DOCUMENTAZIONE TECNICA — WEB APP

1. Titolo del Progetto

FinDrink

Nicolo' Romito & Stefano Montuori

"Esplora, crea, assapora."

La missione di **FinDrink** è quella di offrire un ambiente digitale innovativo in cui gli utenti possano esplorare, cercare e creare cocktail originali su misura dei propri gusti.
Attraverso un sistema intelligente di raccolta e analisi delle preferenze, **FinDrink** permette agli amministratori di comprendere tendenze, preferenze e abitudini degli utenti, migliorando costantemente l'esperienza e anticipando i desideri degli appassionati di mixology.

2. Introduzione

Obiettivo del documento

Questo documento ha come obiettivo principale quello di presentare il progetto **FinDrink**, illustrando nel dettaglio l'idea, gli obiettivi, le funzionalità principali e l'architettura tecnica della web application.

Descrizione generale della finalità di questo elaborato

L'elaborato ha lo scopo di fornire una visione chiara e completa della piattaforma **FinDrink**, ponendo particolare enfasi sulla sua capacità di offrire agli utenti un ambiente digitale innovativo per la ricerca, la creazione e la personalizzazione di cocktail. Vengono inoltre descritte le tecnologie adottate, le modalità di raccolta e analisi dei dati relativi alle preferenze degli utenti e i vantaggi concreti offerti sia agli utenti finali che agli amministratori del sistema.

3. Panoramica della webapp

La web application **FinDrink** è uno strumento progettato per utenti appassionati di cocktail e mixology, pensato per favorire la scoperta, la personalizzazione e la creazione di cocktail originali, attraverso un database costantemente aggiornato e ricco di informazioni.

3.1 Funzionalità principali

- **Ricerca intelligente e personalizzata**
  Gli utenti possono cercare cocktail e ingredienti utilizzando filtri avanzati e ricevere risultati personalizzati in base a preferenze e restrizioni (alcolici, analcolici, ecc.).

- **Creazione e gestione di cocktail originali**
  Gli utenti autenticati possono proporre nuove ricette che vengono revisionate e approvate dagli amministratori, arricchendo continuamente il database interno con cocktail innovativi.

- **Sincronizzazione con TheCocktailDB**
  FinDrink integra un sistema automatizzato di importazione e sincronizzazione periodica dal database esterno **TheCocktailDB**, garantendo un catalogo sempre aggiornato e completo, con le ultime novità e ricette riconosciute a livello internazionale.

3.2 Use Case principali

Utente unlogged

L'utente non autenticato ha un accesso limitato alla piattaforma per garantire il rispetto del **GDPR** e la protezione dei dati personali. Per motivi legati alla tutela dei minori e alla necessità di fornire contenuti personalizzati, **FinDrink** non consente interazioni agli utenti non autenticati, richiedendo obbligatoriamente la registrazione e l'accettazione delle policy.

Utente logged

L'utente autenticato può utilizzare completamente la piattaforma, ma le funzionalità offerte si diversificano ulteriormente:

- **Maggiorenne con visualizzazione cocktail alcolici**
  Ha accesso completo a tutte le ricette, incluse quelle alcoliche, con la possibilità di proporre e creare nuove ricette.
- **Minorenne (o maggiorenne che non desidera vedere risultati alcolici)**
  Ha accesso esclusivamente ai cocktail analcolici, garantendo una navigazione sicura e appropriata alla fascia d'età o alle preferenze individuali.

Inoltre, ciascun utente può scegliere se fornire o meno il consenso al profiling dei dati:

- **Consenso al profiling attivo**
  L'utente beneficia di una ricerca e raccomandazioni personalizzate basate sul suo storico e preferenze raccolte, migliorando notevolmente l'esperienza utente.
- **Consenso al profiling disattivato**
  L'utente può utilizzare tutte le funzionalità generali della piattaforma, ma **FinDrink** non raccoglierà o analizzerà alcun dato personale relativo alle sue ricerche o preferenze.

4. Architettura di sistema

4.1 Architettura Generale

FinDrink adotta un'architettura a microservizi basata su container Docker, orchestrati tramite Docker Compose, che garantisce scalabilità, isolamento e facilità di deploy.
Al centro troviamo un **API Gateway** che instrada tutte le richieste client ai vari microservizi backend (AuthService, UserProfileService, CocktailService, CocktailImportService, SearchService, SearchHistoryService, FavoriteCocktailsService, CocktailSubmissionService, ImageFetcherService).
La persistenza è affidata a un database **PostgreSQL**, mentre la sincronizzazione periodica con **TheCocktailDB** viene gestita da un microservizio dedicato (CocktailImportService) con job di retry automatico. Il frontend, sviluppato in **Angular** (TypeScript + Reactive Forms + PrimeNG), comunica con il gateway via HTTP/HTTPS, gestendo autenticazione JWT e autorizzazioni per ruolo.
Per il monitoraggio e il logging sono configurati **Prometheus** e **Grafana** in container separati, integrati nel Docker Compose, permettendo di analizzare metriche di utilizzo, tassi di errore e prestazioni in tempo reale.

4.2 Componenti Principali

- **Frontend**
  - **Tecnologie**: Angular 19.2.0 (TypeScript, Reactive Forms), PrimeNG 19.0.10 per layout e componenti UI, Chart.js per grafici, jwt-decode per la decodifica dei token JWT
  - **Configurazione**: progetto "cocktail-app" con CLI Angular, file angular.json e tsconfig.json standard, dipendenze gestite via npm.
  - SCSS come linguaggio di stile (inlineStyleLanguage: "scss", stile predefinito per i componenti)
  - SCSS come linguaggio di stile (inlineStyleLanguage: "scss", stile predefinito per i componenti)
  - **Assets** gestiti via glob da public/
  - **Polyfills** caricati (zone.js e supporto a @angular/localize) per internazionalizzazione
  - **Budget di bundle** in produzione (max 500 kB per iniziale, 4 kB per stile componente) e **output hashing** (outputHashing: all)
  - Configurazioni separate per **development** (no ottimizzazioni, source maps) e **production** (minification, estrazione licenze, hashing)

- **Backend**
  - **Piattaforma & Framework**
    - .NET 8.0 (TargetFramework net8.0), SDK Microsoft.NET.Sdk.Web
    - Nullable reference types e implicit usings abilitati per migliore produttività e qualità del codice
  - **Autenticazione & Sicurezza**
    - Microsoft.AspNetCore.Authentication.JwtBearer (v8.x) per gestione JWT
    - System.IdentityModel.Tokens.Jwt (v8.8.0) per creazione e validazione token
  - **Data Access & ORM**
    - Entity Framework Core **9.0.4** (Microsoft.EntityFrameworkCore, Design, Tools)
    - Provider PostgreSQL: Npgsql.EntityFrameworkCore.PostgreSQL (v9.0.4)
  - **Documentazione & API Explorer**
    - Swashbuckle.AspNetCore (v6.6.2) per generazione Swagger/OpenAPI
  - **Integrazione HTTP & Proxy**
    - Microsoft.Extensions.Http (v9.0.4) per HttpClientFactory e resilienza
    - Yarp.ReverseProxy (v2.3.0) impiegato nel **Gateway** per instradamento dinamico delle richieste
  - **Serializzazione**
    - Newtonsoft.Json (v13.0.3) per flessibilità nelle operazioni JSON
  - **Microservizi & Progetti**
    La soluzione Visual Studio include i seguenti progetti, tutti su .NET 8.0 e containerizzati tramite Docker Compose:
    - **AuthService**
    - **Gateway**
    - **UserProfileService**
    - **CocktailImportService** (sincronizzazione TheCocktailDB + retry migrations)
    - **CocktailService**
    - **CocktailSubmissionService**
    - **FavoriteCocktailsService**
    - **SearchService**
    - **SearchHistoryService**
    - **ImageFetcherService**

- **Monitoraggio & DevOps**
  - Docker Compose per orchestrazione container
  - Prometheus + Grafana per raccolta metriche e dashboard in tempo reale
  - Configurazione centralizzata delle variabili via .env (utilizzo di env_file in docker-compose.yml)

- **Documentazione API**
  - Disponibile in formato Postman-flavored Markdown e interattiva su Postman: https://documenter.getpostman.com/view/38702488/2sB2jAbTKT

5. Tecnologie Utilizzate

Componente	Linguaggio / Framework	Versione	Note
Frontend	Angular (TypeScript, Reactive Forms, PrimeNG, SCSS, Chart.js, jwt-decode)	19.2.0; 19.0.10; 4.4.9; 4.0.0	SCSS inline; Production budgets; Dev/Prod configs
Shared Library	Angular Library (ng-packagr, SCSS)	19.2.0	Componenti condivisi
AuthService	ASP.NET Core (net8.0), EF Core, JWT, Swagger	net8.0; EF Core 9.0.4; JwtBearer 8.0.2; Swashbuckle 6.6.2	Npgsql provider; Nullable & ImplicitUsings
Gateway	ASP.NET Core (net8.0), YARP ReverseProxy	net8.0; YARP 2.3.0	Instradamento dinamico delle richieste
UserProfileService	ASP.NET Core (net8.0), EF Core, JWT	net8.0; EF Core 9.0.4; JwtBearer 8.0.4	Soft-delete; Gestione profilazione e consensi
CocktailImportService	ASP.NET Core (net8.0), EF Core, HttpClient, Newtonsoft.Json	net8.0; EF Core 9.0.4; Newtonsoft.Json 13.0.3	Sincronizzazione TheCocktailDB; Retry migrations
CocktailService	ASP.NET Core (net8.0), EF Core, Swagger	net8.0; EF Core 9.0.4; Swashbuckle 6.6.2	CRUD; Filtri (alcoholAllowed, is_verified)
CocktailSubmissionService	ASP.NET Core (net8.0), EF Core	net8.0; EF Core 9.0.4	Workflow di submission e approvazione admin
FavoriteCocktailsService	ASP.NET Core (net8.0), EF Core	net8.0; EF Core 9.0.4	Gestione cocktail preferiti
SearchService	ASP.NET Core (net8.0), EF Core	net8.0; EF Core 9.0.4	Endpoint ingredients (/ingredients, /ingredients-map)
SearchHistoryService	ASP.NET Core (net8.0), EF Core	net8.0; EF Core 9.0.4	Tracciamento ricerche; Consenso al profiling
ImageFetcherService	ASP.NET Core (net8.0), HttpClientFactory	net8.0; Microsoft.Extensions.Http 9.0.4	Integrazione Pexels, Unsplash, Pixabay via JWT
Database	PostgreSQL	15.x	UUID PK; Full-text search (tsvector)
Container Orchestration	Docker Compose	n/a	Include Prometheus & Grafana; env_file in docker-compose
Monitoring	Prometheus, Grafana	n/a	Raccolta metriche e dashboard in tempo reale

6. Requisiti Tecnici e Non Funzionali

- **Hardware**:
  - CPU quad-core (x86_64 o ARM64)
  - 8 GB RAM
  - 10 GB spazio su disco
- **Software**:
  - .NET 8.0 Runtime
  - Node.js 18.x
  - Docker 20.x (CLI & Engine)
  - PostgreSQL 15
- **Browser support**: Chrome, Firefox, Edge, Safari (ultime due versioni)

6.1 Requisiti minimi

- **TLS/HTTPS**: terminazione TLS gestita da NGINX (reverse-proxy esterno)
- **Autenticazione/Autorizzazione**: JWT con rotazione e scadenza dei token
- **CORS**: policy restrittiva definita in codice (commenti indicano come attivare la modalità "production")
- **Gestione credenziali**: variabili d'ambiente
- **Auditing & Logging**: tracciamento centralizzato di accessi e operazioni critiche

6.2 Sicurezza

- **Container & Orchestrazione**: microservizi containerizzati con Docker Compose
- **Bilanciamento del carico**: NGINX + YARP Reverse Proxy
- **Scalabilità orizzontale**: replica di servizi e read-replica PostgreSQL
- **Monitoraggio & Auto-scaling**: metriche Prometheus + dashboard Grafana

6.4 Architettura di Deploy

6.4.1 Deploy Minimo su Hetzner Cloud

- **Load Balancer**
  - Hetzner Cloud Load Balancer (HL1)
  - Terminazione TLS con Certbot (Let's Encrypt)
- **Server Applicativo**
  - VM **CX21** (2 vCPU, 8 GB RAM)
  - OS: Ubuntu 24.04 LTS
  - Storage: 40 GB SSD (root)
  - Docker & Docker Compose per tutti i microservizi + frontend
- **Server Database**
  - VM **CX31** (4 vCPU, 16 GB RAM)
  - OS: Ubuntu 24.04 LTS
  - Volume SSD 100 GB montato su /var/lib/postgresql/data
  - PostgreSQL 15 in modalità primary; snapshot giornalieri
- **Network & Sicurezza**
  - Floating IP assegnata al Load Balancer
  - Private Network tra VM per traffico interno
  - Hetzner Cloud Firewall:
    - SSH (22) solo da IP autorizzati
    - HTTP (80) e HTTPS (443) aperti
- **Monitoring & Logging**
  - Prometheus Node Exporter su ciascuna VM
  - Grafana containerizzato sul Server Applicativo
  - Servizio di logging con Loki + Promtail

7. Roadmap e Ruoli del Team

7.1 Roadmap sintetica

- **Giorni 1–10**: preparazione individuale delle tecnologie; Stefano studia approcci per il frontend, Nicol0' affina competenze in un altro gruppo.
- **Giorno 11**: formazione del team unico; definizione dell'architettura complessiva e assegnazione dei ruoli (Stefano→backend, Nicolo'→frontend).
- **Giorni 12–25** (Settimane 2–3): fase di studio e integrazione del primo microservizio e realizzazione del prototipo di homepage.
- **Giorni 26–40** (Settimane 4–5): sviluppo del modulo di ricerca cocktail e del pannello amministrativo (Admin Panel).
- **Giorni 41–55** (Settimane 6–7): implementazione della funzionalità "Preferiti", sincronizzazione periodica con TheCocktailDB e test end-to-end.
- **Giorno 56-fine**: rifinitura, ottimizzazioni di performance e preparazione della documentazione finale.

7.2 Ruoli e responsabilità

- **Stefano Montuori (Backend)**
  - Progettazione e sviluppo dei microservizi (.NET 8.0)
  - Configurazione e gestione di PostgreSQL (migrations, retry logic)
  - Sincronizzazione con TheCocktailDB e logica di profiling/statistiche
  - Orchestrazione container (Docker Compose) e setup monitoraggio (Prometheus, Grafana)
  - Automazione CI/CD e gestione delle variabili di ambiente
- **Nicolo' (Frontend)**
  - Sviluppo interfaccia in Angular (TypeScript, Reactive Forms, SCSS)
  - Configurazione progetto (angular.json, budgets, output hashing, polyfills)
  - Integrazione PrimeNG e Chart.js per visualizzazione dei dati
  - Implementazione del flusso di autenticazione JWT lato client
  - Realizzazione dell'Admin Panel e gestione della UX

8. Note Finali e Allegati

- https://github.com/orgs/GliScoppiati/repositories
- https://www.loom.com/share/0e37967075d548c08c31cc424fde356c?sid=293fc4da-16dc-41bf-a706-8427db7d1aa3