Esercizio Tecnico: Secret Santa Generator 🎅

Panoramica

Devi creare un'applicazione web per organizzare eventi "Secret Santa" (Babbo Natale Segreto).
L'app permette di creare gruppi, invitare partecipanti, raccogliere wishlist e assegnare casualmente chi regala a chi.

Stack richiesto:

- Laravel 12
- React + Inertia.js
- Tailwind CSS
- HeroUI (libreria componenti React)
- PostgreSQL

Requisiti

1. Setup Progetto

- [ ] Crea un nuovo repository GitHub pubblico chiamato `secret-santa-app`
- [ ] Inizializza il progetto usando lo starter kit ufficiale Laravel con React
- [ ] Installa e configura HeroUI (libreria di componenti UI per React)
- [ ] Configura PostgreSQL nel file `.env`
- [ ] Fai commit frequenti e significativi seguendo il formato Conventional Commits

```
# Esempi di commit validi
feat(events): aggiunta creazione evento con data e budget
feat(participants): implementato invito partecipanti
fix(extraction): corretto algoritmo per evitare auto-assegnazione
```

---

2. Funzionalità Richieste

2.1 Gestione Eventi

Un utente (organizzatore) può:

- Creare un nuovo evento Secret Santa con:
  - Nome evento (es. "Natale Ufficio 2025")
  - Data dello scambio regali
  - Budget massimo per regalo (€)
- Visualizzare la lista dei propri eventi
- Eliminare un evento (solo se non è stata fatta l'estrazione)

2.2 Gestione Partecipanti

I partecipanti sono **utenti già registrati** all'applicazione.

Alla creazione di un evento:

- Tutti gli utenti registrati (incluso l'organizzatore) vengono invitati automaticamente
- Ogni partecipante riceve una **email di invito** (usa Mailpit per testare in locale)
- Ogni partecipante deve **accettare o rifiutare** l'invito

L'organizzatore può:

- Visualizzare la lista dei partecipanti con il loro stato (in attesa, accettato, rifiutato)
- Rimuovere manualmente un partecipante (solo prima dell'estrazione)

2.3 Wishlist

La wishlist è composta da **card separate**, una per ogni oggetto desiderato.

Ogni partecipante (utente autenticato) può:

- Aggiungere oggetti alla propria wishlist (ogni oggetto = una card)
- Modificare o rimuovere singoli oggetti
- Vedere i dettagli dell'evento (nome, data, budget)
- Vedere a quali eventi partecipa

Ogni card oggetto contiene:

- Nome dell'oggetto (obbligatorio)
- Descrizione o link (opzionale)

2.4 Estrazione

L'organizzatore può:

- Se ci sono partecipanti "in attesa", deve segnarli manualmente come accettati o rifiutati
- **Non può avviare l'estrazione finché ci sono partecipanti "in attesa"**
- Avviare l'estrazione quando tutti hanno risposto e ci sono almeno 3 partecipanti accettati
- L'algoritmo deve assegnare casualmente chi regala a chi, rispettando:
  - Nessuno può essere assegnato a se stesso
  - Ogni persona regala a una sola persona
  - Ogni persona riceve da una sola persona

2.5 Visualizzazione Assegnazione

Dopo l'estrazione:

- Ogni partecipante può vedere (dalla propria area):
  - A chi deve fare il regalo
  - La wishlist della persona assegnata
- L'organizzatore **non vede chi regala a chi** (il segreto vale per tutti!)
- L'organizzatore può vedere solo:
  - Chi ha già visualizzato la propria assegnazione
  - Per vedere a chi deve fare il regalo, accede alla sua area partecipante come tutti gli altri

---

3. Requisiti Tecnici

3.1 Database

Progetta lo schema con almeno queste tabelle:

- `users` (organizzatori e partecipanti)
- `events`
- `participants` (pivot user-event)
- `wishlist_items` (oggetti della wishlist per partecipante)
- `assignments` (risultato estrazione)

3.2 Validazione

Implementa validazione appropriata:

- Nome evento: obbligatorio, max 255 caratteri
- Data scambio regali: obbligatoria, deve essere futura
- Budget: obbligatorio, numerico, minimo 1
- Oggetto wishlist:
  - Nome: obbligatorio, max 255 caratteri
  - Descrizione/link: opzionale, max 500 caratteri

3.3 Struttura Codice

- Usa Form Request per la validazione
- Usa Resource Controller dove appropriato
- Mantieni i controller snelli (logica nei Model o Service)
- Crea Factory per ogni Model (per popolare il database con dati di test)

3.4 Interfaccia Utente

Utilizza i componenti HeroUI per costruire l'interfaccia.

Requisiti UI minimi:

- Form con componenti `Input`, `Button`
- Liste/tabelle con `Table` o `Card`
- Feedback visivo con `Chip` per gli stati
- Modal per conferme (es. conferma estrazione)

---

4. Extra (obbligatorio almeno 1)

Scegli e implementa **almeno 1** tra le seguenti funzionalità:

- ☐ **Traduzioni Laravel + Matice**: usa il sistema di localizzazione di Laravel per tutti i testi dell'interfaccia,
  sia backend che frontend. Usa Matice per condividere le traduzioni con React
- ☐ **Esclusioni**: possibilità di specificare "Mario non può essere assegnato a Luigi"
- ☐ **Dark mode**: supporto tema scuro

Criteri di Valutazione

Area	Peso	Cosa guardiamo
Funzionalità	40%	L'app funziona? Copre i requisiti?
Qualità codice	20%	Leggibilità, naming, struttura
Laravel	15%	Uso corretto di Eloquent, validation, struttura
React/Inertia	15%	Componenti puliti, gestione stato
Git	10%	Commit atomici, messaggi chiari

Consegna

1. Assicurati che il repository GitHub sia **pubblico**
2. Includi un `README.md` con:
    - Istruzioni per installare e avviare il progetto
    - Eventuali scelte tecniche che vuoi motivare
3. Invia il link al repository entro la scadenza comunicata

Risorse Utili

- Laravel 12 Starter Kits
- Laravel Localization
- Matice - Laravel translations for JS
- HeroUI Documentation
- HeroUI Components

Buon lavoro e buon divertimento! 🎄