Qamarero

Sistema de gestión de pagos para restaurantes que permite dividir cuentas de múltiples formas.
Desarrollado con Better-T-Stack, un stack moderno de TypeScript.

🚀 Características

- **TypeScript** - Seguridad de tipos y mejor experiencia de desarrollo
- **Next.js** - Framework React full-stack
- **TailwindCSS** - CSS utility-first para desarrollo rápido de UI
- **shadcn/ui** - Componentes UI reutilizables
- **tRPC** - APIs con tipos end-to-end
- **Drizzle** - ORM orientado a TypeScript
- **PostgreSQL** - Motor de base de datos
- **Turborepo** - Sistema de build optimizado para monorepos
- **Husky** - Git hooks para calidad de código
- **Biome** - Linting y formateo

📋 Requisitos Previos

Antes de comenzar, asegúrate de tener instalado:

- **Node.js** (v18 o superior)
- **pnpm** (v10 o superior) - Gestor de paquetes
- **Docker** y **Docker Compose** - Para la base de datos PostgreSQL

Instalar pnpm

Si no tienes pnpm instalado, puedes instalarlo globalmente con:

```
npm install -g pnpm
```

Verificar Docker

Asegúrate de que Docker esté corriendo en tu sistema:

```
docker --version
docker compose version
```

🛠️ Instalación y Configuración

Paso 1: Clonar e Instalar Dependencias

```

```

```
# Clonar el repositorio (si aplica)
git clone <url-del-repositorio>
cd qamarero

# Instalar todas las dependencias del proyecto
pnpm install
```

Paso 2: Configurar Variables de Entorno

Crea un archivo `.env` en la carpeta `apps/web/`:

```
cd apps/web
touch .env
```

Edita el archivo `.env` y agrega la siguiente configuración:

```
DATABASE_URL=postgresql://postgres:password@localhost:5432/qamarero
PORT=3001
BETTER_AUTH_SECRET=A6NmvoerEpMiBgET9lExeheQrru3Uecq
BETTER_AUTH_URL=http://localhost:3001
CORS_ORIGIN=http://localhost:3001
```

Explicación:

- `DATABASE_URL`: URL de conexión a PostgreSQL
  - Usuario: `postgres`
  - Contraseña: `password`
  - Host: `localhost`
  - Puerto: `5432`
  - Base de datos: `qamarero`

> **Nota:** Estos valores coinciden con la configuración del docker-compose.yml. Si cambias los valores en docker-compose, actualiza también esta URL.

Paso 3: Iniciar la Base de Datos con Docker

El proyecto incluye un archivo `docker-compose.yml` que configura PostgreSQL automáticamente.

```
# Desde la raíz del proyecto
cd packages/db

# Iniciar PostgreSQL en modo detached (en segundo plano)
docker compose up -d

# O usar el script del monorepo (desde la raíz)
```

```
cd ../../..
pnpm run db:start
```

Verificar que la base de datos esté corriendo:

```
# Ver los contenedores activos
docker ps

# Deberías ver un contenedor llamado "qamarero-postgres"
```

Paso 4: Aplicar el Esquema de la Base de Datos

Una vez que PostgreSQL esté corriendo, necesitas crear las tablas:

```
# Desde la raíz del proyecto
pnpm run db:push
```

Este comando creará todas las tablas necesarias (`restaurant_table`, `order`, `order_item`) según el esquema definido en `packages/db/src/schema/`.

Paso 5: Poblar la Base de Datos con Datos de Prueba

Si quieres tener datos de ejemplo para probar la aplicación:

```
pnpm run db:seed
```

Esto insertará mesas, pedidos e items de ejemplo.

Paso 6: Iniciar el Servidor de Desarrollo

```
# Desde la raíz del proyecto
pnpm run dev
```

Este comando iniciará:

- El servidor Next.js en `http://localhost:3001`
- Drizzle Studio (interfaz visual para la base de datos) en `http://localhost:4983`

Abrir en el navegador:

- **Aplicación:** http://localhost:3001
- **Drizzle Studio:** http://localhost:4983

📁 Estructura del Proyecto

```
qamarero/
├── apps/
│   └── web/              # Aplicación full-stack (Next.js)
│       └── src/
│           ├── app/      # Rutas y páginas
│           └── components/ # Componentes React
├── packages/
│   ├── api/              # Capa de API / Lógica de negocio (tRPC)
│   │   └── src/routers/  # Routers de tRPC
│   └── db/               # Esquema de base de datos y queries
│       ├── docker-compose.yml # Configuración de PostgreSQL
│       └── src/
│           ├── schema/   # Esquema de Drizzle ORM
│           └── scripts/  # Scripts de seed y utilidades
```

📝 Comandos Disponibles

Desarrollo

- `pnpm run dev` - Inicia todas las aplicaciones en modo desarrollo (Next.js + Drizzle Studio)
- `pnpm run dev:web` - Solo inicia la aplicación web

Base de Datos

- `pnpm run db:start` - Inicia PostgreSQL con Docker (modo detached)
- `pnpm run db:stop` - Detiene PostgreSQL (mantiene los datos)
- `pnpm run db:down` - Detiene y elimina el contenedor (⚠ puede perder datos si no están en un volumen)
- `pnpm run db:push` - Aplica el esquema a la base de datos (crea/actualiza tablas)
- `pnpm run db:studio` - Abre Drizzle Studio (interfaz visual de la BD)
- `pnpm run db:generate` - Genera migraciones
- `pnpm run db:migrate` - Aplica migraciones
- `pnpm run db:seed` - Pobla la base de datos con datos de prueba

Calidad de Código

- `pnpm run check` - Ejecuta Biome (linting y formateo) y corrige problemas automáticamente
- `pnpm run check-types` - Verifica tipos TypeScript en todos los paquetes

Build

- `pnpm run build` - Construye todas las aplicaciones para producción

🐋 Gestión de Docker

Comandos Útiles de Docker

```
# Ver logs de PostgreSQL
docker logs qamarero-postgres

# Ver logs en tiempo real
docker logs -f qamarero-postgres

# Acceder a PostgreSQL directamente
docker exec -it qamarero-postgres psql -U postgres -d qamarero

# Detener todos los contenedores
docker compose -f packages/db/docker-compose.yml down

# Eliminar volúmenes (⚠ esto borra todos los datos)
docker compose -f packages/db/docker-compose.yml down -v
```

🔧 Solución de Problemas

El puerto 5432 ya está en uso

Si ya tienes PostgreSQL corriendo localmente en el puerto 5432, tienes dos opciones:

1. **Detener tu instancia local de PostgreSQL** (recomendado)
2. **Cambiar el puerto en docker-compose.yml:**

   ```
   ports:
     - "5433:5432"  # Cambia 5432 a 5433
   ```

   Y actualiza `DATABASE_URL` en `.env`:

   ```
   DATABASE_URL=postgresql://postgres:password@localhost:5433/qamarero
   ```

Error: "Cannot find module"

Asegúrate de haber ejecutado `pnpm install` desde la raíz del proyecto.

La base de datos no se conecta

1. Verifica que Docker esté corriendo: `docker ps`
2. Verifica que el contenedor esté activo: `docker ps | grep qamarero-postgres`
3. Verifica la variable `DATABASE_URL` en `apps/web/.env`
4. Revisa los logs: `docker logs qamarero-postgres`

Error al ejecutar `db:push`

Asegúrate de que:

1. El contenedor de PostgreSQL esté corriendo
2. El archivo `.env` exista en `apps/web/` con `DATABASE_URL` configurado
3. Los valores de usuario/contraseña coincidan con docker-compose.yml

📚 Documentación Adicional

- Para información sobre decisiones de diseño y futuras iteraciones, consulta ITERACIONES.md

🎯 Modos de Pago Disponibles

La aplicación soporta tres modos de división de pagos:

1. **Pagar Todo** - Pago único de toda la cuenta
2. **Dividir en Partes Iguales** - División automática del total entre grupos
3. **Personalizar** - Asignación manual de items específicos a grupos

🛑 Detener la Aplicación

Para detener todos los servicios:

```
# Detener el servidor de desarrollo (Ctrl+C en la terminal)
# Detener PostgreSQL
pnpm run db:stop
```