`souffleur` — Kennzahlen-Dashboard für Theater

Systembeschreibung

Dokumenttyp: Funktions- und Technikbeschreibung zur Vorlage bei der Compliance-Stelle **Stand:** Juli 2026 **Hinweis:** Diese Beschreibung ist bewusst allgemein gehalten. Sie nennt keine Namen von Personen, nutzenden Häusern oder Servern und ist auf beliebige Häuser (Theater, Bühnen, Veranstaltungsorte) übertragbar, die die genannten Systeme einsetzen. Namentlich genannt ist allein die entwickelnde Organisation, die das Werkzeug als Open Source bereitstellt (Abschnitt 5).

---

1. Worum geht es? (Zusammenfassung für Nicht-Techniker)

Theater arbeiten oft mit **getrennten, untereinander inkompatiblen Software-Systemen** — etwa für die Planung und den Kartenverkauf. Im hier beschriebenen Aufbau sind das diese beiden:

1. **artwork** — eine Open-Source-Software für die *Produktions- und Veranstaltungsplanung*. Hier wird geplant: Welche Vorstellung findet wann in welchem Raum statt, welche Mitarbeitenden haben welche Schichten, wie viele Stunden wurden gearbeitet, wer hat Urlaub.
2. **Pretix** — eine Ticketing-Software (Kartenverkauf). Hier steht, wie viele Plätze eine Vorstellung hat, wie viele Karten verkauft wurden und welche Einnahmen erzielt wurden.

Die beiden Systeme wissen nichts voneinander. Fragen wie *„Wie ausgelastet war die Vorstellung am Freitag und was hat sie eingebracht?“* lassen sich bisher nur beantworten, indem jemand von Hand in beiden Systemen nachschaut und die Zahlen in eine Tabelle überträgt.

`souffleur` **löst genau dieses Problem:** Die Anwendung ist eine kleine Webanwendung (eine Internetseite mit Login), die die Daten aus diesen Systemen **automatisch zusammenführt** und übersichtlich darstellt — pro Veranstaltung: Titel, Datum, Raum, Kapazität, verkaufte Karten, Auslastung in Prozent und Umsatz. Zusätzlich zeigt es einen Überblick über die Ressourcen des Hauses (geplante Schichten, Stundenkonten, Überstunden, Urlaub, offene Aufgaben).

Zum Namen: Der Souffleur ist am Theater die Person, die den Akteuren unauffällig den Text zuflüstert, wenn er gerade fehlt — nichts anderes tut diese Anwendung mit den Zahlen des Hauses. Die Kleinschreibung (`souffleur`) folgt der Konvention für Software-Werkzeuge.

Wichtig zum Verständnis: Das Dashboard ist ein **reines Anzeige-Werkzeug**. Es kann in keinem der angeschlossenen Systeme etwas ändern, löschen oder anlegen — es liest nur.

Die Anwendung ist zudem bewusst **leichtgewichtig**: Sie kann auf einem gemieteten Server laufen — oder **im Haus selbst**, auf bereits vorhandener Technik wie einem Synology-NAS oder einem Raspberry Pi (Einzelheiten in Abschnitt 3.4). Die Daten müssen das Haus dafür nicht verlassen.

---

2. Funktionsweise im Detail

2.1 Was der Nutzer sieht

Nach dem Login (Benutzername/Passwort, Übertragung ausschließlich verschlüsselt per HTTPS) sieht der Nutzer:

- **Veranstaltungsübersicht:** Eine Tabelle aller Veranstaltungen im gewählten Zeitraum. Pro Zeile: Titel, Datum, Raum, Projekt/Produktion, Veranstaltungsart (aus der Planungssoftware) sowie Kapazität, verkaufte Tickets, Auslastung in Prozent und Umsatz (aus dem Ticketing).
- **Ressourcen des Hauses:** Kennzahlen aus der Planungssoftware — Stundenkonten und Überstunden der Mitarbeitenden, geplante Schichten je Gewerk, offene Aufgaben, Urlaubstage.

2.2 Woher kommen die Daten?

Das Dashboard hat **keinen eigenen Datenbestand**. Bei jedem Aufruf holt es die aktuellen Zahlen live aus den Quellsystemen:

Quellsystem	Was wird gelesen	Wie wird zugegriffen
artwork (Planung)	Veranstaltungen (Titel, Datum, Raum, Projekt, Art), Schichten, Stundenkonten, Überstunden, Urlaub, Aufgaben	Direkter **Nur-Lese-Zugriff** auf die Datenbank. Der verwendete Datenbank-Benutzer hat technisch ausschließlich Leserechte (SELECT) — Schreiben ist auf Datenbankebene unmöglich.
Pretix (Ticketing)	Veranstaltungen, Kontingente (Kapazität), verkaufte Karten, Umsätze	Offizielle **REST-API** von Pretix mit einem Zugriffstoken, das der Betreiber des Ticketing-Kontos selbst erzeugt und jederzeit widerrufen kann. Das Token wird mit minimal nötigen Rechten (nur Lesen) angelegt.

Die Zahlen zur **Auslastung** (Kapazität und verkaufte Plätze) stammen vollständig aus dem Ticketing-System; die Planungssoftware liefert den Kontext (welcher Raum, welche Produktion, welche Veranstaltungsart).

2.3 Wie werden die Daten zusammengeführt?

Planungssoftware und Ticketing vergeben keine gemeinsame Kennnummer für dieselbe Veranstaltung. Das Dashboard ordnet daher automatisch zu über:

- **Titelähnlichkeit** (Groß-/Kleinschreibung, Umlaute und Sonderzeichen werden ignoriert) **und**
- **Datumsnähe** (die Termine dürfen höchstens einen Tag auseinanderliegen).

Findet sich keine Entsprechung, wird die Veranstaltung trotzdem angezeigt — mit dem Hinweis, dass sie nur in einem der beiden Systeme existiert. So gehen keine Daten verloren und Zuordnungsfehler sind sichtbar statt versteckt. Für den Dauerbetrieb ist empfohlen, im

Planungssystem die Ticketing-Kennung (den sogenannten Pretix-„Slug") zu hinterlegen; dann entfällt die Zuordnung per Ähnlichkeit.

2.4 Was das Dashboard NICHT tut

- Es **schreibt keine Daten** in die Quellsysteme (technisch unterbunden, nicht nur organisatorisch).
- Es **speichert keine Daten dauerhaft** — keine eigene Datenbank, keine Kopien der Quelldaten, keine Protokollierung von Inhalten. Angezeigt wird immer der Live-Stand.
- Es gibt **keine Daten an Dritte weiter** und bindet keine externen Dienste ein (kein Tracking, keine Analyse-Dienste, keine Schriften oder Skripte von Fremdservern).

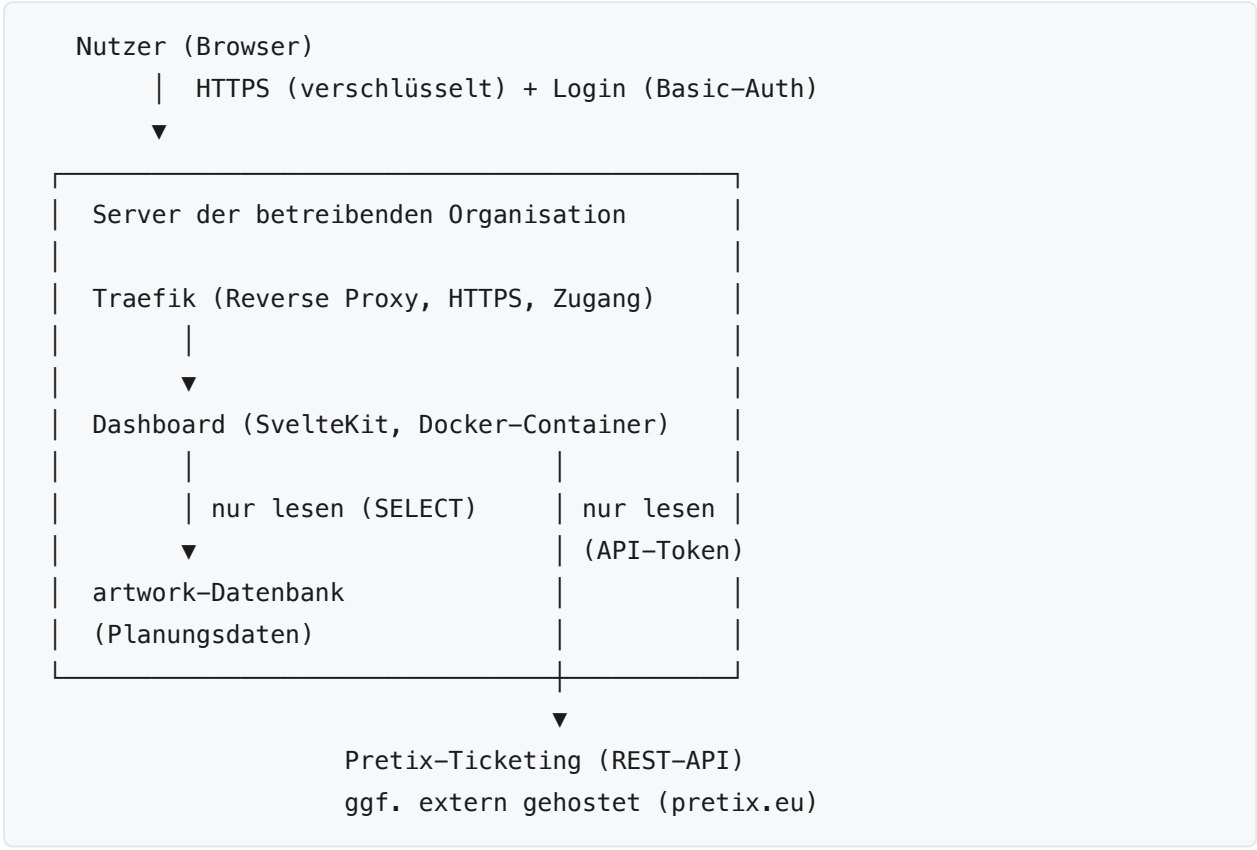
---

3. Technischer Aufbau (Techstack)

3.1 Komponenten

Schicht	Technologie	Zweck
Webanwendung	**SvelteKit** (Node.js, TypeScript)	Framework für die Anwendung; erzeugt die Seiten und holt serverseitig die Daten
Datenbank-Anbindung	**mysql2** (Node.js-Bibliothek)	Nur-Lese-Verbindung zur artwork-Datenbank (MariaDB/MySQL)
API-Anbindung	HTTPS-Aufrufe der **Pretix REST-API**	Ticketing-Daten (Token-authentifiziert)
Container	**Docker / Docker Compose**	Die Anwendung läuft isoliert in einem Container
Reverse Proxy	**Traefik**	Nimmt Anfragen aus dem Internet entgegen, erzwingt HTTPS (Zertifikate via Let's Encrypt) und schaltet die Zugangskontrolle (Basic-Auth) davor
Betriebsumgebung	Linux (x86 oder ARM)	Wahlweise gemieteter Server bei einem europäischen Hosting-Anbieter **oder** hauseigene Hardware — siehe Abschnitt 3.4 (Betriebsmodelle)

3.2 Architektur-Übersicht



Ein entscheidender Punkt für die Bewertung: **Alle Zugangsdaten (Datenbank-Passwort, API-Tokens) liegen ausschließlich auf dem Server** — in einer Konfigurationsdatei, die nur der Server-Administrator lesen kann. Sie werden niemals an den Browser des Nutzers übertragen und sind nicht im Quellcode oder in der Versionsverwaltung enthalten. Der Browser erhält nur die fertigen, aufbereiteten Zahlen.

3.3 Quellcode-Verwaltung

Der Quellcode wird in einem **öffentlichen Git-Repository** unter einer Open-Source-Lizenz geführt (Einzelheiten in Abschnitt 5). Zugangsdaten und Konfigurationsdateien mit Geheimnissen sind vom Repository ausgeschlossen und werden durch die Veröffentlichung nicht offengelegt.

3.4 Betriebsmodelle: gemieteter Server oder eigene Hardware (On-Premise)

Weil die Anwendung nur aus einem einzelnen Container besteht, keinen eigenen Datenbestand hält und nur von wenigen benannten Personen aufgerufen wird, sind ihre Anforderungen an die Hardware minimal. Sie läuft überall dort, wo die Container-Technik Docker verfügbar ist — und damit auch auf Geräten, die in vielen Häusern **bereits vorhanden** sind:

Betriebsmodell	Beispiel	Typischer Fall
Gemieteter Server	Linux-Server bei einem europäischen Hosting-Anbieter	Betrieb durch einen IT-Dienstleister für das Haus (aktuelles Modell)
Synology-NAS	Netzwerkspeicher, wie er in vielen Verwaltungen ohnehin für Dateiablage/Backup steht — der mitgelieferte „Container Manager" genügt. Empfohlene Modelle: **Synology DS923+** (Tischgerät) bzw. **RS822+** (Rack-Einbau, 19 Zoll) — beide herstellerseitig auf bis zu 32 GB Arbeitsspeicher erweiterbar	Haus betreibt das Dashboard selbst, ohne neue Hardware anzuschaffen — oder gleich die gesamte Systemlandschaft (siehe unten)
Raspberry Pi	Kleinstrechner (Anschaffung ca. 50–100 €, wenige Watt Stromverbrauch)	Kostengünstiger Einstieg für das Dashboard allein — da es keinen eigenen Datenbestand hält, geht bei einem Geräteausfall nichts verloren. Für den Betrieb der Quellsysteme mit ihren Produktivdaten ist ein NAS mit Plattenspiegelung die richtige Wahl
Mini-PC	Kompaktrechner (z. B. Lenovo ThinkCentre Tiny, neu ca. 600–700 €, gebraucht deutlich darunter; Arbeitsspeicher günstig erweiterbar)	Preisgünstigste Option für die **gesamte Systemlandschaft**: Der Mini-PC übernimmt das Rechnen, ein bereits vorhandenes NAS — auch ein leistungsschwaches — dient als Ziel für die nächtlichen, versionierten Sicherungen. Leistung vergleichbar mit oder besser als bei den genannten NAS-Modellen
Vorhandener Server / virtuelle Maschine	Bestehende Server-Infrastruktur des Hauses (x86 oder ARM)	Häuser mit eigener IT

Ein ausreichend ausgestattetes NAS wie die genannte DS923+ — oder ebenso ein Mini-PC — kann dabei mehr übernehmen als nur das Dashboard: Auch die **Quellsysteme selbst** — die Produktionsplanung (artwork), eine selbst gehostete Ticketing-Instanz (Pretix) und bei Bedarf eine Kollaborationsplattform wie Nextcloud — laufen als Container auf demselben Gerät. Ein Haus kann so seine gesamte Systemlandschaft auf eigener Hardware betreiben; die vollständige Datenhoheit (siehe unten) erstreckt sich dann nicht nur auf das Dashboard, sondern auf alle beteiligten Systeme einschließlich der Ticketkäuferdaten.

Für die Compliance-Bewertung ist das On-Premise-Modell besonders relevant:

- **Datenhoheit:** Läuft das Dashboard im Haus, verlassen die zusammengeführten Daten (insbesondere die Beschäftigtendaten aus der Planungssoftware) das Haus nicht. Nur der Abruf der Ticketing-Zahlen geht — verschlüsselt — nach außen, sofern das Ticketing extern gehostet ist.
- **Kein zusätzlicher Dienstleister:** Für das Dashboard selbst ist dann kein Hosting-Anbieter und damit ggf. kein zusätzlicher Auftragsverarbeitungsvertrag erforderlich.
- **Zugriff aufs Hausnetz beschränkbar:** Die Anwendung muss nicht öffentlich im Internet erreichbar sein — sie kann ausschließlich im internen Netzwerk (oder per VPN) zugänglich gemacht werden.

Die Schutzmaßnahmen aus Abschnitt 4.2 gelten in beiden Modellen unverändert: verschlüsselter Zugriff, Passwortschutz, Minimalrechte, Geheimnisse nur in einer geschützten Konfigurationsdatei. Der Umzug zwischen den Modellen ist einfach, da die gesamte Anwendung als ein Container ausgeliefert wird — es ändert sich nur der Ort, an dem dieser Container läuft.

---

4. Datenschutz und Sicherheit

4.1 Betroffene Datenarten

Datenart	Personenbezug	Quelle
Veranstaltungsdaten (Titel, Datum, Raum, Kapazität, Verkäufe, Umsatz)	Nein	artwork + Pretix
Schichtplanung, Stundenkonten, Überstunden, Urlaub, Aufgaben	**Ja** — Namen und Arbeitszeitdaten von Mitarbeitenden	artwork

Die Ressourcen-Ansicht zeigt personenbezogene Daten von Beschäftigten (Name, Gewerk, Stunden-/Urlaubssalden). Diese Daten existieren bereits im Planungssystem des Hauses; das Dashboard erzeugt keine neuen Daten, sondern stellt vorhandene dar. **Keine Ticketkäufer-Daten:** Aus dem Ticketing werden nur aggregierte Zahlen gelesen (Summen je Veranstaltung), keine Namen, Adressen oder Zahlungsdaten von Kundinnen und Kunden.

Für den Einsatz in einem konkreten Haus sind daher zu klären (organisatorisch, außerhalb dieser Systembeschreibung): Rechtsgrundlage bzw. Vereinbarung zur Anzeige von Beschäftigtendaten (ggf. Beteiligung der Personalvertretung), Kreis der zugriffsberechtigten Personen und ggf. ein Auftragsverarbeitungsvertrag mit dem Hosting-Anbieter des Ticketing-Systems.

4.2 Technische Schutzmaßnahmen

- **Transportverschlüsselung:** Sämtlicher Zugriff ausschließlich über HTTPS (Let's-Encrypt-Zertifikate, automatische Erneuerung).
- **Zugangskontrolle:** Die Anwendung ist nicht öffentlich einsehbar; vorgeschaltet ist eine Passwort-Abfrage (Basic-Auth auf Proxy-Ebene). Der Nutzerkreis ist klein und benannt.

- **Minimalrechte-Prinzip:**
  - Datenbank-Zugriff nur mit einem eigens angelegten Nur-Lese-Benutzer (ausschließlich SELECT-Recht);
  - API-Zugriff nur mit Lese-Token, das der Dateneigentümer selbst erzeugt und jederzeit entziehen kann.
- **Keine Datenhaltung:** Das Dashboard speichert keine Quelldaten; ein Datenleck im Dashboard könnte daher keinen historischen Datenbestand offenlegen.
- **Serverseitige Geheimnisse:** Zugangsdaten liegen nur in einer geschützten Konfigurationsdatei auf dem Server (Dateirechte 600), nicht im Code, nicht im Browser.
- **Serverhärtung (Betriebsumgebung):** Zugang zum Server nur per SSH-Schlüssel, Firewall mit minimalen offenen Ports, Einbruchserkennungs-/Sperr-Mechanismus (fail2ban), Container-Isolation.

4.3 Rollen

- **Dateneigentümer** bleibt das jeweilige Haus: Es kontrolliert das Ticketing-Token und den Datenbank-Zugang und kann beides jederzeit widerrufen — damit ist das Dashboard sofort „blind".
- **Betreiber** des Dashboards ist die hostende Organisation (Administration von Server und Anwendung).
- **Nutzer** sind ausschließlich benannte Personen des Hauses bzw. des Betreiberteams.

---

5. Open Source und Lizenz

`souffleur` ist **quelloffene Software (Open Source)**. Der vollständige Quellcode wird unter der **MIT-Lizenz** — einer der etabliertesten und freizügigsten Open-Source-Lizenzen — **öffentlich** im Git-Repository der **Technologiestiftung Berlin** bereitgestellt. Jede interessierte Einrichtung kann den Code einsehen, herunterladen, selbst betreiben, an die eigene Systemlandschaft anpassen und weitergeben.

5.1 Was die MIT-Lizenz bedeutet

Die MIT-Lizenz erlaubt ausdrücklich die **freie Nutzung, Veränderung und Weiterverbreitung** — für nicht-kommerzielle wie kommerzielle Zwecke — bei nur einer Auflage: Der ursprüngliche Urheberrechts- und Lizenzhinweis muss erhalten bleiben. Es entstehen **keine Lizenzkosten** und keine Bindung an einen bestimmten Anbieter (kein „Vendor Lock-in"). Die Software wird — wie bei Open-Source-Software üblich — „wie besehen", ohne Gewährleistung, bereitgestellt.

5.2 Warum das für die Compliance-Bewertung relevant ist

- **Transparenz und Prüfbarkeit:** Der gesamte Quellcode ist offen einsehbar. Die in dieser Beschreibung gemachten Aussagen (nur lesender Zugriff, keine dauerhafte Datenspeicherung, keine Einbindung fremder Dienste) lassen sich am Code selbst nachprüfen — nichts ist verborgen.

- **Unabhängigkeit:** Da der Code frei verfügbar ist, ist das nutzende Haus nicht von der ursprünglich entwickelnden Organisation abhängig. Betrieb, Wartung und Weiterentwicklung können jederzeit vom Haus selbst oder einem beliebigen Dienstleister übernommen werden.
- **Nachnutzung durch die öffentliche Hand:** Die öffentliche Bereitstellung folgt dem Grundsatz „Public Money, Public Code" — mit öffentlichen Mitteln entwickelte Software steht der Allgemeinheit zur Verfügung. Andere Kultureinrichtungen können das Werkzeug ohne erneuten Entwicklungsaufwand übernehmen.

5.3 Was NICHT im öffentlichen Repository liegt

Öffentlich ist ausschließlich der **Programmcode**. **Keine** Zugangsdaten, Passwörter, API-Tokens, Konfigurationsdateien mit Geheimnissen oder echten Daten sind Teil des Repositorys — diese liegen ausschließlich auf dem jeweiligen Betriebsserver (siehe Abschnitte 3.3 und 4.2) und werden durch die Veröffentlichung in keiner Weise offengelegt.

---

6. Erweiterbarkeit: weitere Schnittstellen nach Bedarf

Das Dashboard ist bewusst als **Baukasten** angelegt: Jede Datenquelle ist ein eigenes, in sich abgeschlossenes Modul (ein „Konnektor"). Die beiden bestehenden Konnektoren (artwork und Pretix) teilen sich das gleiche Grundprinzip:

1. Daten **nur lesend** aus dem Quellsystem holen (Datenbank-Leserecht oder API-Token),
2. in ein einheitliches internes Format übersetzen,
3. mit den übrigen Quellen zusammenführen und anzeigen.

Neue Quellsysteme — etwa ein anderes Ticketing-System, eine Dienstplan-Software, ein Spendentool, ein Kassensystem, eine **WordPress-Website** (z. B. Besucherzahlen und Aufrufe der Veranstaltungsseiten), **Social-Media-Tools** (z. B. Reichweite und Follower-Entwicklung der Kanäle des Hauses) oder ein **Formular-/Umfragetool** (z. B. Nextcloud Forms, etwa für Ergebnisse von Publikumsbefragungen) — lassen sich nach demselben Muster ergänzen, ohne die bestehende Anwendung umzubauen. Für jedes neue System gilt dieselbe Sicherheits-Checkliste: offizielle Schnittstelle, minimale Leserechte, widerrufbarer Zugang, Geheimnisse nur auf dem Server, keine dauerhafte Datenspeicherung.

Dadurch ist die Lösung auf andere Häuser übertragbar, auch wenn diese eine andere Systemlandschaft einsetzen: Ausgetauscht wird nur der jeweilige Konnektor, der Rest des Dashboards bleibt gleich. Auch das Betriebsmodell wählt jedes Haus selbst — gemieteter Server oder eigene Hardware vor Ort (Abschnitt 3.4).

---

7. Glossar

Begriff	Bedeutung
API / REST-API	Eine offizielle „Steckdose” einer Software, über die andere Programme kontrolliert Daten abfragen können — der vom Hersteller vorgesehene Weg.
API-Token / Schlüssel	Eine Art Passwort für die API. Der Konto-Inhaber erzeugt es selbst, bestimmt die Rechte (hier: nur Lesen) und kann es jederzeit sperren.
Basic-Auth	Einfache, bewährte Passwort-Abfrage des Webservers, bevor die Anwendung überhaupt erreichbar ist.
Container (Docker)	Eine abgeschottete Laufzeitumgebung; die Anwendung läuft isoliert vom übrigen Server.
HTTPS	Verschlüsselte Übertragung zwischen Browser und Server (Schloss-Symbol im Browser).
Nur-Lese-Zugriff (SELECT)	Der Datenbank-Benutzer darf Daten ausschließlich ansehen; Ändern oder Löschen ist auf Datenbankebene technisch ausgeschlossen.
Reverse Proxy	Der „Pförtner” des Servers: nimmt alle Anfragen entgegen, sorgt für Verschlüsselung und Zugangskontrolle und leitet sie erst dann an die Anwendung weiter.
SvelteKit / Node.js / TypeScript	Gängige, quelloffene Web-Technologien, mit denen die Anwendung entwickelt ist.