📄 Mondodoro – Documentazione Tecnica MVP

1. Obiettivo
Creare una piattaforma per **liste regalo e collette online** dedicate a gioiellerie. Gli invitati possono contribuire facilmente con carta di credito (Stripe), mentre i gioiellieri hanno una **dashboard per gestire pagamenti e liste**.

---

2. Architettura del sistema
- **Frontend**: Next.js + Tailwind CSS (UI responsive, SEO friendly)
- **Backend**: Django REST Framework (API REST + gestione utenti)
- **Database**: PostgreSQL
- **Pagamenti**: Stripe (Checkout + Connect)
- **Containerizzazione**: Docker + Docker Compose
- **CI/CD & Versioning**: Git + GitHub/GitLab
- **Hosting**:
  - Frontend: Vercel (oppure containerizzato su Render/Heroku/Docker server)
  - Backend: Render/Railway/Heroku con supporto Docker
  - Database: PostgreSQL managed (Render, Supabase, Railway)

---

3. Funzionalità MVP
- **SuperAdmin (Mondodoro)**: gestisce gioiellerie e fee
- **Gioielliere**:
  - Creazione lista regalo/colletta
  - Dashboard con contributi ricevuti
  - Collegamento Stripe Connect account
- **Invitato**:
  - Visualizza lista via link pubblico
  - Paga tramite Stripe Checkout

---

4. Flusso principale
1. Gioielliere crea lista regalo dal pannello
2. Ottiene link univoco da condividere
3. Invitato apre link → vede target e prodotti
4. Invitato paga con Stripe Checkout
5. Stripe gestisce pagamenti (fee Mondodoro + accredito gioielliere)
6. Webhook aggiorna dashboard gioielliere

---

5. Stack tecnologico

Componente	Tecnologia
Frontend	Next.js (React) + Tailwind CSS
Backend	Django REST Framework / FastAPI
Database	PostgreSQL
Pagamenti	Stripe (Checkout + Connect)
Auth	JWT (Django SimpleJWT)
Container	Docker + Docker Compose
Versioning	GitHub/GitLab

---

6. Struttura repo Git (monorepo)

```
mondodoro/
├── frontend/          # Next.js + Tailwind
│   ├── Dockerfile
│   └── ...
│
├── backend/           # Django REST API
│   ├── Dockerfile
│   └── ...
│
├── docker-compose.yml
├── README.md
```

---

7. Docker Setup

Esempio `docker-compose.yml`

```yaml
version: '3.9'

services:
  backend:
    build: ./backend
    container_name: mondodoro_backend
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - ./backend:/app
    ports:
      - "8000:8000"
```

```
    env_file:
      - ./backend/.env
    depends_on:
      - db

  frontend:
    build: ./frontend
    container_name: mondodoro_frontend
    command: npm run dev
    volumes:
      - ./frontend:/app
    ports:
      - "3000:3000"
    depends_on:
      - backend

  db:
    image: postgres:14
    container_name: mondodoro_db
    restart: always
    environment:
      POSTGRES_USER: mondodoro
      POSTGRES_PASSWORD: secret
      POSTGRES_DB: mondodoro_db
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```

Esempio `backend/Dockerfile`
```dockerfile
FROM python:3.11

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8000
```

CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

### Esempio `frontend/Dockerfile`
```dockerfile
FROM node:18

WORKDIR /app

COPY package.json package-lock.json ./
RUN npm install

COPY . .

EXPOSE 3000
CMD ["npm", "run", "dev"]
```

---

8. Workflow con Git
- **Main branch** = stabile (deploy in produzione)
- **Dev branch** = sviluppo
- **Feature branches** = nuove funzioni
- **Pull Request + Code Review** prima di mergiare in `main`

Setup iniziale
```bash
git clone https://github.com/tuo-utente/mondodoro.git
cd mondodoro
docker-compose up --build
```

---

9. Sicurezza
- HTTPS (gestito da hosting / Nginx reverse proxy)
- Stripe gestisce i dati sensibili → niente PCI compliance da gestire
- Variabili sensibili in `.env` (non committate su Git)

---

10. Roadmap sviluppo
- **Sprint 1**: Setup repo, auth gioielliere, creazione liste regalo
- **Sprint 2**: Stripe Checkout + Connect, webhook pagamenti
```

- **Sprint 3**: Dashboard frontend, email notifiche
- **Sprint 4**: Deploy con Docker + hosting cloud