**Documentazione: Implementazione Backend Recensioni con Netlify & Supabase**

**Obiettivo:** Aggiungere funzionalità dinamiche (invio e visualizzazione di recensioni degli utenti) a un sito web statico preesistente (HTML, CSS, JavaScript), mantenendo i costi di infrastruttura al minimo possibile.

**Tecnologie Scelte:**

* **Git/GitHub:** Per il controllo di versione del codice e come trigger per il deployment automatico.
* **Netlify:** Piattaforma per l'hosting del sito statico e l'esecuzione di funzioni serverless (backend "leggero"). Scelta per il suo generoso piano gratuito e l'integrazione con Git.
* **Supabase:** Piattaforma Backend-as-a-Service (BaaS) open-source. Usata come database (PostgreSQL) per memorizzare le recensioni. Scelta per il suo piano gratuito e la facilità d'uso.
* **Node.js:** Utilizzato come ambiente di runtime per scrivere le funzioni serverless su Netlify (essendo basate su JavaScript).

**Fasi di Implementazione:**

**Fase 1: Prerequisiti e Setup Progetto Locale con Git**

1. **Installazione Strumenti:** Verifica/installazione degli strumenti di sviluppo necessari localmente: Node.js (con npm), Git, un editor di codice (es. VS Code).
2. **Organizzazione Progetto:** Raccolta di tutti i file del sito web statico (HTML, CSS, JS, immagini) in una cartella dedicata sul computer.
3. **Inizializzazione Git:** Apertura del terminale nella cartella del progetto ed esecuzione del comando git init per creare un repository Git locale.
4. **Primo Commit:** Aggiunta di tutti i file del progetto all'area di staging (git add .) e creazione del primo commit nella storia locale (git commit -m "Initial commit"), salvando lo stato iniziale del sito.
5. **Creazione Repository Remoto:** Creazione di un nuovo repository **vuoto** su GitHub (senza README, .gitignore o licenza iniziali).
6. **Collegamento Remoto:** Collegamento del repository Git locale al repository remoto appena creato su GitHub tramite il comando git remote add origin <URL\_REPOSITORY\_GITHUB>.
7. **Push Iniziale:** Caricamento del codice locale (primo commit) sul repository GitHub tramite il comando git push -u origin main.

**Fase 2: Configurazione Database (Supabase)**

1. **Creazione Account/Progetto:** Registrazione di un account gratuito su Supabase e creazione di un nuovo progetto, scegliendo una regione e impostando una password sicura per il database.
2. **Creazione Tabella reviews:** Utilizzo del "Table Editor" nella dashboard di Supabase per creare una nuova tabella chiamata reviews.
3. **Definizione Colonne:** Impostazione delle colonne necessarie per la tabella reviews:
   * id (int8, chiave primaria, auto-generata)
   * created\_at (timestamptz, default now(), auto-generata)
   * name (text)
   * rating (int2, per voti 1-5)
   * comment (text)
   * is\_approved (bool, default false)
4. **Sicurezza (RLS):** Disabilitazione temporanea della Row Level Security (RLS) sulla tabella reviews per semplificare l'accesso iniziale durante lo sviluppo.
5. **Recupero Credenziali API:** Individuazione e copia dell'URL del progetto e della chiave API anon public dalla sezione "Project Settings" > "API" di Supabase. Queste credenziali sono necessarie per permettere al codice di interagire con il database.

**Fase 3: Configurazione Hosting e Funzioni (Netlify)**

1. **Creazione Account/Sito:** Registrazione di un account gratuito su Netlify e creazione di un nuovo sito importando il repository GitHub creato nella Fase 1.
2. **Configurazione Build:** Verifica delle impostazioni di build (solitamente auto-detectate da Netlify per siti statici).
3. **Setup Locale Funzioni:** Creazione nel progetto locale della struttura di cartelle netlify/functions/ per ospitare il codice delle funzioni serverless.
4. **Configurazione netlify.toml:** Creazione di un file netlify.toml nella root del progetto per indicare a Netlify dove trovare le funzioni (functions = "netlify/functions") e altre impostazioni di build/deploy.
5. **Installazione Dipendenze:** Utilizzo di npm install @supabase/supabase-js nel terminale (nella root del progetto) per aggiungere la libreria client di Supabase, necessaria alle funzioni per comunicare con il database. Questo crea/aggiorna package.json e package-lock.json.
6. **Impostazione Variabili d'Ambiente:** Configurazione sicura delle credenziali Supabase all'interno delle impostazioni del sito su Netlify (Site configuration > Build & deploy > Environment variables):
   * Creazione variabile SUPABASE\_URL con l'URL del progetto Supabase.
   * Creazione variabile SUPABASE\_ANON\_KEY con la chiave API anon public di Supabase.

**Fase 4: Sviluppo Funzioni Serverless (Logica Backend)**

1. **Creazione submit-review.js:** Scrittura del codice Node.js per la funzione (file netlify/functions/submit-review.js). Questa funzione:
   * Verifica che la richiesta sia di tipo POST.
   * Legge i dati JSON (nome, rating, commento) inviati dal frontend.
   * Recupera le credenziali Supabase dalle variabili d'ambiente Netlify.
   * Esegue una validazione base dei dati ricevuti.
   * Utilizza il client Supabase per inserire una nuova riga nella tabella reviews, impostando is\_approved su false.
   * Gestisce eventuali errori di inserimento.
   * Restituisce una risposta JSON al frontend (successo o errore).
2. **Creazione get-reviews.js:** Scrittura del codice Node.js per la funzione (file netlify/functions/get-reviews.js). Questa funzione:
   * Verifica che la richiesta sia di tipo GET.
   * Recupera le credenziali Supabase dalle variabili d'ambiente.
   * Utilizza il client Supabase per interrogare la tabella reviews.
   * Seleziona solo le colonne necessarie (name, rating, comment, created\_at) delle recensioni dove is\_approved è true.
   * Ordina i risultati (es. dalle più recenti).
   * Gestisce eventuali errori di query.
   * Restituisce l'array di recensioni trovate (o un array vuoto) come JSON al frontend.

**Fase 5: Integrazione Frontend (HTML & JavaScript)**

1. **Modifica contenuti.html:**
   * Aggiunta della sezione "Recensioni" con titoli, paragrafi introduttivi.
   * Inclusione di un'area (<div class="testimonial-gallery">) per mostrare immagini statiche di testimonianze.
   * Inclusione di un contenitore vuoto (<div id="reviews-list-container">) dove verranno caricate dinamicamente le recensioni approvate.
   * Inclusione di un modulo HTML (<form id="review-form">) con campi per nome (id="reviewName"), rating (id="reviewRating" usando un <select>), commento (id="reviewComment" usando <textarea>), un bottone (<button type="submit">) e un'area messaggi (<div id="review-form-message">). **Nota:** L'attributo action del form viene omesso perché l'invio è gestito via JavaScript.
2. **Modifica script.js:**
   * Aggiunta funzione helper displayStars(rating) per generare l'HTML delle stelle.
   * Aggiunta funzione asincrona loadReviews() per:
     + Effettuare una chiamata Workspace GET alla funzione /api/get-reviews (o /.netlify/functions/get-reviews).
     + Elaborare la risposta JSON.
     + Creare dinamicamente gli elementi HTML per ciascuna recensione approvata e inserirli in #reviews-list-container.
     + Gestire stati di caricamento ed errori.
   * Aggiunta funzione asincrona handleReviewSubmit(event) per:
     + Prevenire l'invio di default del form.
     + Recuperare e validare i dati inseriti nel form.
     + Mostrare un messaggio di "Invio in corso..." all'utente.
     + Effettuare una chiamata Workspace POST alla funzione /api/submit-review (o /.netlify/functions/submit-review), inviando i dati della recensione nel body come JSON.
     + Gestire la risposta (successo/errore) aggiornando l'area messaggi e resettando il form.
   * **Modifica del listener DOMContentLoaded:** Aggiunta del codice per:
     + Associare la funzione handleReviewSubmit all'evento submit del form #review-form (addEventListener).
     + Chiamare loadReviews() al caricamento della pagina contenuti.html per visualizzare le recensioni esistenti.
   * **Aggiornamento Traduzioni:** Aggiunta delle chiavi e delle traduzioni per tutti i nuovi testi e label della sezione recensioni nell'oggetto languages.

**Fase 6: Deployment Finale e Moderazione**

1. **Commit & Push Finali:** Salvataggio di tutte le modifiche (HTML, JS frontend, JS funzioni backend, netlify.toml, package.json, package-lock.json) in Git e caricamento su GitHub (git add ., git commit, git push).
2. **Build & Deploy Netlify:** Verifica del completamento automatico del processo di build e deploy su Netlify, controllando i log per eventuali errori.
3. **Test Funzionale:** Navigazione sul sito live Netlify, test approfondito della visualizzazione delle recensioni (inizialmente vuota) e del processo di invio di una nuova recensione, monitorando la console del browser e i log delle funzioni Netlify per eventuali problemi.
4. **Moderazione Iniziale:** Accesso alla dashboard di Supabase, navigazione alla tabella reviews e modifica manuale della colonna is\_approved da false a true per le recensioni che si desidera pubblicare.
5. **Verifica Finale:** Ricaricamento della pagina dei contenuti sul sito live per confermare che le recensioni approvate vengano ora visualizzate correttamente.

**(Fase Opzionale) Pagamenti:** Per i pagamenti si è deciso di **non** implementare un backend dedicato in questa fase. Le opzioni sono:

* **Bonifico Bancario:** Gestito manualmente tramite comunicazione privata dell'IBAN.
* **PayPal (Semplice):** Utilizzo di un link PayPal.Me fornito da Maria, da inserire sul sito o comunicare manualmente.
* Entrambi i metodi non richiedono modifiche all'infrastruttura backend/serverless creata per le recensioni. È stata aggiornata solo la nota informativa nella pagina "Piani".

Questa documentazione riassume il flusso di lavoro seguito per trasformare un sito statico aggiungendo funzionalità dinamiche tramite un approccio serverless e BaaS mantenuto all'interno dei piani gratuiti dei servizi scelti.