AUTOLOGBUCH

Installation unter Windows 11

Kurzanleitung
für Einsteiger

Diese Anleitung führt Schritt für Schritt durch die Installation von AutoLogbuch auf einem eigenen Windows-11-PC — praktisch zum Ausprobieren oder als dauerhafte Lösung, wenn der Rechner ohnehin durchläuft. Genutzt wird **Docker Desktop** mit grafischer Oberfläche; nur für ein paar Befehle wird kurz ein **PowerShell-Fenster** gebraucht (in Windows bereits eingebaut, nichts zusätzlich nötig). Jeder Befehl steht hier vollständig zum Abtippen bzw. Kopieren bereit. Der **Grundteil (A–G)** reicht aus, damit die App auf diesem PC nutzbar ist.

GRUNDINSTALLATION (A–G)
App läuft auf diesem PC, im Browser über `localhost` erreichbar. Ca. 30–45 Minuten (inkl. Wartezeiten). Kein Vorwissen nötig.

OPTIONAL: VON ANDEREN GERÄTEN ERREICHBAR, AUTOSTART (H–I)
App auch von Handy/Tablet im selben WLAN nutzen, automatisch starten beim Hochfahren. Setzt A–G voraus.

Das brauchst du

- Einen PC oder Laptop mit **Windows 11** (jede Edition, auch „Home" reicht aus) — 64-Bit-Prozessor mit aktivierter Virtualisierung (bei den meisten PCs ab Werk bereits an)
- Admin-Rechte auf diesem PC (für die einmalige Einrichtung), mindestens **6 GB freien Speicherplatz** und **8 GB RAM** (4 GB reichen technisch, 8 GB empfohlen)
- Einen kostenlosen GitHub-Account (nur zum Herunterladen der App-Dateien, siehe Teil D)
- Internetzugang für die einmalige Einrichtung

A Virtualisierung (WSL2) aktivieren

1 PowerShell als Administrator öffnen

Startmenü öffnen, „PowerShell" eingeben, mit **Rechtsklick → Als Administrator ausführen** starten. Windows fragt danach, ob Änderungen zugelassen werden sollen — mit **Ja** bestätigen.

2 WSL2 mit einem Befehl einrichten

Den Befehl rechts eintippen und mit **Eingabe** bestätigen. Dieser eine Befehl aktiviert alle nötigen Windows-Funktionen und installiert die Linux-Umgebung, die Docker Desktop im Hintergrund braucht — **Docker selbst wird dabei noch nicht installiert**, das folgt in Teil B.

War WSL2 bereits aktiv (z. B. weil schon einmal genutzt), meldet der Befehl das einfach zurück — kein Fehler, einfach weiter mit Schritt 3.

3 PC neu starten

Nach Abschluss des Befehls den PC einmal neu starten, damit die Änderung wirksam wird.

Administrator: Windows PowerShell

```
PS> wsl --install
Wird heruntergeladen: Windows-Subsystem für Linux
Wird heruntergeladen: Ubuntu
// … ein bis zwei Minuten Textausgabe …
Der Vorgang wurde erfolgreich abgeschlossen.
```

Ist Virtualisierung überhaupt aktiv?
Task-Manager öffnen (`Strg+Umschalt+Esc`) → Reiter **Leistung** → **CPU**: unten muss „Virtualisierung: Aktiviert" stehen. Falls nicht, im BIOS/UEFI des PCs nachsehen (Bezeichnung je Hersteller unterschiedlich, oft „Intel VT-x"/ „AMD-V") — bei den meisten aktuellen PCs ist das bereits ab Werk eingeschaltet.

B Docker Desktop installieren

4 Herunterladen und installieren

`docker.com/products/docker-desktop` öffnen, **Download for Windows** anklicken, die heruntergeladene Datei ausführen. Der Haken bei „Use WSL 2 instead of Hyper-V" ist bereits Standard — nichts umstellen, einfach durchklicken.

5 Docker Desktop starten

Nach der Installation (ggf. erneuter Neustart) Docker Desktop über das Startmenü öffnen. Nutzungsbedingungen akzeptieren — ein **Docker-Konto ist für die reine Nutzung auf diesem PC nicht nötig**, ein Anmelde-Hinweis kann übersprungen werden. Docker Desktop läuft danach im Hintergrund (Symbol unten rechts in der Taskleiste).

Der erste Start dauert manchmal etwas länger, während Docker Desktop die WSL2-Umgebung fertig einrichtet — das Symbol in der Taskleiste zeigt an, sobald „Docker Desktop is running" erreicht ist.

Docker Desktop Installer

☑ Use WSL 2 instead of Hyper-V (empfohlen)

☐ Verknüpfung auf dem Desktop hinzufügen

Ok

Fertig eingerichtet, sobald…
… das Wal-Symbol in der Taskleiste ruhig steht (nicht mehr animiert) und ein Klick darauf „Docker Desktop is running" zeigt.

C Git installieren

6 Git für Windows herunterladen

`git-scm.com/download/win` öffnen — der Download startet automatisch. Installer ausführen und mit den **Standardeinstellungen** einfach mehrfach „Next" klicken, am Ende „Install", danach „Finish". Git wird gebraucht, um die App-Dateien von GitHub herunterzuladen.

D App-Dateien laden

7 PowerShell öffnen (diesmal ohne Administrator)

Startmenü → „PowerShell" eingeben → mit einfachem Klick (nicht „Als Administrator") öffnen.

8 App-Dateien herunterladen

Ersten Befehl eintippen (der Zugriffstoken ist bereits enthalten) — legt den Ordner `autoapp` im Nutzer-Verzeichnis automatisch mit allen App-Dateien an. Danach mit dem zweiten Befehl in den neuen Ordner wechseln — **ab jetzt bleiben alle weiteren Befehle in diesem Ordner**, das PowerShell-Fenster kann dafür einfach offen bleiben.

Windows PowerShell

```
PS> git clone https://github_pat_11AB…
@github.com/…/AutoLogbuch.git autoapp
PS> cd autoapp
```

E Zugangsdaten eintragen (.env-Datei)

9 Vorlage kopieren

Ersten Befehl rechts eintippen — legt eine Kopie namens `.env` an, die eigentliche Vorlage `.env.example` bleibt unverändert erhalten.

10 Datei mit Editor öffnen und einen Wert ändern

Zweiten Befehl eintippen — öffnet die Datei direkt im gewohnten **Editor** (Notepad). Dort **nur** die folgende Zeile ändern, alles andere unverändert lassen:

`POSTGRES_PASSWORD` – frei erfundenes Passwort, mind. 12 Zeichen

`PUBLIC_APP_URL` steht bereits passend auf `http://localhost:8085` für die Nutzung auf diesem PC — nur für Teil H (von anderen Geräten erreichbar) muss das später angepasst werden. Stehen bereits Zeilen wie `SMTP_…`, `ANTHROPIC_API_KEY=…` oder `FEEDBACK_SYNC_ROLE=…` mit einem Wert darin? Dann nichts ändern — E-Mail-Versand, die optionale KI-Felderkennung und ggf. die Verbindung zu einer zentralen App sind bereits fertig eingerichtet. Soll diese Verbindung erst noch eingerichtet werden: optionaler **Teil J** auf der letzten Seite.

11 Speichern und schließen

`Strg+S` speichert, Editor-Fenster danach einfach schließen.

Windows PowerShell — autoapp

```
PS> copy .env.example .env
PS> notepad .env
```

Editor — .env

```
1  POSTGRES_USER=autoapp
2  POSTGRES_PASSWORD=MeinSicheres-Passwort9
3  POSTGRES_DB=autoapp
4  …
9  PUBLIC_APP_URL=http://localhost:8085
```

F App bauen und starten

12 Ersten Bauvorgang anstoßen

Im PowerShell-Fenster (weiterhin im Ordner `autoapp`) den Befehl rechts eintippen. Baut Datenbank, Backend und Oberfläche und startet alles automatisch. Dauert beim allerersten Mal **10–20 Minuten** — das Fenster bleibt währenddessen mit Textausgaben beschäftigt, bis am Ende wieder die normale Eingabezeile erscheint.

13 Status prüfen

Mit dem zweiten Befehl den Status aller drei Bausteine prüfen — bei allen drei sollte `running` bzw. `healthy` stehen.

Windows PowerShell — autoapp

```
PS> docker compose up -d --build
// … mehrere Minuten Textausgabe …

PS> docker compose ps
postgres running (healthy)
backend running
frontend running
```

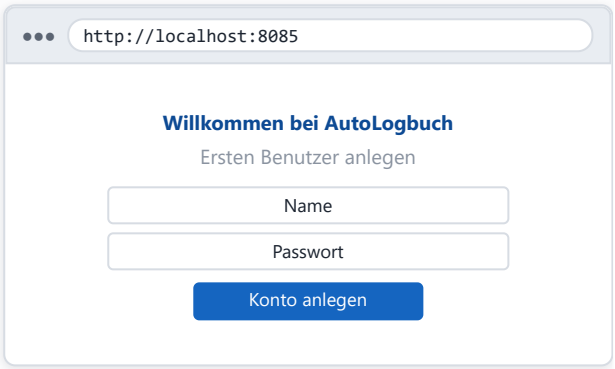
Was passiert hier?

Docker Desktop lädt und baut im Hintergrund die drei Bausteine der App und startet sie — sichtbar auch als grafische Übersicht direkt in Docker Desktop selbst (Reiter „Containers").

G App öffnen und einrichten

14 Adresse im Browser öffnen

Beliebigen Browser öffnen, `http://localhost:8085` aufrufen. Der Einrichtungs-Assistent legt beim allerersten Aufruf den ersten Benutzer an — **fertig!**



Grundinstallation abgeschlossen

Die App ist jetzt auf diesem PC nutzbar. Für Zugriff von anderen Geräten im WLAN weiter mit Teil H, für automatischen Start beim Hochfahren mit Teil I.

Testphase

Ohne Lizenzschlüssel ist die App ab jetzt 30 Tage lang voll nutzbar. Danach kann nur noch gelesen werden, bis der erste angelegte Benutzer unter **Mein Konto** einen Lizenzschlüssel einträgt — dafür bei der Person nachfragen, die diese Anleitung zur Verfügung gestellt hat.

H Optional: von anderen Geräten im WLAN erreichbar machen

Standardmäßig ist die App nur auf diesem PC selbst erreichbar. Für den Zugriff von Handy, Tablet oder einem anderen PC im selben WLAN sind zwei zusätzliche Schritte nötig.

H1 IP-Adresse dieses PCs ermitteln

Im PowerShell-Fenster den Befehl rechts eintippen, die Zeile „IPv4-Adresse" unter dem aktiven Netzwerkadapter (WLAN oder LAN) notieren.

Windows PowerShell

```
PS> ipconfig
IPv4-Adresse. . . . . . . . . . . : 192.168.1.60
```

H2 Adresse eintragen und neu starten

In der `.env` (Teil E) `PUBLIC_APP_URL` auf `http://PC-IP:8085` ändern (eigene IP aus H1 einsetzen), speichern, danach im PowerShell-Fenster erneut `docker compose up -d --build` ausführen, damit die Änderung übernommen wird.

H3 Windows-Firewall freigeben

Windows blockiert eingehende Verbindungen von anderen Geräten standardmäßig. Befehl rechts in der **administrativen** PowerShell (wie Teil A, Schritt 1) einmalig ausführen — gibt Port 8085 gezielt für andere Geräte im selben Netzwerk frei.

Administrator: Windows PowerShell

```
PS> New-NetFirewallRule -DisplayName "AutoLogbuch"
-Direction Inbound -LocalPort 8085 -Protocol TCP -
Action Allow
```

Danach von jedem Gerät im selben WLAN: `http://192.168.1.60:8085` öffnen (eigene IP-Adresse einsetzen).

I Optional: automatischer Start beim Hochfahren

I1 Docker Desktop automatisch starten lassen

Docker Desktop öffnen → **Einstellungen** (Zahnrad oben rechts) → **General** → Haken bei **„Start Docker Desktop when you sign in"** setzen.

Alle drei Bausteine der App sind bereits so eingerichtet, dass sie sich selbst neu starten (`restart: unless-stopped`) — sobald Docker Desktop hochfährt, startet die App also automatisch mit, ganz ohne zusätzliches Skript oder erneutes Eintippen von Befehlen.

J Optional: Verbindung zu einer zentralen Installation (Feedback-Synchronisation)

Nur nötig, wenn über **„Feedback einreichen"** eingereichte Meldungen dieser Installation automatisch an eine bereits bestehende ANDERE AutoLogbuch-Installation (die „Zentrale") weitergeleitet werden sollen. Nur eine Installation im Einsatz: dieser Teil kann übersprungen werden.

J1 Zugangs-Token auf der Zentrale erzeugen

Auf der ANDEREN, bereits bestehenden Installation (der Zentrale) als Admin einloggen → **Einstellungen** → **Installationen** → **Neue Installation**, einen Namen vergeben. Der Zugangs-Token wird nur EINMAL im Klartext angezeigt — sicher an die Person weitergeben, die DIESEN PC hier betreibt.

Windows PowerShell — autoapp

```
PS> docker compose up -d --build
```

Dieser PC soll selbst die Zentrale sein?
Dann `FEEDBACK_SYNC_ROLE=central` statt `satellite` setzen (J2 sonst entfällt) — jede WEITERE Installation durchläuft dann J1–J3 mit DIESER Adresse als Zentrale.

J2 Zugangsdaten in dieser Installation eintragen

Wie in Teil E: `notepad .env` öffnen, folgende drei Zeilen eintragen bzw. anpassen:

```
FEEDBACK_SYNC_ROLE = satellite
FEEDBACK_CENTRAL_URL – Adresse der Zentrale
FEEDBACK_CENTRAL_TOKEN – der in J1 erzeugte Token
```

J3 Neu starten

Wie im PowerShell-Fenster gezeigt: `docker compose up -d --build` erneut ausführen. Ab jetzt werden eingereichte Meldungen automatisch (alle 15 Minuten) an die Zentrale übertragen, deren Bearbeitungsstatus wird ebenfalls automatisch zurückgemeldet. Ohne diese drei Zeilen funktioniert „Feedback einreichen" weiterhin ganz normal, nur rein lokal — dieser Teil ist jederzeit nachträglich einrichtbar.

? Häufige Fragen

Docker Desktop meldet einen WSL-Fehler beim Start. Administrative PowerShell öffnen, `wsl --update` ausführen, danach den PC neu starten und Docker Desktop erneut öffnen.
Ein Baustein steht nicht auf „running"/„healthy". Mit `docker compose logs backend` (bzw. `postgres` / `frontend`) die Fehlermeldung ansehen — meist ein falsch eingetragener Wert in der `.env` (Teil E), danach `docker compose up -d --build` erneut ausführen.
Die Passwort-vergessen-E-Mail kommt nicht an. Der E-Mail-Versand ist bereits fertig eingerichtet (Teil E) — dafür bei der Person nachfragen, die diese Anleitung zur Verfügung gestellt hat. **Teil D, Schritt 8 lädt nichts herunter?** Der enthaltene Zugriffstoken kann abgelaufen sein — ebenfalls bei dieser Person nachfragen, sie kann einen neuen ausstellen. **Nach 30 Tagen nur noch Lesezugriff?** Testphase abgelaufen — Lizenzschlüssel bei dieser Person anfragen.
Updates erhalten. Im Ordner `autoapp` per PowerShell: `git pull; docker compose up -d --build` — deine Daten (Fahrzeuge, Einträge) bleiben unberührt.
PC bleibt für die App durchgehend an? Nur nötig, wenn die App dauerhaft erreichbar sein soll (z. B. Teil H) — Energieoptionen so einstellen, dass der PC nicht in den Ruhezustand wechselt (Einstellungen → System → Energie → „Ruhezustand": Nie).