

Guía Completa: Configurar Laravel Sail en Windows 11

Con Problemas Reales y Soluciones Probadas (Versión Actualizada)

Autor: Basado en la experiencia real de configuración en Windows 11

Fecha: Junio 2025

Objetivo: Levantar un proyecto Laravel con Docker Sail desde GitHub

Tabla de Contenidos

1. [Requisitos Previos](#)
2. [Instalación de Herramientas Base](#)
3. [Configuración de WSL](#)
4. [Instalación de Docker Desktop](#)
5. [Clonado y Configuración del Proyecto](#)
6. [Problemas Encontrados y Soluciones](#)
7. [Configuración del Dominio Personalizado](#)
8. [Configuración del Alias de Sail](#)
9. [Configuración del Gestor de Base de Datos](#)
10. [Comandos Útiles](#)
11. [Conclusiones](#)

Requisitos Previos

- **Windows 11** (probado en esta versión)
- **Conexión a internet estable**
- **Permisos de administrador**
- **Al menos 8GB de RAM** (recomendado)
- **20GB de espacio libre** en disco

Instalación de Herramientas Base

1. Instalar WSL (Windows Subsystem for Linux)

¿Por qué WSL? Laravel Sail está optimizado para entornos Unix/Linux.

```
powershell  
# Ejecutar en PowerShell como Administrador  
wsl --install
```

 **PROBLEMA ENCONTRADO:** Despues de instalar WSL, Windows requiere reinicio.

 **SOLUCIÓN:**

1. Reiniciar la computadora obligatoriamente
2. Al reiniciar, se abrirá una terminal automáticamente para configurar Ubuntu
3. Crear usuario y contraseña cuando se solicite

2. Verificar Instalación de WSL

```
bash  
  
# Verificar versión de WSL  
wsl --version  
  
# Verificar distribuciones instaladas  
wsl --list --verbose
```

Instalación de Docker Desktop

1. Descargar Docker Desktop

- Ir a: <https://www.docker.com/products/docker-desktop/>
- Descargar la versión para Windows
- **Importante:** Asegurarse de que incluya soporte para WSL 2

2. Instalación

1. Ejecutar el instalador como administrador
2.  **MARCAR:** "Use WSL 2 instead of Hyper-V" durante la instalación
3. Completar instalación y reiniciar si es necesario

3. Configuración Post-Instalación

 **PROBLEMA ENCONTRADO:** Docker Desktop no se integra automáticamente con WSL.

 **SOLUCIÓN:**

1. Abrir Docker Desktop
2. Ir a **Settings → Resources → WSL Integration**
3. **Activar:** "Enable integration with my default WSL distro"
4. **Activar:** La distribución Ubuntu específica
5. Hacer clic en "**Apply & Restart**"

4. Verificar Funcionamiento

```
bash

# Desde terminal WSL
docker --version
docker-compose --version

# Probar Docker
docker run hello-world
```

Clonado y Configuración del Proyecto

1. Configurar Git (si no está configurado)

```
bash

git config --global user.name "Tu Nombre"
git config --global user.email "tu.email@ejemplo.com"
```

2. Crear Estructura de Directorios

```
bash

# Crear directorio para proyectos
mkdir ~/proyectos
cd ~/proyectos
```

3. Clonar el Proyecto

```
bash

# Clonar proyecto desde GitHub
git clone https://github.com/usuario/laravel_avanzado.git
cd laravel_avanzado
```

4. Verificar Contenido del Proyecto

bash

```
# Verificar que existe el archivo sail  
ls -la vendor/bin/sail  
  
# Si no existe vendor/, instalar dependencias primero  
composer install
```

Problemas Encontrados y Soluciones

Problema 1: Comando `sail` no encontrado

 Error:

bash

```
$ sail ps  
Command 'sail' not found, but can be installed with:  
sudo apt install bsdgames
```

 **SOLUCIÓN:** Sail no es un comando global, debe ejecutarse desde el directorio del proyecto:

bash

```
# Forma correcta  
./vendor/bin/sail ps  
  
# Crear alias (ver sección específica más adelante)
```

Problema 2: Conflicto de Puertos con Apache

 Error: Al acceder a `localhost` se muestra la página de Apache en lugar de Laravel.

 **DIAGNÓSTICO:**

bash

```
# Ver puertos mapeados  
./vendor/bin/sail ps
```

 **SOLUCIÓN:**

1. Identificar el puerto correcto de Laravel en la salida de `sail ps`

2. Si Apache está usando el puerto 80, detenerlo:

bash

```
sudo service apache2 stop
```

3. Acceder a Laravel en el puerto correcto (ej: `localhost:8080`)

Problema 3: Puerto 8080 no responde

✗ **Error:** `localhost:8080` muestra "This site can't be reached"

🔍 **DIAGNÓSTICO:**

bash

```
# Verificar logs del contenedor  
./vendor/bin/sail logs laravel.test
```

✓ **SOLUCIÓN:** Los logs mostraron que Laravel estaba sirviendo en puerto 80 interno, mapeado al 80 externo:

1. Detener Apache local: `sudo service apache2 stop`

2. Acceder a: `http://localhost` (puerto 80)

Problema 4: Dominio Personalizado

✗ **Error:** `laravel_avanzado.test` no funciona, solo `localhost`

🔍 **CAUSA:** El comando PowerShell agregó espacios entre caracteres:

bash

```
echo "127.0.0.1 laravel_avanzado.test" >> C:\Windows\System32\drivers\etc\hosts
```

✓ **SOLUCIÓN:**

1. Abrir como administrador:

powershell

```
notepad C:\Windows\System32\drivers\etc\hosts
```

2. Agregar manualmente al final:

127.0.0.1 laravel_avanzado.test

3. Guardar y cerrar
4. Limpiar caché DNS:

powershell

ipconfig /flushdns

Problema 5: Alias de Sail - PATH Corrupto

 **Error:** Despues de crear alias y ejecutar `(source ~/ .bashrc)`, todos los comandos básicos fallan:

bash

```
$ grep
Command 'grep' is available in the following places
 * /bin/grep
 * /usr/bin/grep
The command could not be located because '/usr/bin:/bin' is not included in the PATH environment
grep: command not found
```

 **CAUSA:** En `.bashrc` había una línea que sobrescribía completamente el PATH:

bash

```
export PATH="/home/r10/.config/composer/vendor/bin/"
```

SOLUCIÓN:

1. Editar `.bashrc`:

bash

nano `~/ .bashrc`

2. Cambiar la línea problemática de:

bash

```
export PATH="/home/r10/.config/composer/vendor/bin/"
```

A:

```
bash
```

```
export PATH="/home/r10/.config/composer/vendor/bin:$PATH"
```

3. **Importante!** El `:$PATH` al final agrega la ruta sin sobrescribir el PATH existente

4. Guardar y recargar:

```
bash
```

```
source ~/.bashrc
```

🌐 Configuración del Dominio Personalizado

1. Modificar Archivo Hosts

En Windows (como Administrador):

```
powershell
```

```
notepad C:\Windows\System32\drivers\etc\hosts
```

Agregar al final:

```
127.0.0.1 laravel_avanzado.test
```

2. Actualizar Laravel .env

```
bash
```

```
# Editar archivo .env
```

```
nano .env
```

```
# Cambiar La Línea APP_URL
```

```
APP_URL=http://laravel_avanzado.test
```

3. Reiniciar Sail

```
bash
```

```
./vendor/bin/sail down
```

```
./vendor/bin/sail up
```

4. Acceder al Proyecto

Abrir navegador en: http://laravel_avanzado.test

⚡ Configuración del Alias de Sail

¿Por qué necesitas un alias?

En lugar de escribir `./vendor/bin/sail` cada vez, podrás usar simplemente `sail`.

Paso 1: Verificar el Estado Actual

```
bash

# Verificar si hay aliases conflictivos
grep -n "sail" ~/.bashrc

# Ver el PATH actual
echo $PATH
```

Paso 2: Limpiar Configuraciones Anteriores (si las hay)

```
bash

# Eliminar Líneas previas de sail si existen
sed -i '/sail/d' ~/.bashrc
```

Paso 3: Verificar/Arreglar la Configuración del PATH

⚠ CRÍTICO: Antes de agregar el alias, asegurarse de que el PATH esté correctamente configurado.

```
bash

# Revisar el contenido de .bashrc
cat ~/.bashrc | grep "export PATH"
```

Si encuentras una línea como:

```
bash

export PATH="/home/usuario/.config/composer/vendor/bin/"
```

Cámbiala por:

```
bash
```

```
export PATH="/home/usuario/.config/composer/vendor/bin:$PATH"
```

Paso 4: Agregar el Alias

```
bash
```

```
# Agregar alias con ruta absoluta
echo "alias sail='/home/r10/proyectos/laravel_avanzado/vendor/bin/sail'" >> ~/.bashrc
```

Nota: Cambia `/home/r10/` por tu ruta de usuario real.

Paso 5: Recargar Configuración

```
bash
```

```
source ~/.bashrc
```

Paso 6: Verificar Funcionamiento

```
bash
```

```
# Verificar que el alias existe
alias | grep sail
```

```
# Verificar que los comandos básicos siguen funcionando
grep --version
ls --version
```

```
# Probar el alias desde cualquier directorio
cd ~
sail ps
```

Resultado Esperado

Deberías poder ejecutar:

```
bash
```

```
sail up
sail down
sail ps
sail artisan migrate
```

Desde **cualquier directorio** sin problemas.

Configuración del Gestor de Base de Datos

Recomendación: DBeaver Community

¿Por qué DBeaver?

- Gratuito y open source
- Multiplataforma
- Soporta MySQL, PostgreSQL, SQLite, etc.
- Interfaz moderna y limpia

Instalación

1. Descargar desde: <https://dbeaver.io/>
2. Instalar normalmente en Windows

Configuración de Conexión

Datos de conexión:

- **Host:** `localhost`
- **Puerto:** `3306`
- **Database:** `laravel_avanzado` (o el nombre en tu .env)
- **Usuario:** `sail`
- **Contraseña:** `password`

Problema Común: "Public Key Retrieval is not allowed"

SOLUCIÓN:

1. En la configuración de conexión, ir a "**Driver properties**"
2. Agregar estas propiedades:

```
allowPublicKeyRetrieval = true  
useSSL = false
```

Verificación

Una vez conectado, deberías ver las tablas de Laravel:

- `users`
 - `migrations`
 - `cache`
 - `jobs`
 - `sessions`
 - Y más...
-

Comandos Útiles

Comandos Docker/Sail Básicos

`bash`

Iniciar entorno

`sail up`

Iniciar en background

`sail up -d`

Parar entorno

`sail down`

Ver estado de contenedores

`sail ps`

Ver Logs

`sail logs`

Acceder al contenedor

`sail shell`

Comandos Laravel

```
bash

# Migrar base de datos
sail artisan migrate

# Crear controlador
sail artisan make:controller MiController

# Crear modelo
sail artisan make:model MiModelo

# Limpiar caché
sail artisan cache:clear

# Generar clave de aplicación
sail artisan key:generate
```

Comandos de Debugging

```
bash

# Ver configuración Docker
docker ps

# Ver logs específicos
sail logs laravel.test

# Verificar puertos en uso
netstat -tulpn | grep :80

# Verificar PATH actual
echo $PATH

# Verificar aliases
alias | grep sail
```

⚡ Flujo de Trabajo Diario

Al Iniciar el Día

```
bash

# 1. Abrir WSL
wsl

# 2. Ir al proyecto (o usar el alias desde cualquier Lado)
cd ~/proyectos/laravel_avanzado

# 3. Iniciar entorno
sail up -d

# 4. Verificar que funciona
curl http://laravel_avanzado.test
```

Al Terminar el Día

```
bash

# Parar entorno para Liberar recursos
sail down
```

🎯 Consejos Importantes

✓ Buenas Prácticas

1. **Siempre usar WSL** para comandos de desarrollo
2. **Verificar integración Docker-WSL** antes de empezar
3. **Cuidar la configuración del PATH** en `.bashrc`
4. **Detener Apache local** si causa conflictos de puerto
5. **Editar hosts manualmente** en lugar de usar comandos PowerShell
6. **Usar alias con ruta absoluta** para evitar problemas de directorio

⚠️ Errores Comunes a Evitar

1. **No reiniciar** después de instalar WSL
2. **Ejecutar Docker** desde PowerShell en lugar de WSL
3. **Sobrescribir el PATH** en lugar de agregarlo (usar `:$PATH`)
4. **Olvídar detener Apache** local
5. **Usar comandos PowerShell** para editar archivos críticos del sistema

6. No verificar el PATH antes de crear aliases

Troubleshooting Rápido

Problema	Verificación	Solución
<code>sail</code> no funciona	<code>ls vendor/bin/sail</code>	Usar <code>./vendor/bin/sail</code> o crear alias
Puerto ocupado	<code>sail ps</code>	Detener Apache: <code>sudo service apache2 stop</code>
Docker no responde	<code>docker ps</code>	Verificar integración WSL en Docker Desktop
Dominio no funciona	<code>ping laravel_avanzado.test</code>	Editar hosts manualmente
Comandos básicos fallan	<code>echo \$PATH</code>	Arreglar PATH en <code>.bashrc</code>
Alias no funciona	<code>alias grep sail</code>	Verificar ruta absoluta en alias
Contenedor no inicia	<code>sail logs</code>	Revisar logs para errores específicos

Resultado Final

Después de seguir esta guía tendrás:

- **WSL 2** funcionando con Ubuntu
- **Docker Desktop** integrado con WSL
- **Laravel Sail** levantando tu proyecto
- **Dominio personalizado** (`laravel_avanzado.test`)
- **Alias de sail** funcionando desde cualquier directorio
- **Base de datos MySQL** en contenedor
- **DBeaver** conectado para gestionar la BD
- **Entorno de desarrollo** completamente funcional

Conclusiones

Lo Bueno:

- Una vez configurado, el entorno es estable y reproducible
- Docker isola dependencias y evita conflictos
- Sail simplifica la gestión de servicios Laravel
- El alias hace el trabajo diario mucho más fluido

Lo Malo:

- Windows añade complejidad innecesaria
- Configuración inicial toma tiempo
- Errores crípticos requieren experiencia para resolver
- La gestión del PATH en WSL puede ser traicionera

Lo Aprendido:

- **DevOps es complejo al principio** pero vale la pena
- **Cada error enseña** algo importante
- **La paciencia es clave** en la configuración inicial
- **La documentación salva vidas** (y tiempo futuro)

Recomendación Final:

Si es posible, considera migrar a **macOS** o **Linux** para desarrollo web. La experiencia es significativamente mejor. Mientras tanto, esta guía te permitirá trabajar productivamente en Windows con todas las comodidades.

Enlaces Útiles

- [Documentación Laravel Sail](#)
- [Docker Desktop WSL 2](#)
- [Guía WSL Microsoft](#)
- [DBeaver Community](#)

Nota Personal

Esta guía fue creada basándose en problemas REALES encontrados durante la configuración. Cada error, solución y "momento de frustración" fue documentado para ayudar a otros desarrolladores que enfrentan los mismos desafíos en Windows.

¡Especial agradecimiento por la paciencia durante el proceso de debugging! 😊

 **Última actualización:** Junio 2025

 **Estado:** Completamente funcional y probado

 **Entorno:** Windows 11 + WSL 2 + Docker Desktop + Laravel Sail