KI-PROGRAMMIEREN MIT BOLT.NEW

Expecto patronum

Aus Prompt mach Programm: Bolt.new schreibt Software mithilfe von KI. Wie gut das schon funktioniert, hat dotnetpro ausprobiert.

Ein bisschen was von Zauberei hat das schon: Ein paar dürre Worte reichen aus, und heraus kommt eine App, die sofort in Aktion treten kann. Das ist die eine Seite der Medaille. Auf der anderen Seite aber stellt sich in so einem Fall die überlebenskritische Frage: Wer braucht dann noch Softwareentwickler?

Ungeachtet dieser Fragestellung gibt es bereits erste Tools, die sich die Kompletterzeugung von Anwendungen auf die Fahnen geschrieben haben. Eines davon heißt Bolt.new [1]. Dabei handelt es sich um ein KI-gestütztes Web-Entwicklungswerkzeug, das es ermöglicht, vollständige Web- und Mobilanwendungen direkt im Browser zu erstellen, zu bearbeiten, auszuführen und bereitzustellen, ohne dazu eine lokale Entwicklungsumgebung einrichten zu müssen.

1. **Project Creation and Purpose**
   - **Command/Instruction:** Start by initiating a new project called "shopty."
   - **Goal:** Develop a collaborative shopping-list application that supports real-time syncing and sharing among multiple users.
2. **User Management and Authentication**
   - **Command/Instruction:** Incorporate an authentication mechanism (email + password, or social sign-on) so each user can have a secure, individual account.
   - **Goal:** Ensure users can log in, manage their profiles (name, email, optional avatar), and have private access to their lists.
3. **Data Models**
   - **Command/Instruction:** Define three primary data entities:
     1. **User** – Holds user information (name, email, avatar).
     2. **ShoppingList** – Contains a title, creation date, and an association with the user(s).
     3. **Item** – Represents an individual product or good with attributes like name, quantity, notes, purchased status, and timestamps.
   - **Goal:** Provide a structured appro to storing and retrieving data for each piece of the shopping experience.

Vorarbeit: Anforderungen, formuliert von ChatGPT (Bild 1)

Ein herausragendes Merkmal von Bolt.new ist die Fähigkeit, durch natürliche Spracheingaben Anwendungen zu erstellen. Nutzer können beispielsweise Anweisungen wie „Erstelle eine E-Commerce-Produktseite mit Filter- und Sortierfunktionen" eingeben, und das Tool generiert daraufhin die entsprechende App. Dieses Vorgehen ermöglicht eine schnelle Prototypenerstellung und erleichtert das Experimentieren mit verschiedenen Technologien und Frameworks.

Bolt.new unterstützt zahlreiche Webtechnologien und Frameworks, darunter React, Vue und Astro. Es unterstützt das Installieren und Ausführen von NPM-Paketen, das Betreiben von Node.js-Servern und die Interaktion mit APIs. Zudem bietet es Integrationen mit Diensten wie Netlify für das Hosting und Supabase für Datenbank- und Authentifizierungsfunktionen. Anwendungen lassen sich direkt aus dem Browser heraus bereitstellen und teilen.

Die KI-gestützte Generierung passt sich in der Regel dem gewählten Framework an. Das heißt: Gibt man zum Beispiel den Wunsch nach einer „Blog-Anwendung mit Routing und Markdown-Unterstützung in Astro" ein, wird genau dieses Setup umgesetzt – inklusive passender Verzeichnisstruktur, Konfiguration und Beispieldateien. Bolt.new achtet dabei auf Konventionen des jeweiligen Frameworks, sodass man sich in den generierten Projekten schnell zurechtfindet.

Auch bei der Wahl der Architektur bleibt man flexibel. Es lassen sich beispielsweise klassische clientseitige Anwendungen umsetzen, aber auch serverseitige Komponenten wie APIs oder Middleware integrieren – Bolt unterstützt Node.js-Prozesse direkt im Browser dank der sogenannten WebContainers-Technologie [2]. Ebenso kann man moderne Architekturkonzepte wie Komponenten-basierte Entwicklung, JAMstack oder sogar erste Headless-Ansätze testen.

Einschränkungen gibt es nur dort, wo Frameworks oder Tools systemnahe Abhängigkeiten erfordern oder sehr spezifische Build-Prozesse nutzen, die im Browser nicht ohne Weiteres ausführbar sind. Für solche Spezialfälle bleibt eine lokale Entwicklungsumgebung weiterhin die robustere Lösung.

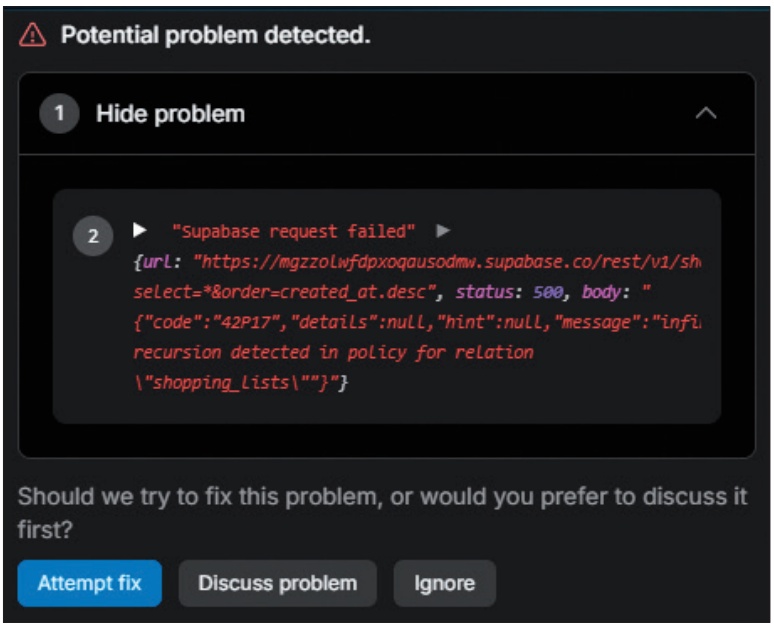
Bolt.new basiert auf der Web-Entwicklungsumgebung StackBlitz [3]. Mit einem Klick lässt sich die Anwendung, die Bolt.new erzeugt hat, in StackBlitz öffnen. Darüber ist auch das Einchecken in ein GitHub-Repository möglich.

Web okay, nativ nicht

In erster Linie ist Bolt.new für das Erstellen von Webanwendungen (Web-Apps) und Websites gedacht. Dementspre-

chend ist die Plattform auf Webtechnologien wie HTML, CSS, JavaScript, TypeScript sowie Frameworks wie React, Vue, Astro, Svelte und weitere ausgelegt. Die generierten Projekte laufen direkt im Browser. Damit eignet sich Bolt.new für klassische Websites, Single-Page-Apps (SPA), Landingpages, Blogs, Dashboards oder kleine Tools. Somit sind auch Web-Apps kein Problem. Etwas eingeschränkter sieht es bei Progressive Web Apps (PWAs) aus: Theoretisch lassen sich mit dem passenden Setup auch PWAs bauen – also Web-Apps, die sich wie native Apps anfühlen und offline funktionieren können. Ob man dabei alles vollständig in Bolt konfigurieren kann, hängt aber vom Framework und dem Setup ab.

Native Mobile-Apps für iOS oder Android mithilfe von Swift, Kotlin oder gar Flutter/React Native sind nicht umsetzbar. Dafür bräuchte man eine spezialisierte Entwicklungsumgebung. Allerdings kann man über Umwege wie Capacitor oder Ionic Web-Apps als hybride Mobile-Apps verpacken. Dafür ist Bolt aber nicht optimiert.



Sackgasse: In voller Fahrt in den Fehler (Bild 2)

Ebenso wenig ist Bolt für klassische Desktop-Apps konzipiert. Zwar ließe sich ein Teil des Codes (wie ein React-UI) dort wiederverwenden, doch ein vollständiges Desktop-Build-Tooling ist nicht integriert.

Los gehts!

Kann Bolt.new die vielen Versprechen halten? Das prüfen wir nun! Die Shopping-List-App Bring! [4] soll als Vorbild dienen: Einer vorher angelegten Liste lassen sich durch Auswahl von Einträgen einer Vorschlagsliste oder durch Freitext neue Einträge hinzufügen beziehungsweise durch Klick auf einen Eintrag wieder daraus entfernen. Bring! verwaltet mehrere Listen. So kann eine Liste für Lebensmittel und eine für Arzneimittel angelegt werden. Außerdem lassen sich mehrere Geräte inklusive Website-Interface mit einer Liste verbinden. Auf diese Weise haben alle Gruppenmitglieder die Möglichkeit, Einträge der Liste zu verwalten.

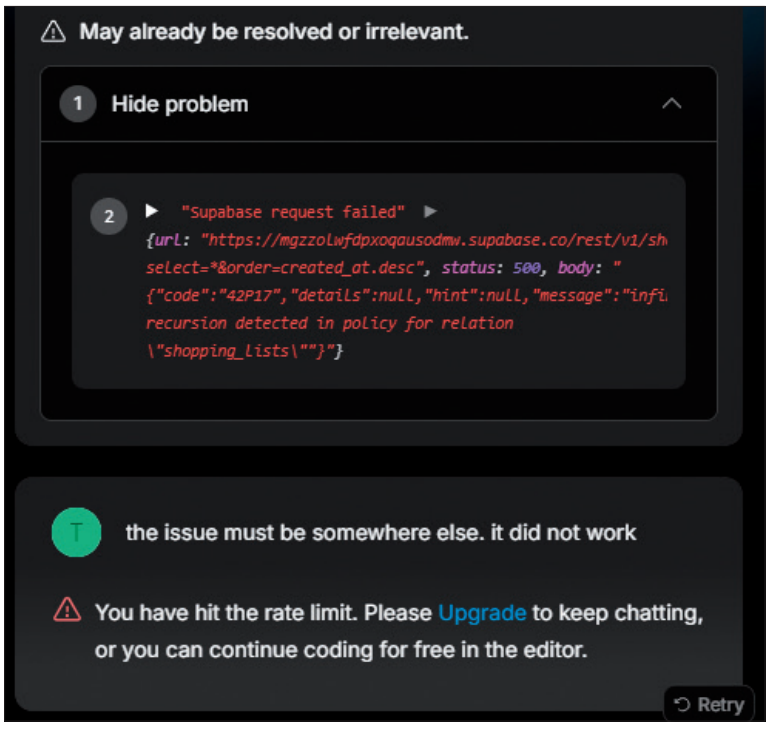
Um eine etwas ausführlichere Beschreibung dessen, was die App leisten soll, zu erhalten, wird das gute alte ChatGPT bemüht. Damit ist recht schnell ein Pflichtenheft entworfen:

```
Take a look at the app Bring! a shopping list
  app. Write a description what this app does
  so that a tool like Bolt.new can program a
  similar app.
```

Als Ergebnis spuckt ChatGPT eine Anforderungsliste aus (Bild 1). Dieses Ergebnis in das Textfeld auf der Bolt.new-Homepage gefüttert führt erst einmal dazu, dass man einen Account anlegen muss. Das Gleiche verlangt das System dann gleich auch noch für StackBlitz und – aller guten Dinge sind drei – auch noch für Supabase, denn diese Datenbank soll alle Daten von Shopping List, wie die Anwendung kreativ benannt wird, speichern.

Danach fängt Bolt.new an, Dutzende Code- und UI-Beschreibungsdateien sowie Tabellen in Supabase anzulegen. Ein paar Minuten später mündet das Ganze dann leider in einen Fehler (Bild 2).

Leider reicht das Wissen des Autors über Supabase und den Code nicht aus, um den Fehler zu fixen. Interessanterweise gilt das aber auch für Bolt.new: Mehrere Versuche, Bolt.new dazu zu bringen, den Fehler zu beheben, schlagen fehl. ▶



Nichts geht heute mehr: Die Frei-Tokens sind aufgebraucht (Bild 3)

Das Tool erkennt den Fehler und versucht ihn zu lösen, ist aber nicht erfolgreich. Der Fehler bleibt, wie er ist. Beim dritten Mal gibt es dann noch die wenig ermutigende Meldung, dass nun die Frei-Tokens aufgebraucht seien (Bild 3).

Neues hui, Altes pfui

Bisher haben wir versucht, in guter alter Hello-World-Manier eine neue Anwendung anzulegen. Hier tun sich alle Generatoren leicht. Keine Altlasten, keine vorhandenen Code-Fragmente, der Generator ist Schöpfer des gesamten Codes.

Wie aber sieht es aus, wenn es schon ein Projekt gibt, das mithilfe von Bolt.new erweitert werden soll? Hierzu gibt es eine klare Aussage: Bolt.new eignet sich vor allem für Greenfield-Projekte, also Neuentwicklungen, bei denen man auf der grünen Wiese beginnt. Bei Brownfield-Projekten – also bestehenden Anwendungen oder Codebasen, die erweitert oder modernisiert werden sollen – stößt die Plattform derzeit an gewisse Grenzen. Zwar lassen sich auch bestehende Repositories in Bolt.new importieren, und man kann mit Webtechnologien wie React, Vue oder Astro direkt im Browser weiterentwickeln. Dennoch ist die Plattform primär auf eine KI-gestützte Generierung von neuen Anwendungen ausgelegt, nicht auf die komplexe Weiterarbeit an bestehenden Projekten mit vielen Abhängigkeiten, Build-Tools oder spezifischen Infrastrukturvorgaben.

Ein häufiges Problem bei Brownfield-Projekten in Bolt.new ist, dass die Entwicklungsumgebung im Browser nicht alle Systemzugriffe und Tools bereitstellt, die in größeren oder älteren Projekten benötigt werden – etwa benutzerdefinierte Build-Skripte, systemnahe Abhängigkeiten oder CI/CD-Integrationen. Auch die Möglichkeiten zur Konfiguration sind aktuell eingeschränkter als in einer klassischen lokalen Entwicklungsumgebung.

Dennoch eignet sich Bolt.new gut, um einzelne Teile eines Brownfield-Projekts zu analysieren, prototypisch neu zu bauen oder umzustrukturieren. Man kann zum Beispiel Komponenten oder neue UI-Elemente mit KI-Hilfe entwickeln und diese dann zurück ins Hauptprojekt integrieren. Auch das Testen von Alternativen (zum Beispiel neue Frameworks oder Architekturen) funktioniert gut in der isolierten Umgebung von Bolt.

Alles hat seinen Preis

Die erste Sackgasse hat gleich aufgezeigt, wo mögliche Probleme bei der Nutzung von Bolt.new liegen können: Die verfügbaren Tokens schmelzen dahin wie Butter in der Sonne. Das gilt insbesondere für den kostenlosen Zugang: Bei diesem erhält der Nutzer täglich laut Website ein Kontingent von 150 000 Tokens und einer Million Tokens im Monat. Klingt nach viel, doch stellt sich als wenig heraus.

Laut Nutzerberichten [5] liegt das Limit aber eher bei etwa 100 000 Tokens pro Tag, wobei jede Anfrage ungefähr 25 000 Tokens verbraucht. Dies ermöglicht es, täglich einige Anfragen zu stellen und kleinere Projekte zu realisieren.

Für umfangreichere Projekte oder häufigere Nutzung braucht es mehr. Hierfür bietet Bolt.new verschiedene Abonnements an [6]:

- Pro: 20 Dollar pro Monat für 10 Millionen Tokens.
- Pro 50: 50 Dollar pro Monat für 26 Millionen Tokens.
- Pro 100: 100 Dollar pro Monat für 55 Millionen Tokens.
- Pro 200: 200 Dollar pro Monat für 120 Millionen Tokens.

Zusätzlich zu den Tokens bieten die kostenpflichtigen Pläne weitere Vorteile, etwa die Möglichkeit, private Projekte zu erstellen. Das Abonnement lässt sich jederzeit kündigen.

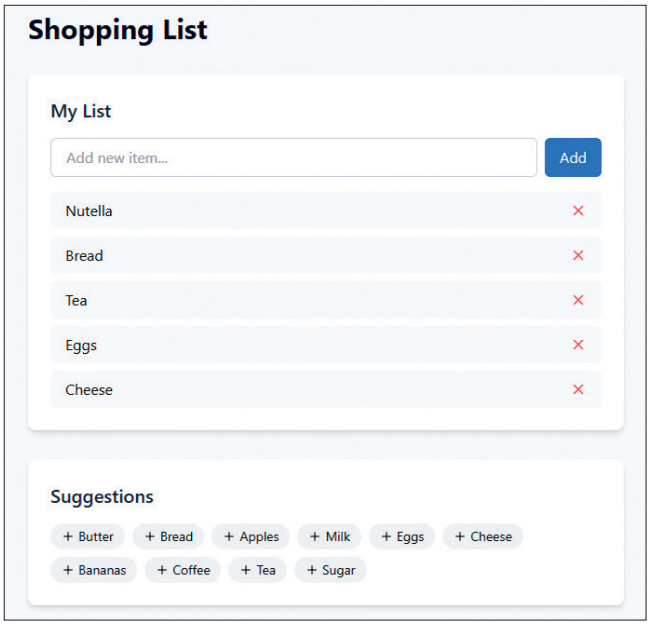
Bei Bolt.new verfallen ungenutzte Tokens aus dem monatlichen Abonnement am Ende des Abrechnungszeitraums und werden nicht in den nächsten Monat übertragen – ein im Netz durchaus kritisierter Modus, da er nicht klar kommuniziert wird.

Anders verhält es sich mit zusätzlich erworbenen Tokens, sogenannten Token Reloads: Diese bleiben über den aktuellen Monat hinaus gültig und können in den Folgemonaten genutzt werden, vorausgesetzt, man verfügt weiterhin über ein aktives kostenpflichtiges Abonnement.

Alles auf Start

Nach dem unbefriedigenden Ende der ersten Session hilft nur eines: Noch einmal von vorne anfangen. Offenbar war die Aufgabe für Bolt.new dann doch zu komplex. Also reduzieren wir die Komplexität und bauen Shopping List Stück für Stück auf. Schlau, wie der Autor ist, holt er sich mit einer anderen E-Mail-Adresse ein frisches Kontingent an Frei-Tokens und beginnt am selben Tag von Neuem:

```
I want to build a shopping list web app. The app holds a
  list of items that should be bought. A second list
  holds proposals. With a click on one proposal the user
  adds the proposal to the shopping list. As alternative
  the user can type text into a textbox and add an item
  to the shopping list that is not in the proposal list.
  In the proposals there are items like „Butter“,
  „Bread“, „Apples“.
```



Respektabel: Die erste Version kann sich sehen lassen (Bild 4)

Mit diesem Prompt kann Bolt.new etwas anfangen, und das Ergebnis sorgt gleich für gute Laune (Bild 4). Das Besondere an Bolt.new ist, dass es die Anwendung nicht nur baut, sondern sie gleich aktiv zur Verfügung stellt. Es lässt sich also unmittelbar prüfen, ob die Version an das herankommt, was gewünscht war.

Und tatsächlich: Es gibt die Vorschlagsliste und das Texteingabefeld. Ein Klick auf einen Eintrag der Vorschlagsliste fügt ihn der Einkaufsliste hinzu – genauso wie das Eingeben eines Eintrags in das Textfeld. Dabei fügt die App nur Einträge hinzu, die noch nicht in der Liste sind, was sehr sinnvoll ist. Über einen Klick auf das rote X lassen sich die Einträge wieder löschen. Auch das Aussehen der App ist ansprechend.

Von diesem Ergebnis ermutigt fordern wir Bolt.new auf, das Ganze doch bitte verteilt zu machen.

```
Very good. Now the app shall be able to
  run on different devices or different
  browsers and it should synchronize the
  list items between them. Can you
  please implement this?
```

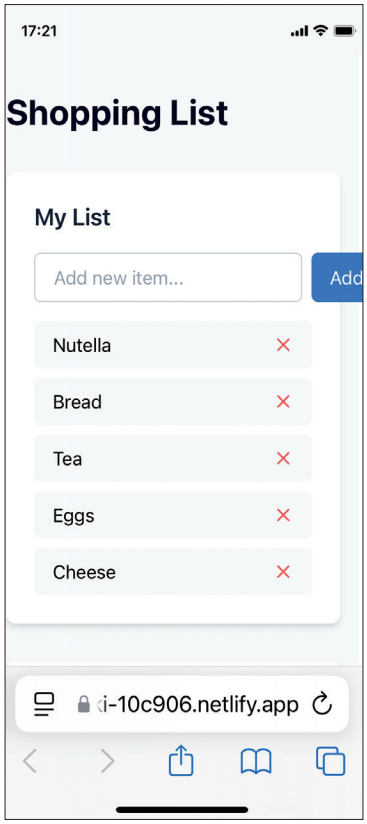
Jetzt bringt Bolt.new Supabase ins Spiel. Über dessen Synchronisationsfähigkeiten soll der Abgleich zwischen den verschiedenen Endgeräten laufen. In Supabase müssen dazu eine Organisation und ein Projekt angelegt werden. Bolt.new übernimmt das, will aber dazu das Okay des Users. Kaum hat man das bestätigt, funktioniert die App nicht mehr, und es tut sich auch nach ein paar Minuten nichts mehr. Wieder eine Sackgasse?

Der Autor probiert *go ahead* als Prompt, und siehe da, es geht weiter. Bolt.new rödelt, und plötzlich enthält die App wieder Einträge und Vorschläge. Aber funktionieren will sie nicht mehr.

Also nachgefragt und nachgehakt und gleich eine Fehlermeldung kassiert. Und dann spannt Bolt.new den Autor als Programmierer ein. Er möge doch bitte mal nachsehen, ob in den Entwicklertools von Chrome nach [F12] irgendwelche Fehlermeldungen in der Konsole auftauchen.

Gehorsam öffnet der Autor die Tools, und plötzlich ist die Funktionalität für das Hinzufügen und Löschen wieder aktiv. Auch lässt sich die App in einem zweiten Tab öffnen, was vorher nicht funktioniert hatte. Fügt man neue Items in einem Tab hinzu, werden sie nach Reload auch im zweiten angezeigt. Leider arbeitet die Synchronisation anfangs nicht, obwohl dafür ein Supabase Channel im Code angelegt wird.

Nach einem kurzen Blick in die Oberfläche von Supabase fällt dem Autor der Knopf *Realtime on* auf. Ein Klick darauf, und schon funktioniert alles: Änderungen im zweiten Browser oder im Browser auf dem Smartphone werden sofort in alle anderen Instanzen synchronisiert. Perfekt schön sieht die App auf dem Smartphone noch nicht aus, aber sie funktioniert wunderbar (Bild 5).



Die Zeit ist abgelaufen: Die im Freikontingent fertiggestellte App auf dem Desktop und auf dem Smartphone (Bild 5)

Dieser Triumph wird allerdings von der Meldung begleitet, dass auch jetzt wieder mein Kontingent an Frei-Tokens aufgebraucht sei. Weitere Anpassungen müssen also bis zum nächsten Tag warten.

Einen Trost gibt es aber: Das Deployment funktioniert trotzdem, und so lässt sich die App per Klick auf Netlify verteilen. Dort funktioniert sie genauso wie in Bolt.new und ist von anderen Geräten aus erreich- und bedienbar.

Im Team ist es schöner

Bolt.new bietet Teams die Möglichkeit, gemeinsam an Projekten zu arbeiten und ihre Entwicklungsprozesse zu optimieren. Hierfür stellt die Plattform spezielle Team-Angebote bereit, die auf die Bedürfnisse von Arbeitsgruppen zugeschnitten sind.

Um ein neues Team zu erstellen, können Administratoren auf StackBlitz unter [7] ein Team anlegen oder über die Option *Select Account* die Funktion *Create a team* wählen.

Wichtig zu beachten ist, dass es für Teams keinen kostenlosen Zugang gibt. Die Kosten für Team-Pläne richten sich nach der Anzahl der Mitglieder. Jedes Mitglied erhält ein eigenes monatliches Token-Kontingent, das nicht mit anderen geteilt wird. Administratoren können Abonnements verwalten, indem sie im Seitenmenü unter *My Subscription* das entsprechende Team auswählen und den gewünschten Plan buchen. Zahlungen werden über Stripe abgewickelt. Nutzer können zwischen ihrem persönlichen Konto und Team-Konten wechseln, indem sie im Seitenmenü die Option *Select Account* nutzen. Dies ermöglicht eine klare Trennung zwischen individuellen und Team-Projekten.

Ein paar Tage später

Ein paar Tage später begrüßt mich Bolt.new mit der Meldung, ich möge die Änderungen, die Bolt.new in Supabase durchführen möchte, bestätigen oder verwerfen. Bestätigen führt zur Fehlermeldung, dass es die Tabelle *suggestions* schon gebe. Der Versuch, das Tool diesen Fehler beheben zu lassen, klappt nicht. Dieser Fehler poppt auch im weiteren Verlauf immer wieder auf. Woran das liegt, wird nicht klar. Die SQL-Anweisungen für das Anlegen der Tabellen enthalten beide das *IF NOT EXISTS*.

Leider ist es damit auch nicht möglich, die ursprünglich englischen Vorschläge durch deutsche zu ersetzen. Denn Bolt.new bietet die Datenbankoperationen nur im Paket an. Die fehlerhafte Operation lässt sich nicht überspringen.

Die Option *Discuss Problem* führt schließlich zu einer Lösung. Hier reflektiert Bolt.new über das Problem und teilt ▶

mir mit, dass es nicht berücksichtigt habe, dass die Tabelle und die Rechte unter Umständen schon bestünden. Nach einem erneuten *Apply* sind alle Vorschläge in deutscher Sprache.

Mutig geworden sollen alle Einträge einer Kategorie hinzugefügt werden. *Butter* wird also beispielsweise *Kühltheke* zugeordnet. Das klappt sehr gut: Bolt.new fügt selbstständig sogar die Möglichkeit hinzu, auch per Freitext eingegebene Einträge den vorhandenen Kategorien zuordnen zu können.

Bolt.new gibt nach solchen Prompts immer den Plan aus, den es für die Änderungen ausgearbeitet hat (Bild 6).

Schließlich sollen die Vorschläge alphabetisch geordnet gelistet werden. Die jeweilige Kategorie soll in Klammern dahinterstehen. Auch das erledigt Bolt.new klaglos.

Figma to Code

Auch wenn noch nicht alles reibungslos verläuft, baut der Hersteller StackBlitz neue Features in die Plattform ein. So ist inzwischen Figma to Code hinzugekommen. Es ermöglicht, Designs aus Figma [8] in funktionsfähigen Code zu überführen. Diese Integration wird durch die Zusammenarbeit mit Anima realisiert, einem Tool, das Figma-Designs in HTML, React und Vue umwandeln kann.

Mithilfe dieses Features könnte ein Screen Designer das UI gestalten und der Entwickler es per Bolt.new mit Funktionalität hinterlegen. Das Vorgehen sieht so aus:

- In Figma den gewünschten Frame auswählen, der in Code umgewandelt werden soll.
- Link kopieren: Rechtsklick auf den Frame und Auswahl von *Copy/Paste as* | *Copy link to selection*.
- Auf der Startseite von Bolt.new auf *Import from Figma* klicken, den kopierten Figma-Link einfügen und *Import Figma frame into Bolt* auswählen.

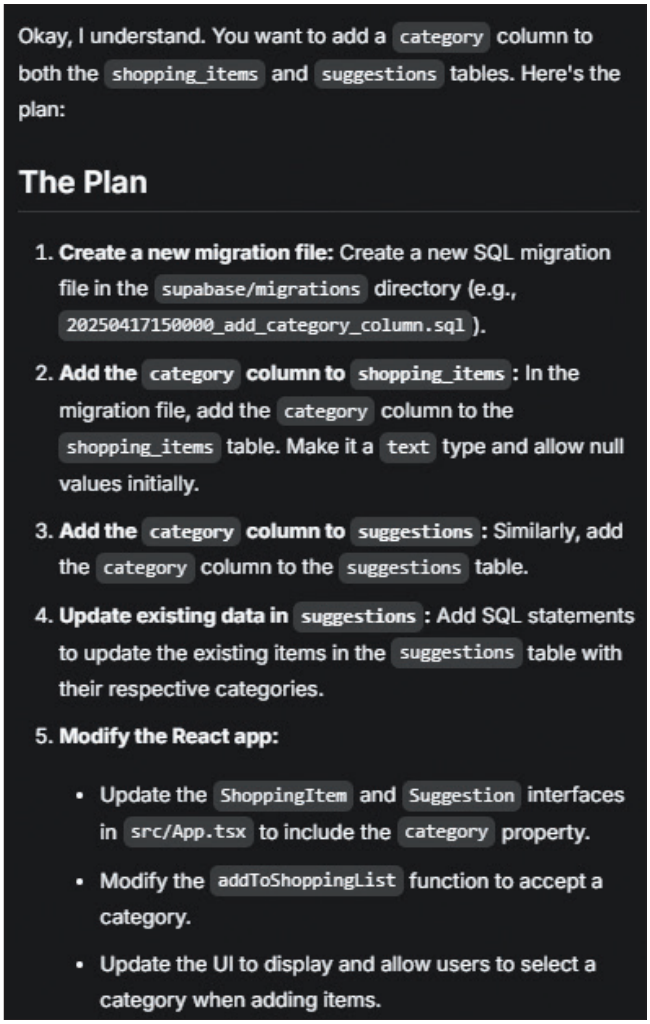
Anschließend generiert Bolt.new mithilfe von Anima den entsprechenden Code für das Design. Für optimale Ergebnisse empfiehlt es sich, in Figma Auto-Layouts und Anima Breakpoints zu nutzen, um ein responsives Design sicherzustellen. Detaillierte Anleitungen dazu bietet Anima in ihren Best Practices unter [9].

Nutzerberichte und Testvideos zeigen, dass das Tool insbesondere bei gut strukturierten und einfachen Designs überzeugende Ergebnisse liefert. Allerdings gibt es in der Community Hinweise darauf, dass bei komplexeren Designs oder weniger strukturierten Figma-Dateien die Ergebnisse nicht optimal ausfallen. Nachdem der Zeitbedarf für das erste Programm schon recht groß war, haben wir dieses Feature nicht getestet.

Shopping List, die dritte

Im nächsten Schritt soll eine weitere Seite eingefügt werden, über die man die Kategorien editieren kann, also neue Kategorien hinzufügen, vorhandene löschen, besonders aber die Sortierung der Kategorien manuell anpassen kann.

Nach dem Befehl gibt es die zusätzliche Seite für die Kategorienverwaltung. Die geforderten Features funktionieren. Allerdings speichert die Anwendung die Änderungen nicht, weshalb dieses Feature dann noch nutzlos ist – und auch erst



Informativ: Bolt.new gibt seinen Plan für die Veränderung preis (Bild 6)

mal bleiben wird, denn wieder stoße ich an das Tageslimit. Es ist schon enorm, wie viele Tokens man verbrät, während man sich an das Ziel herantastet.

Lokal oder remote

Wer sich vom Tageslimit genervt fühlt, kann sich entweder einen bezahlten Account bei Bolt.new holen, oder er verwendet Bolt.diy zusammen mit seinem eigenen Account bei einem der LLMs.

Bei Bolt.diy [10] handelt es sich um die offizielle Open-Source-Version von Bolt.new. Der Hauptunterschied zwischen beiden besteht darin, dass Bolt.diy es den Nutzern ermöglicht, das gewünschte Large Language Model (LLM) für jede Anfrage frei auszuwählen. Aktuell werden Modelle von OpenAI, Anthropic, Ollama, OpenRouter, Gemini, LMStudio, Mistral, xAI, HuggingFace, DeepSeek und Groq unterstützt. Zudem ist die Architektur von Bolt.diy erweiterbar, sodass weitere Modelle integriert werden können.

Bolt.diy wurde ursprünglich von Cole Medin initiiert und hat sich rasch zu einem umfangreichen Community-Projekt entwickelt, das darauf abzielt, den besten Open-Source-KI-Coding-Assistenten bereitzustellen. Die Beziehung zwischen

Bolt.new und Bolt.diy ist somit die einer kommerziellen Plattform und ihrer Open-Source-Variante.

Der Mitbewerb: Cursor, Lovable und v0

Bolt.new, Cursor, v0 und Lovable sind allesamt KI-gestützte Entwicklungswerkzeuge, die unterschiedliche Schwerpunkte und Funktionen bieten.

Cursor ist ein auf Visual Studio Code basierender Editor, der erweiterte KI-Funktionen integriert. Er richtet sich an erfahrene Entwickler und bietet eine anpassbare Umgebung für das Schreiben und Debuggen von Code. Cursor unterstützt die Integration mit bestehenden Codebasen und verschiedenen Frameworks, was diesen Editor besonders für komplexe Projekte geeignet macht. Allerdings erfordert die Nutzung von Cursor eine gewisse Einarbeitung und Programmierkenntnisse.

v0 konzentriert sich auf die schnelle Erstellung von Frontend-Komponenten und nutzt dabei die Framework-unabhängige Komponentenbibliothek shadcn. Es eignet sich besonders für die schnelle Prototypenerstellung von Benutzeroberflächen. Neuere Updates haben v0 um Full-Stack-Funktionen erweitert, wodurch es nun auch Backend-Logik und Datenbankintegration unterstützt. Dennoch verbleibt der Schwerpunkt auf der Frontend-Entwicklung.

Lovable.dev ist eine KI-gestützte No-Code-Plattform, die es Nutzern ermöglicht, Webanwendungen durch einfache Sprachbeschreibungen zu erstellen. Die Plattform kombiniert KI-Technologie mit einer intuitiven Benutzeroberfläche, um komplexe Coding-Schritte zu automatisieren.

Und was ist mit Deployment?

Das Deployment einer Anwendung ist direkt über Bolt.new möglich. Das Tool bietet nach dem Generieren des Projekts einen Live-Vorschau-URL an. Dieser ist öffentlich teilbar, eignet sich jedoch vor allem für Tests oder Präsentationen – nicht für den produktiven Einsatz.

Wer die Anwendung produktiv einsetzen will, kann sich jederzeit den Code herunterladen. Hierzu gibt es einen eigenen Punkt im Menü *Export*. Damit lässt sich die Anwendung dann beispielsweise auf dem eigenen Server hosten.

Der Wunschpartner für Bolt.new ist allerdings Netlify. Neben Vercel ist das eine der populärsten Hosting-Plattformen für Webanwendungen. Wer in Bold.new auf den nicht übersehbaren Knopf *Deploy* drückt, schickt sein Projekt dorthin und erhält einen Link zurück, über den die Anwendung sofort benutzbar ist. Kosten muss man dafür nicht fürchten: Es gibt einen kostenlosen Account für einen Single User, 100 GB Bandbreite pro Monat und 300 Build-Minuten pro Monat.

Fazit

Ganz kann es Shopping List nicht mit Bring! aufnehmen. Dafür müsste man noch einige Tage investieren. Aber das Ergebnis ist bislang beachtenswert.

Kurz gesagt: Bolt.new ist beeindruckend. Wie aus dürren Wörtern eine Anwendung wächst, für die ich mit althergebrachter Methode wesentlich, wesentlich länger gebraucht hätte, ist schlicht toll. Schrittweise nähert man sich der gewünschten App an. Fehlt noch etwas? Kein Problem: Ein weiterer kleiner Befehl, und Bolt.new fügt es hinzu. Die Entwicklerin oder der Entwickler muss sich nicht darum kümmern, in welcher Datei irgendetwas liegt oder wie der Code zwischen CSS und HTML und so weiter aufgeteilt werden muss.

Über die Verbindung mit StackBlitz ist theoretisch auch die Anbindung an ein Repository auf GitHub möglich. Leider gab es im Test hierbei Probleme. Was ist, wenn Bolt.new durch die Änderung die App unbrauchbar macht?

Im Test ist das aber nicht passiert. Auch wenn insbesondere die Supabase-Operationen immer wieder zu Fehlern führten, funktionierte zum Schluss alles wie gewünscht.

Für die neue kleine Webanwendung, die eben mal schnell entstehen soll, ist Bolt.new genau das Richtige. Anders sieht es bei Brownfield-Szenarien aus. Dafür ist das Tool derzeit noch nicht gemacht.

Allen Entwicklerinnen und Entwicklern sei auf jeden Fall dringend geraten, sich Bolt.new anzusehen. Was damit auf welche Weise möglich ist, ist beeindruckend und gibt einen Ausblick darauf, wie künftig Softwareentwicklung oder besser: Softwarebeschreibung aussehen wird. Bolt.new kommt dem tatsächlichen No-Code-Ansatz schon sehr nahe. Hier entstehen nicht nur irgendwelche Formulare, die zusammengeklickt werden, sondern komplette Anwendungen mit Datenbankanbindung und Logik.

Das läuft zwar alles noch nicht hundertprozentig rund, aber die Basis ist wie oben geschrieben schon beeindruckend und ist ein bisschen, aber nicht ganz anders als Zauberei.

Eines bleibt auf jeden Fall: Agiles Vorgehen wird wichtiger denn je. Die Situation, in der der Software Engineer live und on the fly mit dem Kunden die Anwendung inkrementell aufbaut, ist nicht mehr weit weg. ■

[1] Bolt.new, https://Bolt.new
[2] WebContainers, https://webcontainers.io
[3] StackBlitz, https://stackblitz.com
[4] Bring!, www.getbring.com
[5] Limits von Bolt.new free, www.dotnetpro.de/SL2506-07Bolt1
[6] Preise von Bolt.new, https://Bolt.new/?showPricing
[7] Teamverwaltung in Bolt.new, www.dotnetpro.de/SL2506-07Bolt2
[8] Figma, www.figma.com/de-de/
[9] Bolt.new Figma to Code Review – Is It REALLY That Good? (Honest Test), www.dotnetpro.de/SL2506-07Bolt3
[10] Bolt.diy, www.dotnetpro.de/SL2506-07Bolt4

Tilman Börner
war 20 Jahre Chefredakteur der dotnetpro. Heute unterstützt er das Team von Developer Education bei der Ebner Media Group. Daneben bringt er den Kolleginnen und Kollegen den Einsatz der KI nahe.
tilman.boerner@ebnermedia.de

dnpCode A2506-07Bolt