HOJA DE REFERENCIA DESARROLLADOR CLI, GIT & ARQUITECTURA

Herramientas de Código Abierto, Atajos de Terminal y Arquitectura (2026)

1. Flujo de Trabajo Git Moderno & Control de Versiones

Motor VCS

- `git switch -c <rama>` — Establecer nueva rama y cambiar inmediatamente.
- `git commit --amend --no-edit` — Agregar cambios olvidados al último commit.
- `git rebase -i HEAD~3` — Rebase interactivo para unificar (squash) los commits.
- `git cherry-pick <hash>` — Incorporar un commit puntual desde otra rama.
- `git stash push -m "nota"` — Guardar trabajo en progreso; reanudar con `stash pop`.
- `git restore --staged <archivo>` — Quitar archivo del área de staging sin perder edición.
- `git reset --soft HEAD~1` — Deshacer commit conservando archivos en staging.
- `git log --oneline --graph --all` — Visualizar el árbol de ramas mediante ASCII limpio.

2. GitHub CLI (gh) & Atajos de Lazygit

TUI & CLI

Herramienta	Comando / Tecla	Funcionalidad y Ganancia de Productividad
gh	`gh pr create --web`	Abrir navegador para publicar Pull Request con plantilla.
gh	`gh pr checkout <no>`	Descargar rama de PR localmente y activarla de inmediato.
gh	`gh run watch`	Monitorear en vivo la ejecución de flujos de GitHub Actions.
lazygit	`Espacio` / `c` / `P`	Agregar a staging / Escribir commit / Enviar upstream (Push).
lazygit	`z` / `e` / `[ ]`	Deshacer acción / Abrir en editor / Alternar entre paneles.

3. Gestores de Paquetes y Entornos de Alta Velocidad

uv • pnpm • fnm

```
# uv (Python - Construido en Rust, 10 a 100x más rápido que pip)
uv venv && source .venv/bin/activate # Entorno virtual en milisegundos
uv pip install -r req.txt # Caché global para instalación ultra rápida
uv run script.py # Ejecuta dependencias en entorno aislado efímero
```

```
# pnpm (Node.js - Enlaces duros que ahorran 80% de disco)
pnpm install && pnpm add -D typescript # Almacén direccionable por contenido
pnpm dlx degit <repo> <destino> # Ejecución instantánea sin instalación previa
```

```
# fnm (Fast Node Manager - Cambio instantáneo en Rust)
fnm install --lts && fnm use 22 # Cero latencia de inicio en el shell
```

4. Navegación en Terminal & Atajos Readline

Atajos Rápidos

Tecla	Acción	Tecla	Acción
`Ctrl + A`	Mover cursor al inicio de línea	`Ctrl + E`	Mover cursor al final de línea
`Ctrl + U`	Cortar desde cursor al inicio	`Ctrl + K`	Cortar desde cursor al final
`Ctrl + W`	Borrar palabra anterior	`Ctrl + Y`	Pegar (Yank) último texto cortado
`Ctrl + R`	Búsqueda inversa en historial	`Ctrl + L`	Limpiar la pantalla de la terminal
`Alt + .`	Insertar último argumento usado	`Ctrl + C`	Interrumpir proceso en ejecución

5. Códigos de Estado HTTP & Arquitectura REST

Estándares API

Código	Definición	Uso Práctico y Nota de Arquitectura
200 OK	Éxito	Petición GET/PUT/PATCH procesada con éxito.
201 Created	Recurso Producido	POST produjo un nuevo recurso (Encabezado Location retornado).
204 No Content	Cuerpo Vacío	Operación exitosa; sin contenido en la respuesta.
304 Not Mod.	Caché Válida	ETag verificado; cero transferencia de datos.
400 Bad Req.	Petición Inválida	Sintaxis errónea o validación de esquema rechazada.
401 Unauth.	No autenticado	Faltan credenciales o token de acceso no es válido.
403 Forbidden	Acceso Prohibido	Identidad válida pero carece de permisos RBAC.
404 Not Found	No Encontrado	El recurso solicitado no existe en el servidor.
409 Conflict	Conflicto	Restricción de clave única o conflicto de versiones.
422 Unproc.	Entidad Invalida	JSON sintácticamente válido pero falla lógica de negocio.
429 Rate Limit	Límite Excedido	Demasiadas peticiones; revisar encabezado Retry-After.
500 / 502 / 503	Errores Servidor	Fallo interno / Puerta de enlace fallida / Servidor saturado.

6. Expresiones Regulares (Regex) Esenciales

Validación

Correo: `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`

URL (HTTP/S): `^https?:\/\/[\w.-]+(?:\.[\w\.-]+)+[\w\-\._~:/?#[\]@!\$&'\(\)\*\+,;=.]+$`

Dirección IPv4: `^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$`

UUID v4: `^[0-9a-f]{8}-[0-9a-f]{4}-4[0-9a-f]{3}-[89ab][0-9a-f]{3}-[0-9a-f]{12}$`

Fecha ISO 8601: `^\d{4}-(?:0[1-9]|1[0-2])-(?:0[1-9]|[12]\d|3[01])$`

Versión Semántica: `^v?(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-[0-9A-Za-z.-]+)?$`

7. Indexación SQL & Arquitectura Relacional

Reglas SGBD

Regla del Prefijo Izquierdo: `INDEX(a, b, c)` optimiza únicamente búsquedas en `(a)`, `(a, b)` o `(a, b, c)`. Una consulta con `WHERE b = 5` causará un Full Table Scan.

Mecánica de JOINs: `INNER` devuelve la intersección. `LEFT JOIN` preserva todas las filas izquierdas. Evite `SELECT *` para minimizar consumo de memoria e I/O.

Garantías ACID: Atomicidad (todo o nada), Consistencia (integridad de reglas), Aislamiento (concurrencia segura), Durabilidad (persistencia en disco).

Análisis de Consultas: Examine siempre las consultas lentas con `EXPLAIN ANALYZE` para verificar el uso de índices sobre escaneos secuenciales.

8. Utilidades CLI Modernas (Ecosistema Rust & Go)

Aceleradores

Clásico	Moderno	Ventaja de Rendimiento & Funcionalidad
`cd`	**zoxide (z)**	Aprende sus hábitos; salta directamente con `z proyecto`.
`cat`	**bat**	Resaltado de sintaxis, números de línea e integración Git.
`grep`	**ripgrep (rg)**	Hecho en Rust, respeta .gitignore, búsqueda hiperveloz.
`find`	**fd**	Sintaxis intuitiva con exploración paralela de directorios.
`ls`	**eza**	Iconos, permisos detallados y estado Git con colores.