpenID Connect)

- **Tournaments / Matches**: トーナメント構造および個々の試合データ（開始時刻、スコア、勝者）。
- **Notifications**: Web Push 通知の購読情報および通知履歴。

3.4　API 設計

RESTful API を採用し、リソースごとにエンドポイントを定義している。

- `/api/auth`: 認証関連（Google ログイン、プロファイル取得）
- `/api/events`: イベント情報の取得・更新
- `/api/admin`: 管理機能全般（試合結果更新、マスタ管理）
- `/api/student`: 学生向け読み取り専用 API
- `/api/ws`: WebSocket エンドポイント（トーナメント表のリアルタイム更新等）

3.5　ディレクトリ構成

本システムの主要なソースコード構成は以下の通りである。

- `frontapp/` (Frontend: SvelteKit)
  - `src/routes/`: ページコンポーネントおよび SvelteKit のリ API エンドポイント。
  - `src/lib/`: 再利用可能な UI コンポーネント、ユーティリティ関数、ストア（状態管理）。
  - `static/`: 静的ファイル（画像、フォント等）。
- `backapp/` (Backend: Go)
  - `cmd/`: アプリケーションのエントリーポイント (main.go)。
  - `internal/router/`: Gin フレームワークを用いたルーティング定義。
  - `internal/handler/`: HTTP リクエストを処理するビジネスロジック層。

- `internal/repository/`: データベース操作を抽象化するデータアクセス層。
- `internal/models/`: データベースのテーブル構造に対応する Go 構造体定義。
- `db/migrations/`: データベースのスキーマ定義およびマイグレーション用 SQL ファイル。

3.6 データベース設計

主要なテーブルとその役割は以下の通りである。

表 4 主要テーブル一覧

テーブル名	概要
`users`	ユーザー情報（UUID, メールアドレス, 表示名, ロール等）
`classes`	クラス情報（クラス名, 学生数, イベント ID）
`events`	イベント情報（大会名, 年度, 開催期間）
`sports`	競技種目マスター（サッカー, バスケ等）
`teams`	クラス・競技ごとのチーム情報
`matches`	試合情報（トーナメント位置, 対戦チーム, スコア, 勝者）
`notifications`	Web Push 通知の送信ログと内容
`noon_game_*`	昼企画（リレー等）に関連する設定・結果データ群

3.7 処理フロー例 (トーナメント進行)

試合結果が入力され、トーナメント表が更新されるまでのデータフローは以下の通りである。

1. **結果入力**: Admin ユーザーが管理画面から試合のスコアを入力し、確定ボタンを押下する。
2. **API リクエスト**: フロントエンドから `POST /api/admin/match/result` が送信される。
3. **バックエンド処理**:
   - `matches` テーブルの該当レコード（スコア、勝者 ID、ステータス）を更新する。
   - 勝者が決定した場合、トーナメントの構造に基づき、次戦 (`next_match_id`) の対戦チームとして勝者を自動設定する。

- WebSocket を通じて、接続中の全クライアントに対して「試合更新イベント」をブロードキャストする。

4. **リアルタイム更新**:
   - 各クライアント（Student/Admin 画面）が WebSocket メッセージを受信する。
   - トーナメント表コンポーネントが再レンダリングされ、リロードなしで最新の勝敗結果が反映される。

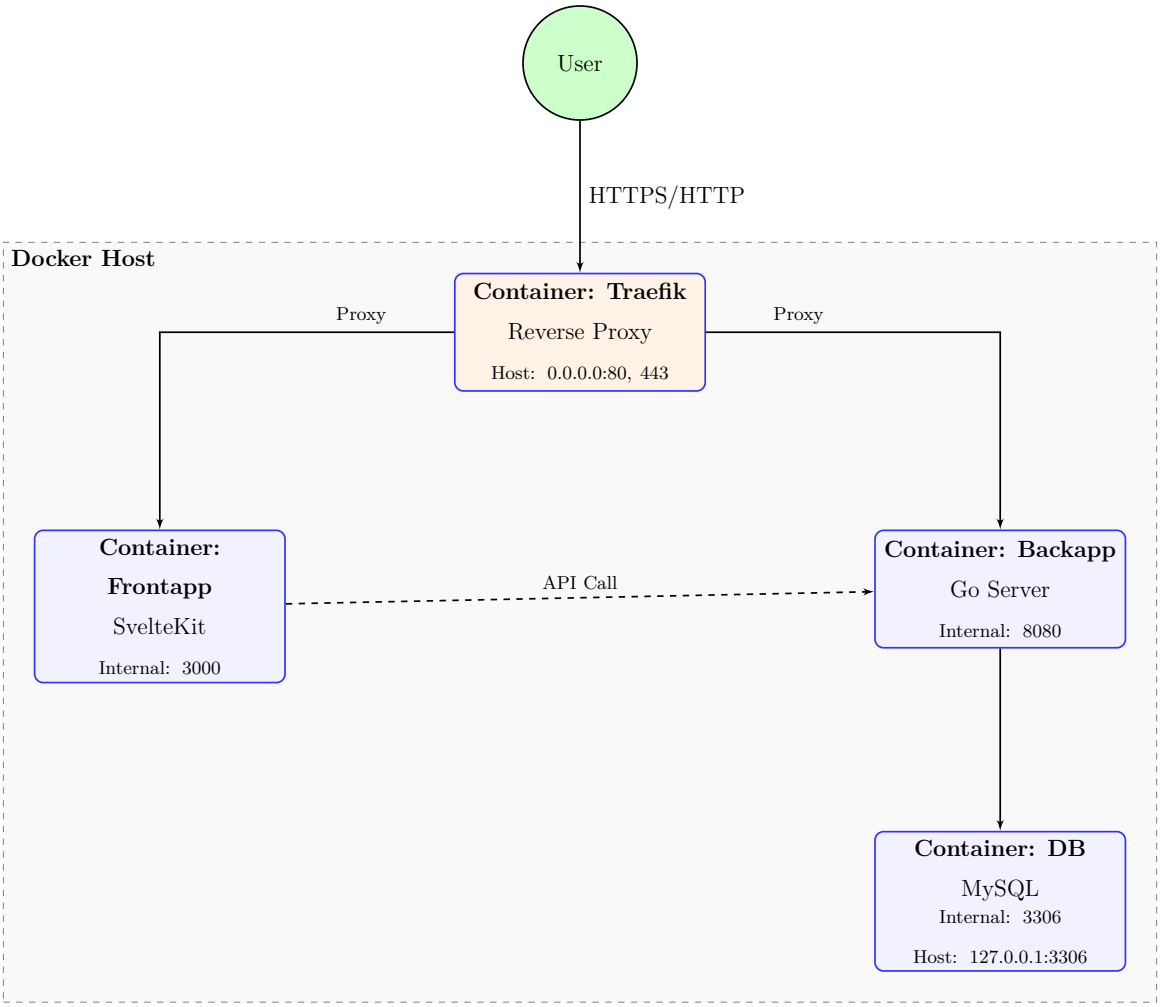

図 2　システム構成図（Docker コンテナ構成）