

Pepper der Roboter

28.08.2022

Demo

PandaSuite installieren

Ein Account pro Gruppe:

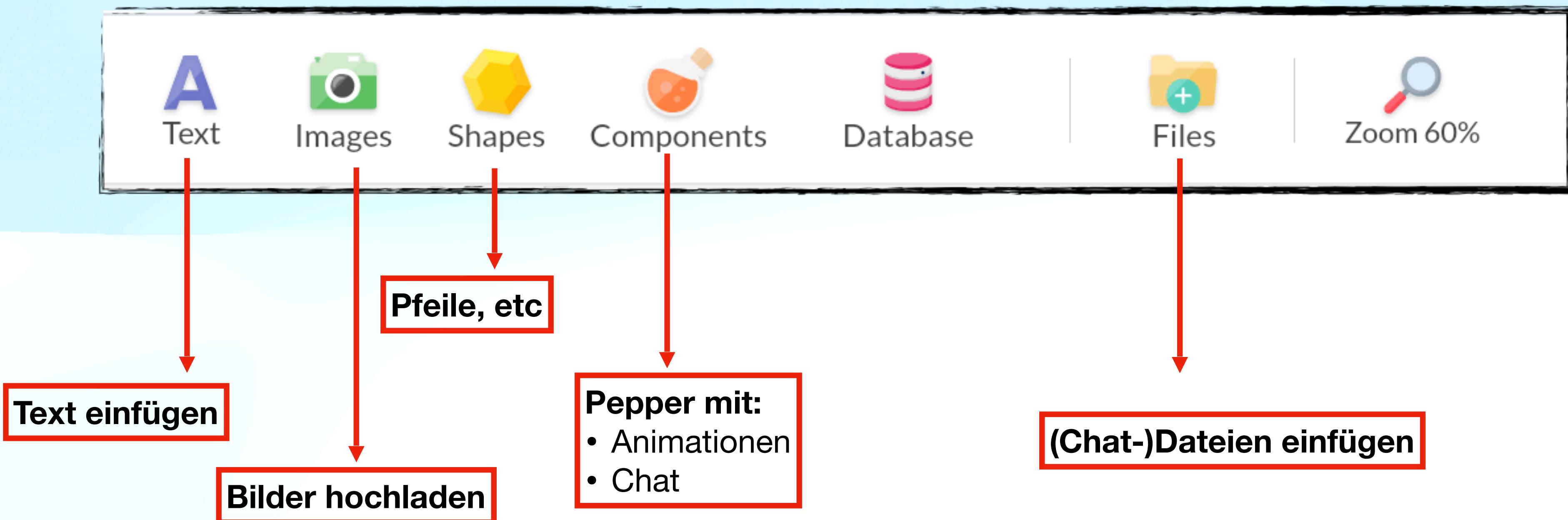
<https://pandasuite.com/dashboard/register>

Hier das Programm runterladen:

<https://pandasuite.com/dashboard/download-studio>

Interface

Gestalterische Komponenten



Interface Backgrounds



Interface Foregrounds



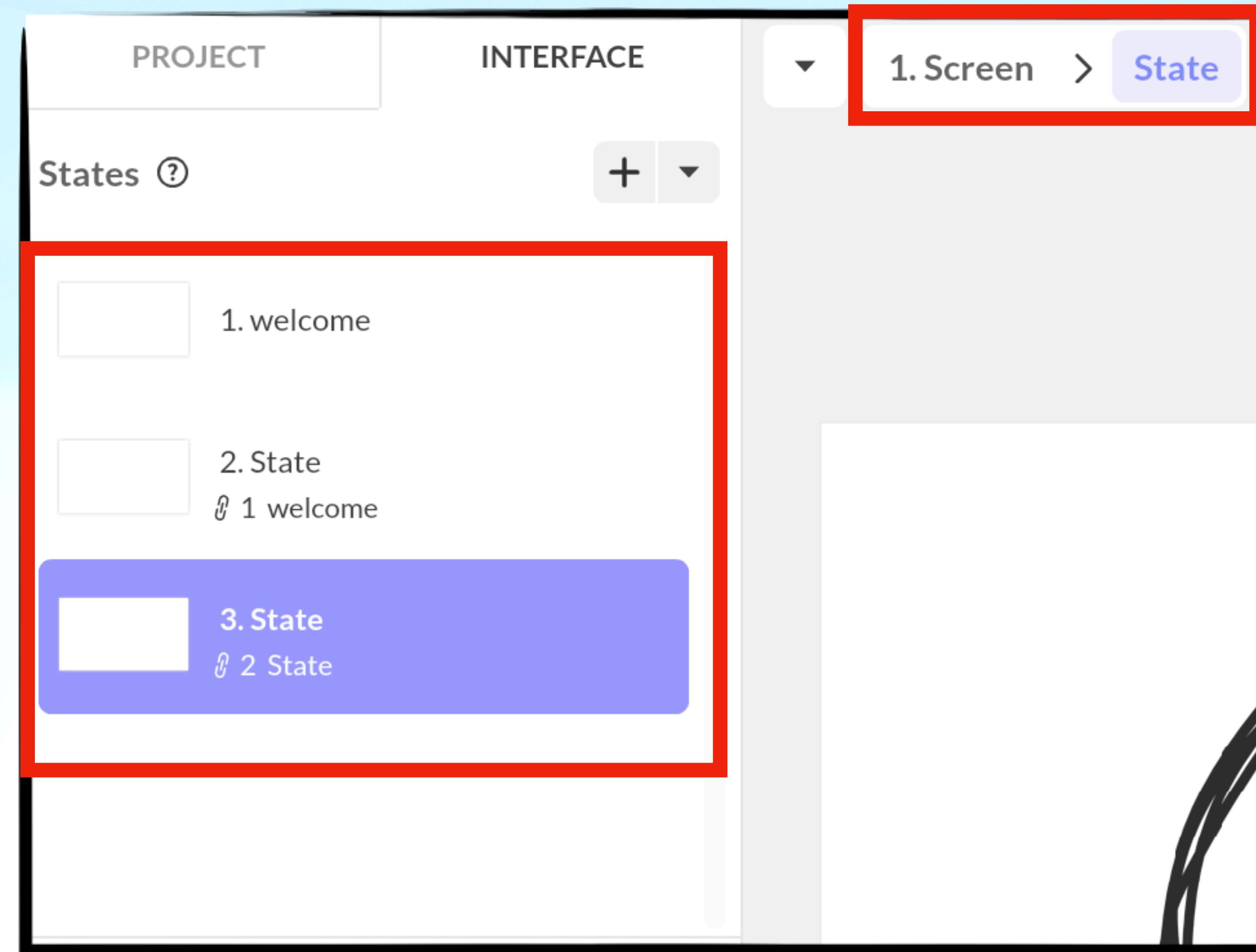
Interface Screens

The image shows a user interface design application with the following interface elements:

- Top Bar:** PROJECT, INTERFACE, 1. Screen (selected), PROPERTIES, ACTIONS.
- Screens Panel:** Screens (3). A list of screens is shown, with the first screen highlighted by a red border. The screen has a purple header and a white body, labeled "1. Screen".
- Main Area:** A large speech bubble containing the text "Willkommen!" in blue. The speech bubble is outlined in black and has a red border.
- Properties Panel:** Styles, Color (Add), Screen (Dynamic), State (Create / Edit, No default), Foreground (Current), Background (1. Background).
- Bottom Left:** Tools for selection, zoom, and text input, with the message "No object here".

Interface

Screens können in States gegliedert werden



Interface

Übergänge zwischen verschiedene States

The screenshot shows a software interface for managing project states. On the left, the 'PROJECT' tab is selected, displaying a list of states:

- 1. welcome
- 2. State** (highlighted in purple, containing '1 welcome')
- 3. State (containing '2 State')

The main area shows a large black silhouette of a head with the word "Willkommen" written inside. The 'INTERFACE' tab is selected, showing the current screen: "1. Screen > State".

The 'ACTIONS' panel on the right contains two actions:

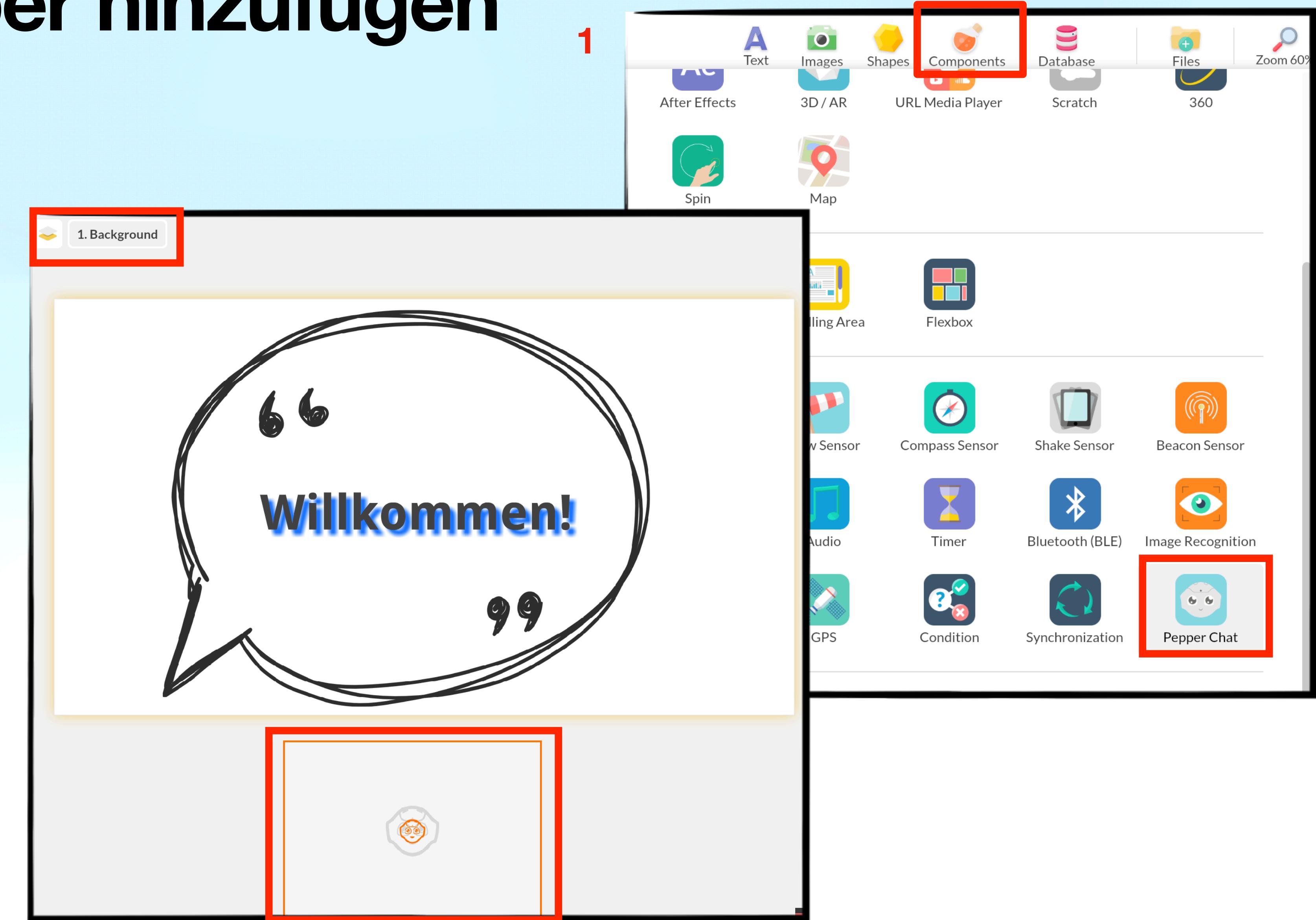
- Action 1 (highlighted with a red box):
 - DISPLAY Current state (State)
 - Next stateProperties:
 - Screen: Current screen
 - Delay: 0.4 sec.
 - Duration: 0.4 sec.
 - Animation: Ease Out
 - Interval: 1 by 1
 - Loop mode: Yes
- Action 2 (highlighted with a red box):
 - Say (Pepper-Chat11716)
 - Go to state: Next

Red numbers 1, 2, and 3 are overlaid on the interface to indicate specific elements:

- 1: Points to the 'Actions' section of Action 1.
- 2: Points to the 'Go to state' action in Action 1.
- 3: Points to the 'Actions' section of Action 2.

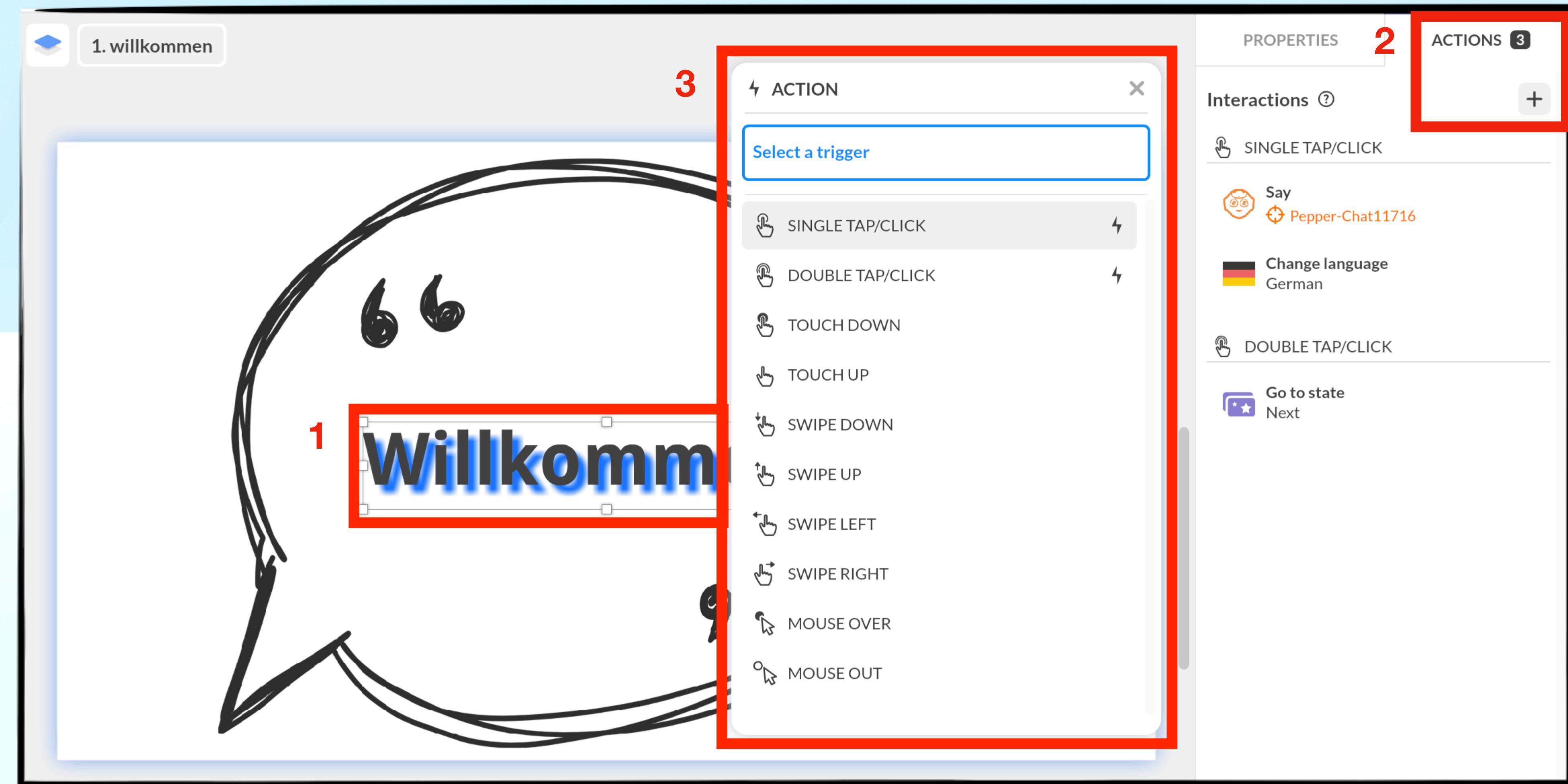
Interface: Pepper hinzufügen

- Wähle aus der Kategorie “Components” den “Pepper Chat” aus
- Ziehen diesen auf deinem Hintergrund



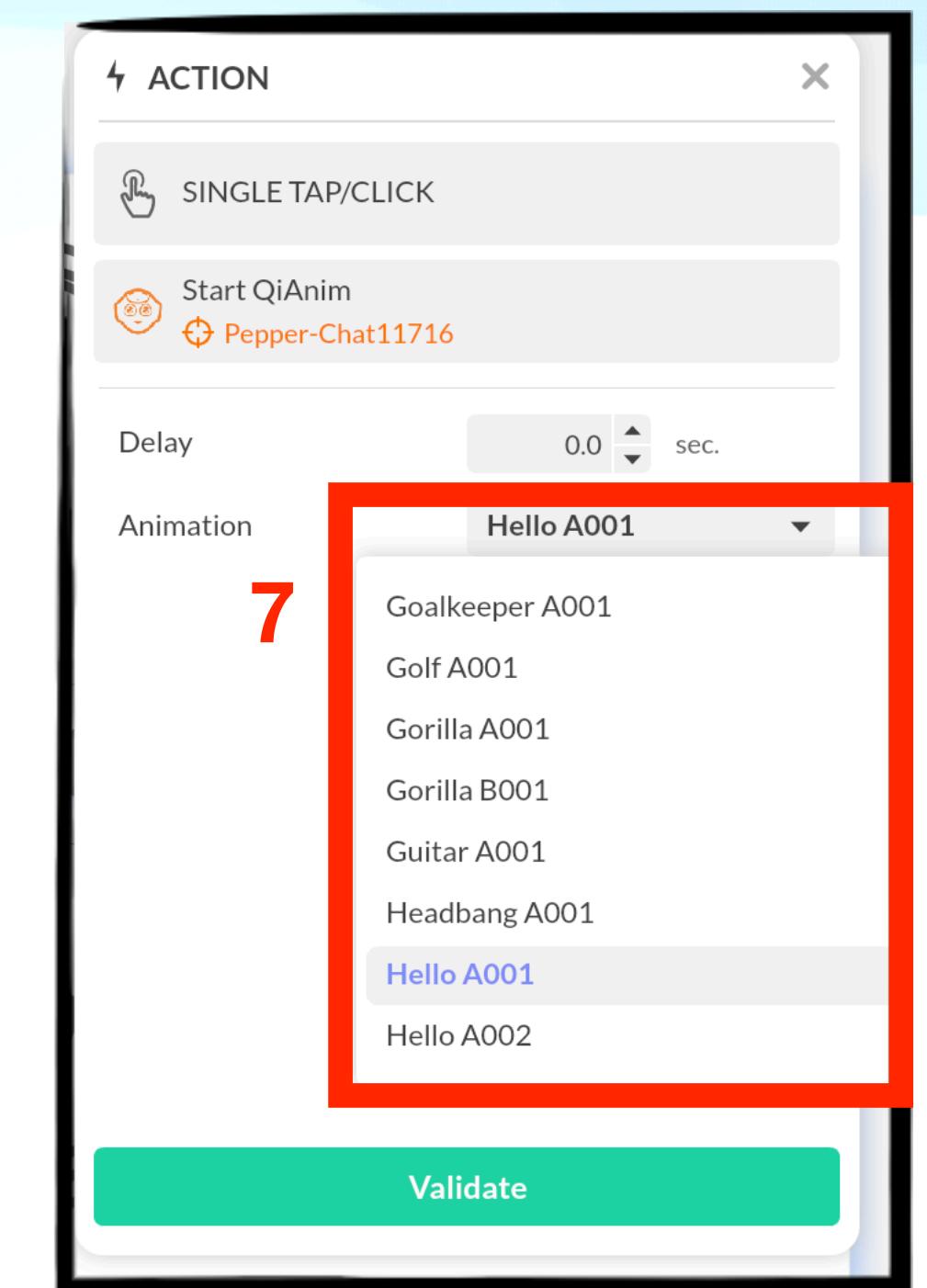
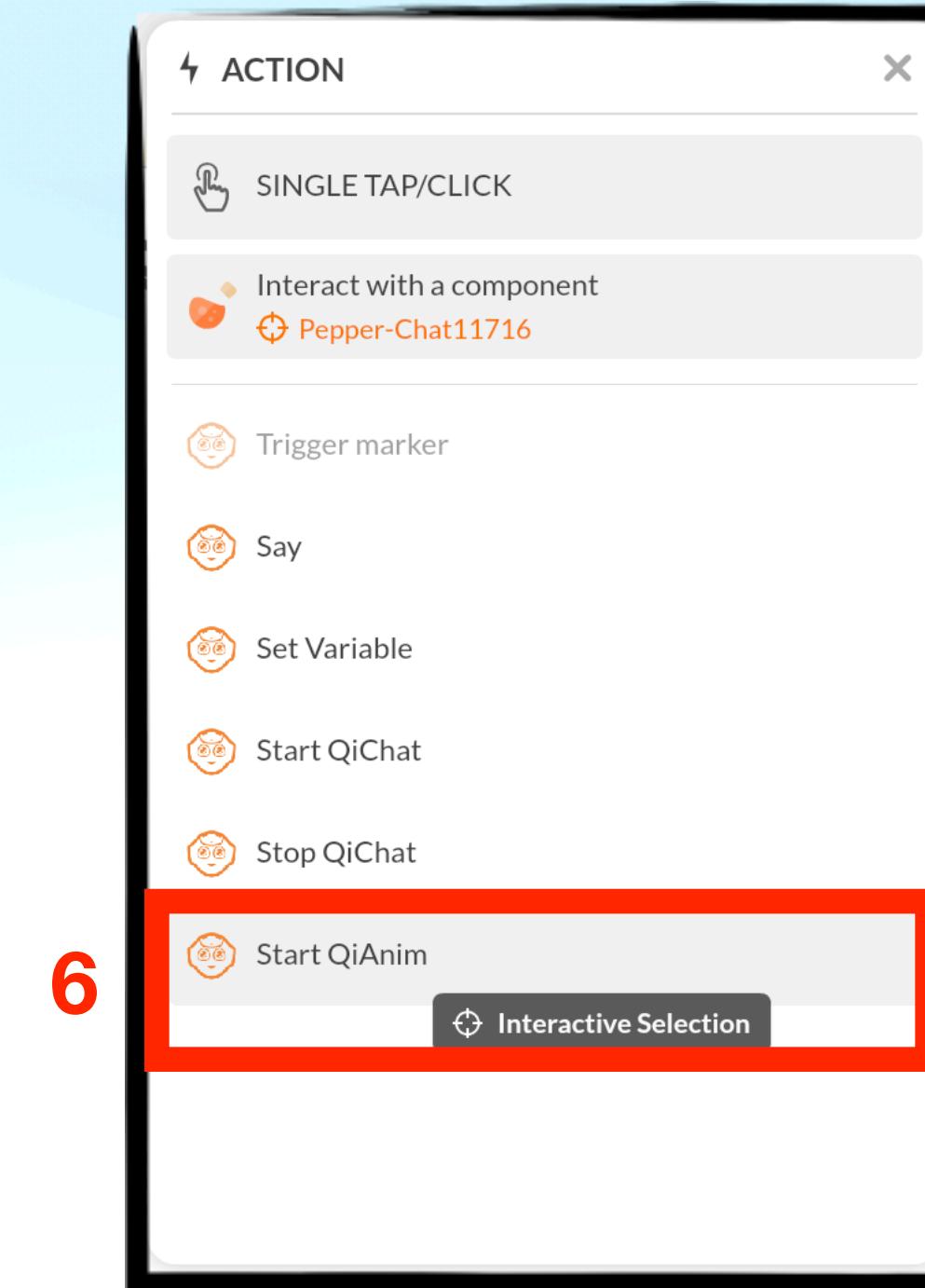
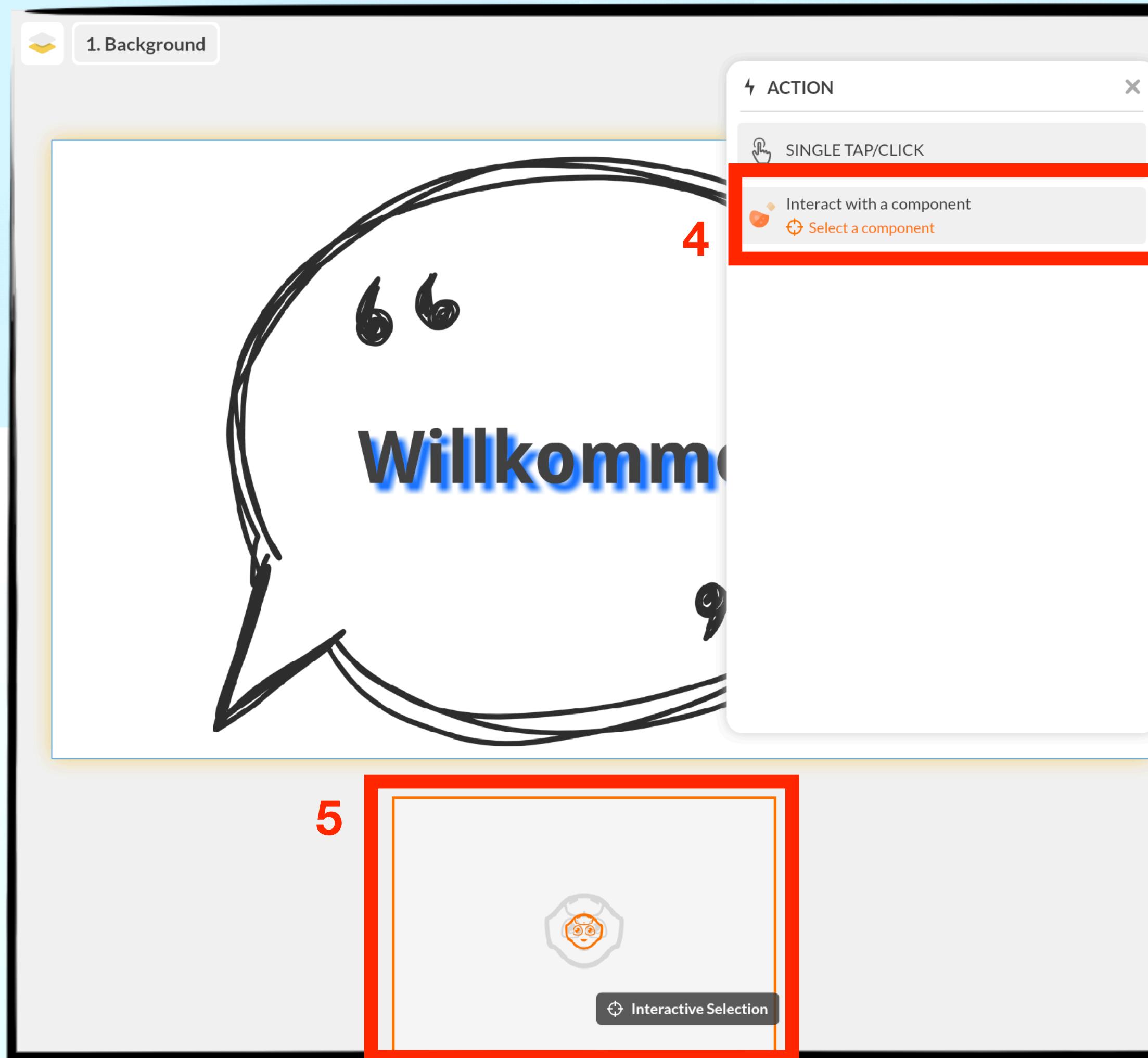
Interface

Pepper gestikulieren lassen (1)



Interface

Pepper gestikulieren lassen (2)



Interface Sprache ändern

PROJECT INTERFACE Project PROPERTIES ACTIONS

Layout ?

Desktop 1366 x 768

August 9, 2022 at 9:52:16 AM

No object here

Willkommen!

Background ... #000000

Responsive Yes No

Magazine m... Yes No

Presentation... Yes No

Background ... Yes No

Keep Screen ... Yes No

Selectable te... Yes No

System UI Hidden

Languages

+ Add

English

German [initial]

Modules

+ Add

Project ?

Background ... #000000

Responsive Yes No

Magazine m... Yes No

Presentation... Yes No

Background ... Yes No

Keep Screen ... Yes No

Selectable te... Yes No

System UI Hidden

Languages

+ Add

English

German [initial]

Modules

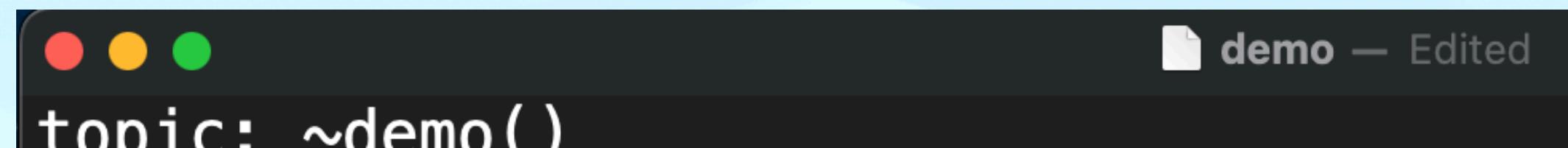
+ Add

Interaktion: Chat

Datei erstellen (topic)

- Wir brauchen eine **Topic-Datei**, also mit Endung **.top**
- Die **erste Zeile** soll aussehen wie im Bild:

topic: ~<NAME>()



A screenshot of a terminal window titled "demo — Edited". The window contains the text "topic: ~demo()". The terminal has a dark background with red, yellow, and green window control buttons at the top.

```
topic: ~demo()
```

Interaktion: Chat

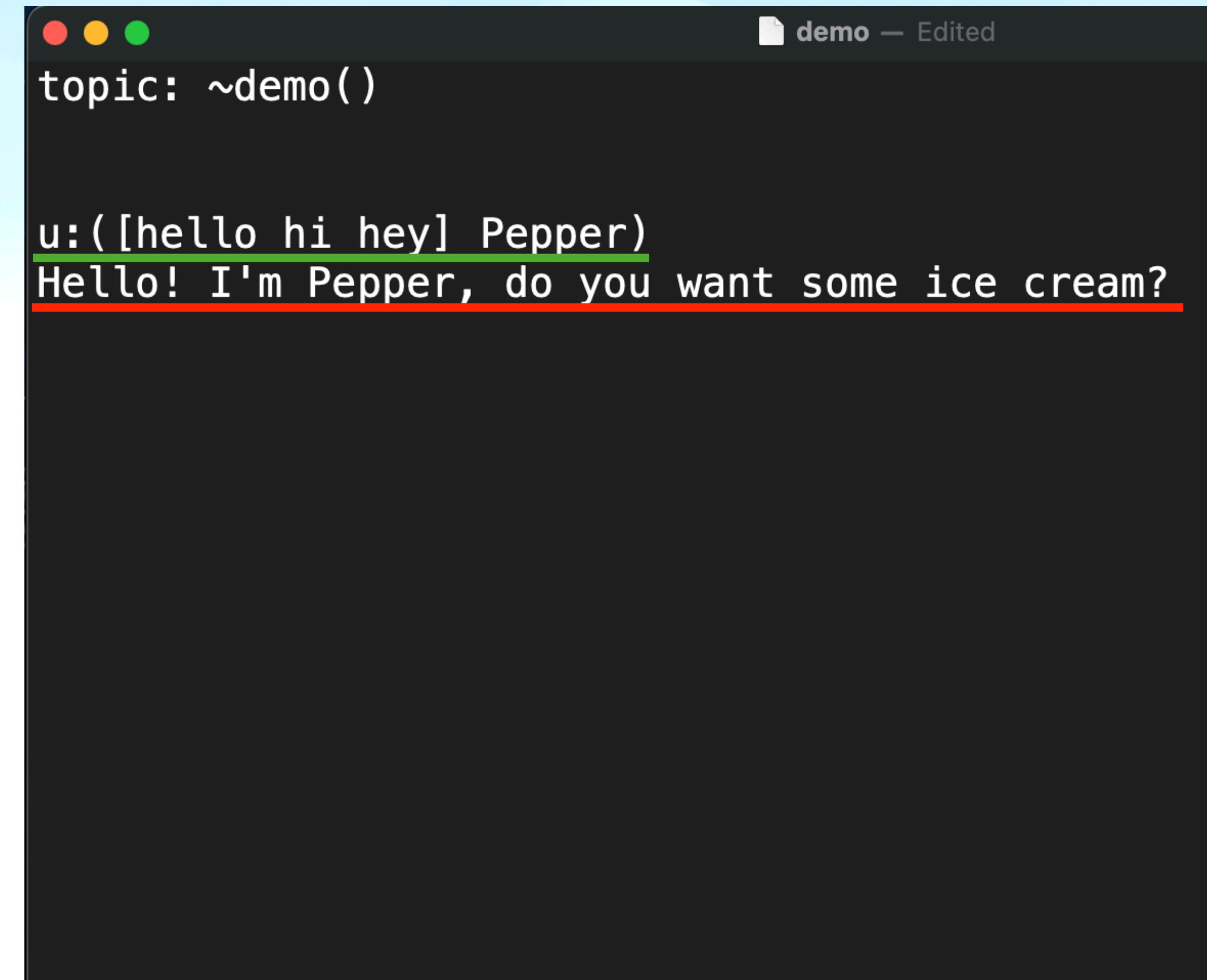
Dialog (rules)

- **u:** steht für die Eingabe, was du sprichst (wie grün unterstrichen im Bild)

[**a1 a2 ...**] ist die Syntax für Synonyme bzw. alternative Wörter!

- Darauf antwortet dann Pepper (wie rot unterstrichen im Bild).

Peppers Antworten sind immer festgeschrieben, sie enthalten keine Alternativen!



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are three colored dots (red, yellow, green) followed by the text "topic: ~demo()". In the main area, the user input "u:([hello hi hey] Pepper)" is shown in green, underlined. Pepper's response "Hello! I'm Pepper, do you want some ice cream?" is shown in red, underlined. The terminal window has a title bar at the top right that says "demo — Edited".

Interaktion: Chat Dialog (proposals)

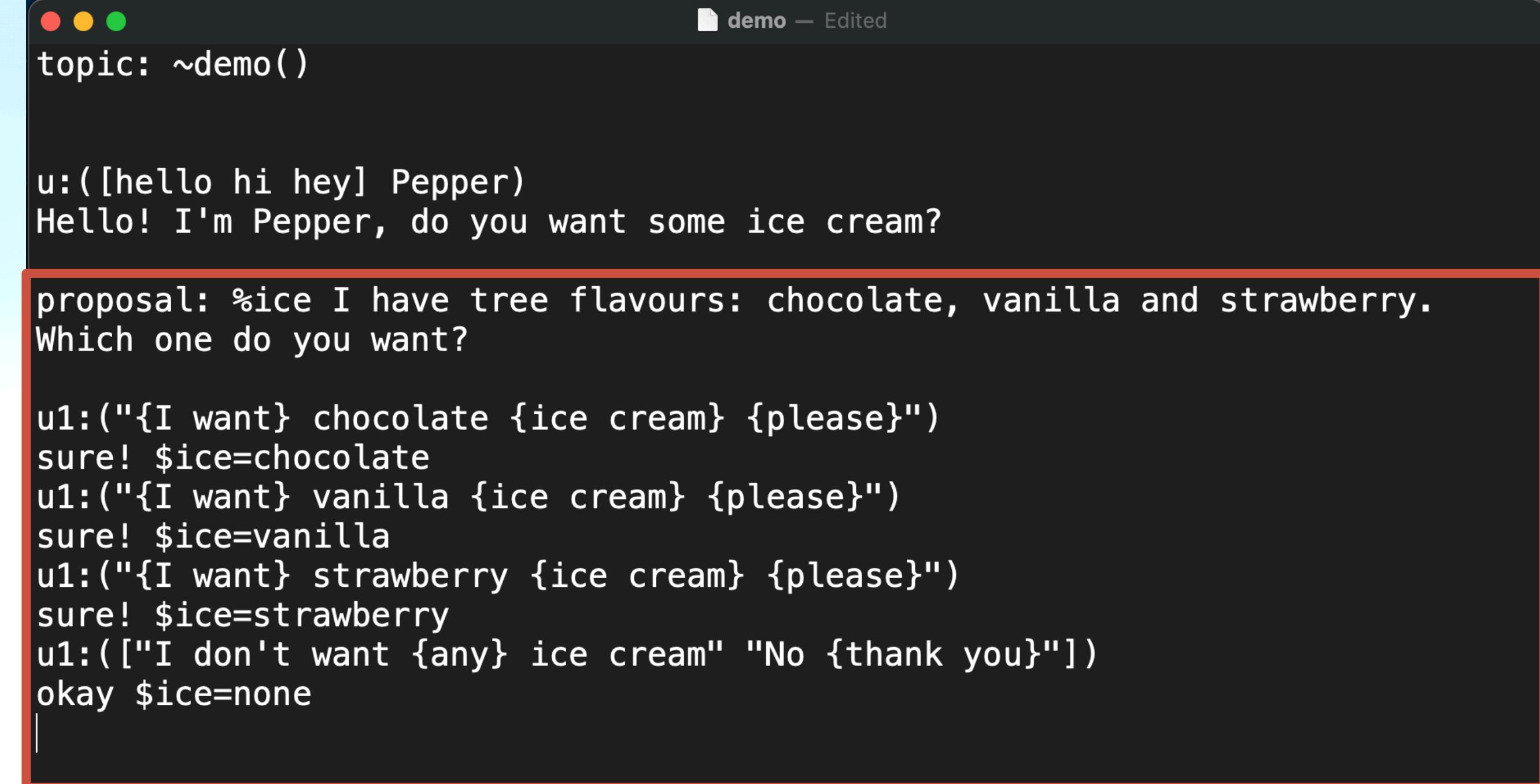
- Ein **proposal** ist eine Aussage die Pepper dir stellt

Diese wird mit %<tag> markiert

- Darauf kannst du antworten

Es kann mehrere alternative Antworten geben: Kennzeichne diese mit %<tag>=wert

{op} ist die Syntax für optionale Satzteile!



```
topic: ~demo()

u:([hello hi hey] Pepper)
Hello! I'm Pepper, do you want some ice cream?

proposal: %ice I have tree flavours: chocolate, vanilla and strawberry.
Which one do you want?

u1:"{I want} chocolate {ice cream} {please}")
sure! $ice=chocolate
u1:"{I want} vanilla {ice cream} {please}")
sure! $ice=vanilla
u1:"{I want} strawberry {ice cream} {please}")
sure! $ice=strawberry
u1:(["I don't want {any} ice cream" "No {thank you}"])
okay $ice=None
```

Interaktion: Chat

Dialog (GoTo)

- Mit **enableThenGoTo(...)** leitet man auf die nächste Aussage weiter

Der Befehl hierfür lautet :

^enableThenGoTo(<tag>)

```
topic: ~demo()

u:([hello hi hey] Pepper)
Hello! I'm Pepper, do you want some ice cream?

proposal: %ice I have tree flavours: chocolate, vanilla and strawberry.
Which one do you want?

u1:"{I want} chocolate {ice cream} {please}")
sure! $ice=chocolate ^enableThenGoto(serve)
u1:"{I want} vanilla {ice cream} {please}")
sure! $ice=vanilla ^enableThenGoto(serve)
u1:"{I want} strawberry {ice cream} {please}")
sure! $ice=strawberry ^enableThenGoto(serve)
u1:(["I don't want {any} ice cream" "No {thank you}")]
okay $ice=none ^enableThenGoto(name)

proposal: %serve Here comes your ice cream!
u1:(["thank you" "thanks"])
You are welcome! $serve=finished ^enableThenGoto(name)
```

Interaktion: Chat

Dialog (Information speichern)

- Damit Pepper eine Antwort wiederholen kann muss man Information speichern.

Mit _<...> deutet man an, was gespeichert werden soll.

Hier steht * für jede Art von Antwort. Aber es geht auch genauer!

\$1 ist die zuerst gespeicherte Antwort zum wiedergeben.

```
Hello! I'm Pepper, do you want some ice cream?  
proposal: %ice I have three flavours: chocolate, vanilla and strawberry.  
Which one do you want?  
u1:"{I want} chocolate {ice cream} {please}"  
sure! $ice=chocolate ^enableThenGoto(serve)  
u1:"{I want} vanilla {ice cream} {please}"  
sure! $ice=vanilla ^enableThenGoto(serve)  
u1:"{I want} strawberry {ice cream} {please}"  
sure! $ice=strawberry ^enableThenGoto(serve)  
u1:[["I don't want {any} ice cream" "No {thank you}"]]  
okay $ice=None ^enableThenGoto(name)  
  
proposal: %serve Here comes your ice cream!  
u1:[["thank you" "thanks"]]  
You are welcome! $serve=finished ^enableThenGoto(name)  
  
proposal: %name What is your name?  
u1:{My name is} _*  
Nice to meet you $1 !
```

Interaktion: Chat

Komplexere Syntax (verschachtelte Antworten)?

- ...

Priority and scope: how does it work?

User rules are active as long as the Chat action is running. The **order** in which rules are written has no incidence on matching, and any of them can be matched at any moment.

This is not the case for **subrules**: they are subordinated to a user rule (or another subrule), and are only matchable once their parent rule has been matched.

For example, here is the correct way to ask for confirmation:

```
u:(I'm bored)
Shall we do some tongue twisters then?
u1:(yes)
Great!
u1:(no)
No problem!
```

Interaktion: Chat

Komplexere Syntax

Zum Nachlesen, oder falls ihr mehr Möglichkeiten haben wollt:

[https://
developer.softbankrobotics.com/pepper-qisdk/
lessons/discovering-
qichat-sbr-language-
creating-chatbots/
creating-topic-qichatbot-
and](https://developer.softbankrobotics.com/pepper-qisdk/lessons/discovering-qichat-sbr-language-creating-chatbots/creating-topic-qichatbot-and)

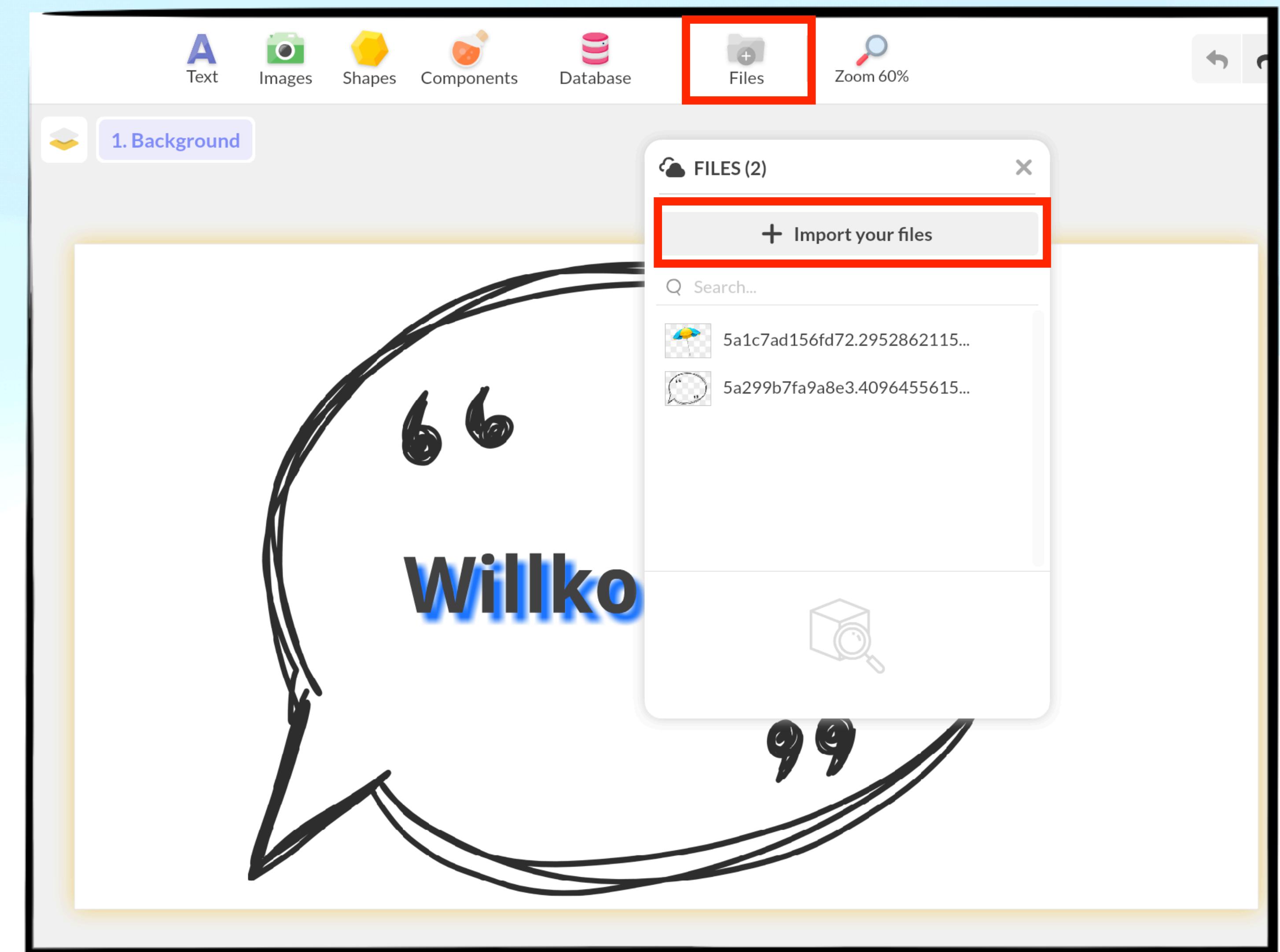
Detailed skills

- Building a Chat and QiChatbot
- QiChat syntax:
 - List of synonyms (or): []
 - Optional words: {}
 - Concepts: ~
 - Lexicon
 - Input storing: _ and \$1
 - Wildcard: *
 - Random function: ^rand
 - First function: ^first

Interaktion: Chat-Datei hochladen

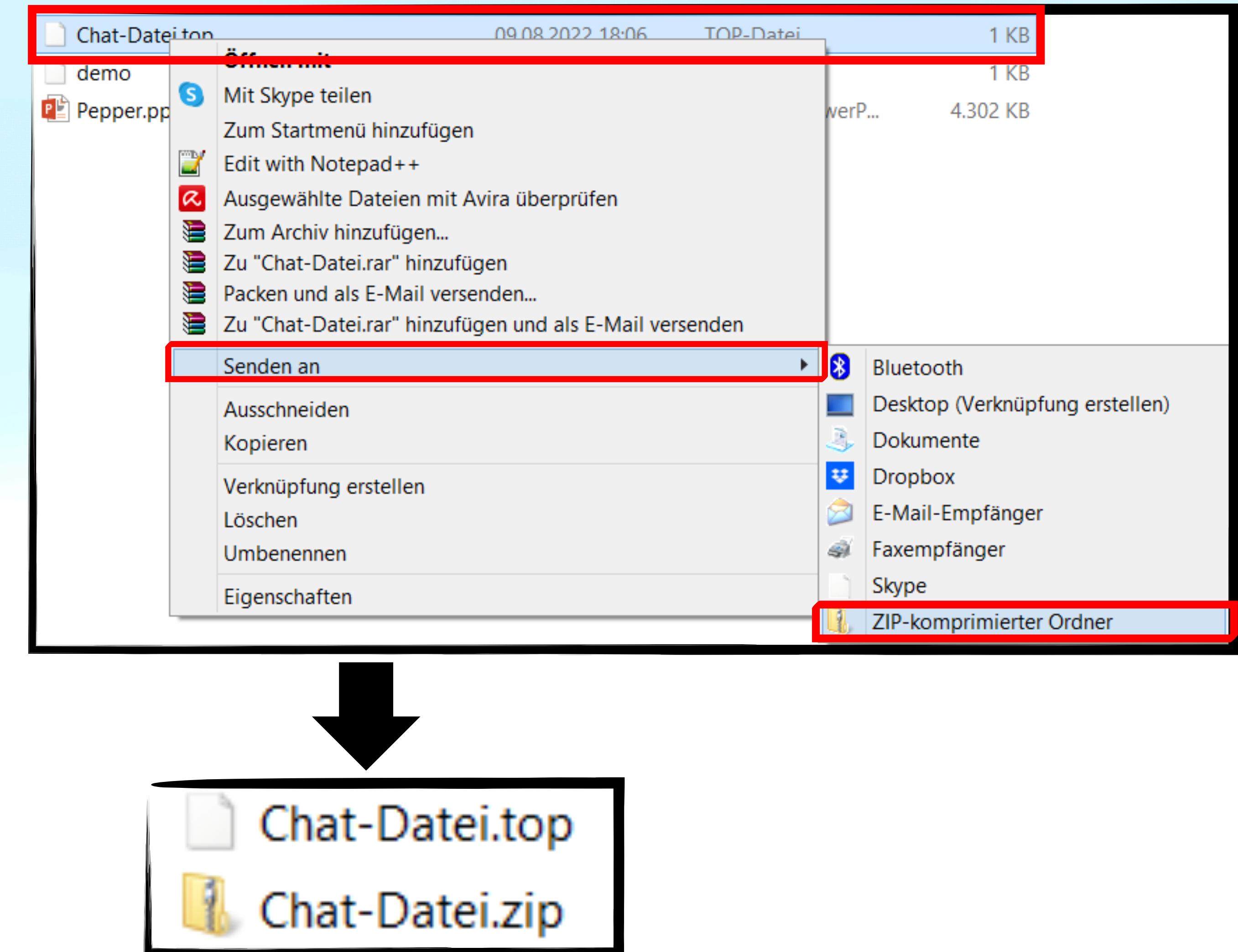
Füge deine Chat-Datei
über “Files” ein

Die Datei muss
einzelN in einem **ZIP-**
Ordner hochgeladen
werden!



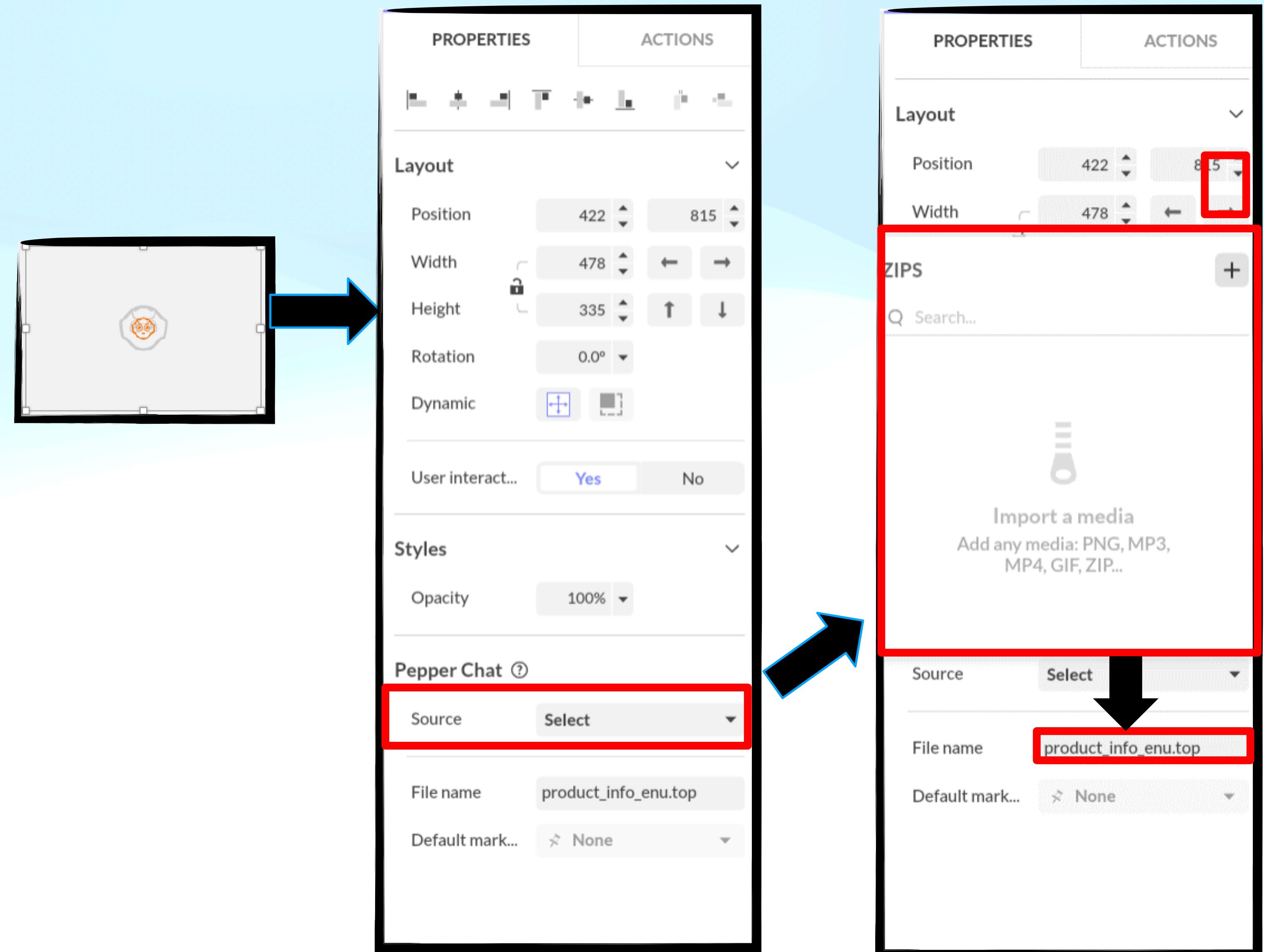
Erstellen des zip-Ordners

- Rechtsklick auf die Chat-Datei
- Senden an → zip-komprimierter Ordner
- Jetzt gibt es einen zip-Ordner mit dem gleichen Namen wie die Datei



Datei verknüpfen mit Pepper Talk

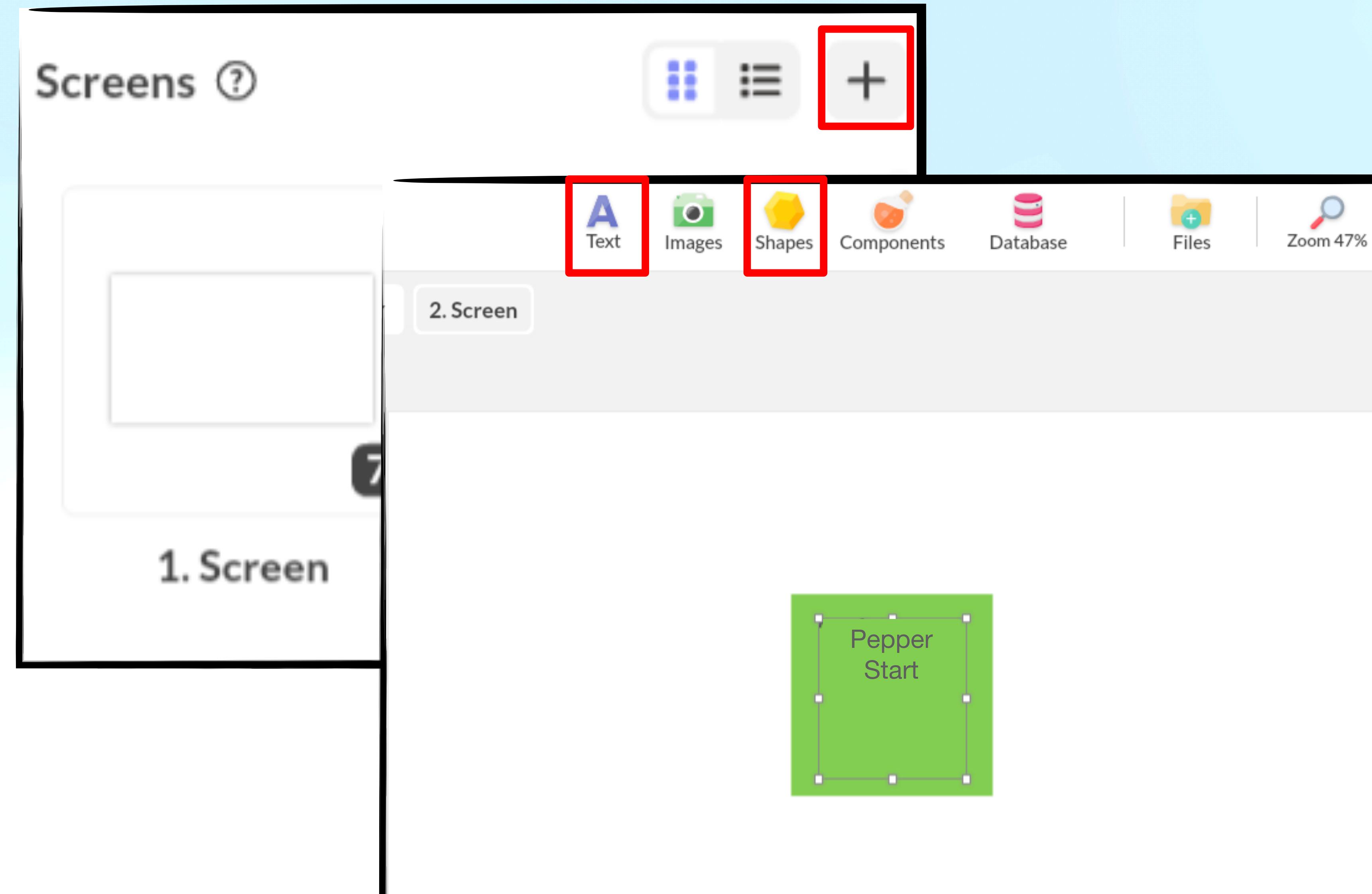
- Im Hintergrund Pepper Talk anklicken
- Bei „Properties“ > Source den vorher hochgeladenen zip-Ordner auswählen („+“ anklicken)
- Bei „File name“ muss der Name von der Datei, die in dem Ordner liegt, stehen



Interaktion: Chat zum Laufen bringen (1)

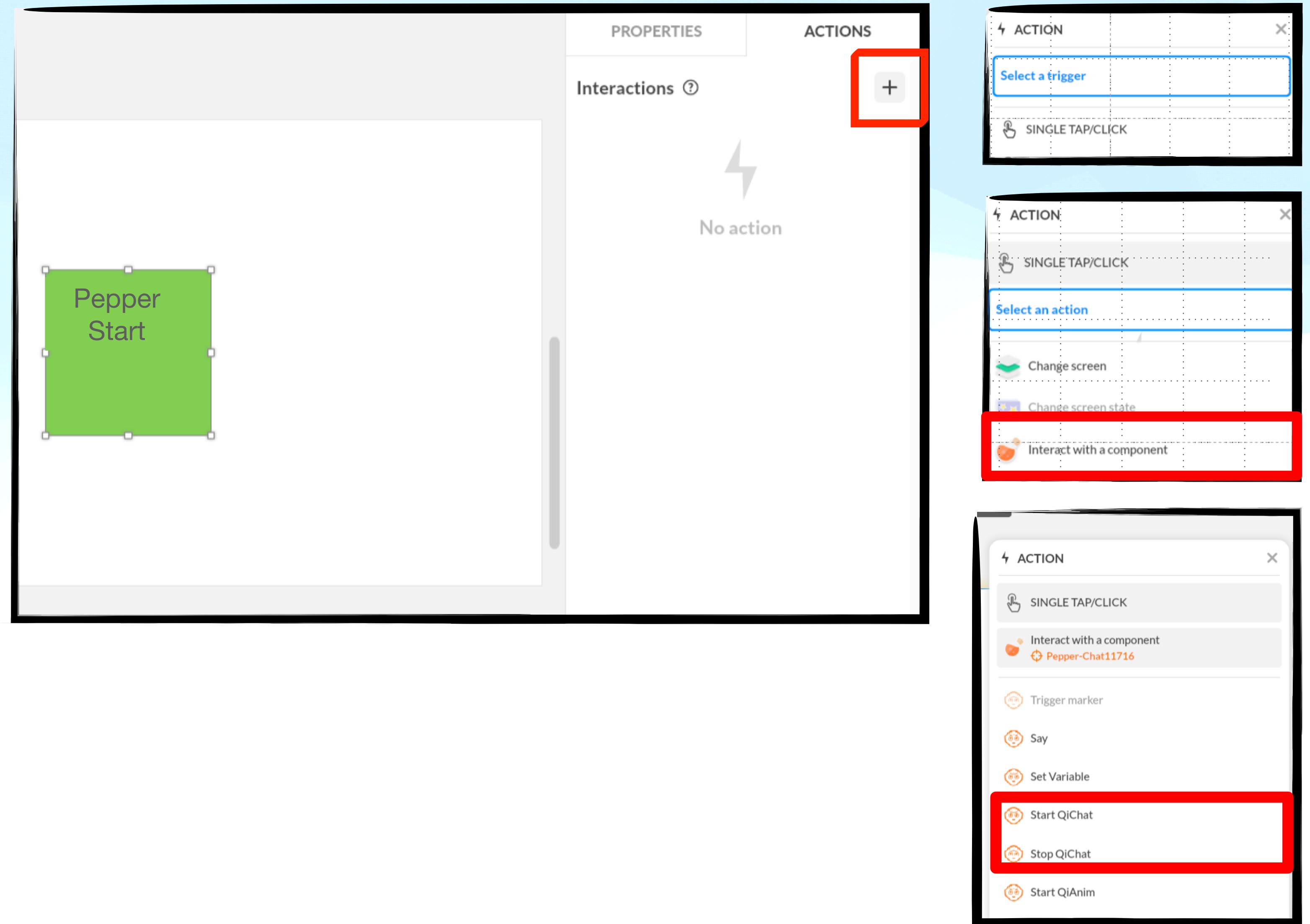
Erstelle einen Screen-State mit einem Foreground mit zwei Knöpfen:

- Chat-Start
- Chat-End



Interaktion: Chat zum Laufen bringen (2)

- Füge die Aktion Chat-Start zum Start-Knopf hinzu
- Und die Aktion Chat-End zum End-Knopf



Interaktion: Chat

State wechseln

<https://www.youtube.com/watch?v=7vTBvtkwLal&t=1185s>
19:30

Aufgabe: GUI

<Thema> (am Besten gleiches Thema wie in der Demo)

- 1 x Background
- 1 x Screen
- 2 x States je ein Foreground
 - + 1 x Übergang
- 1 x Pepper reden lassen
- 1 x Animation

Aufgabe: Chat

<Thema> (am Besten gleiches Thema wie in der Demo)

- 1 x Begrüßung
- 1 x Pepper Fragt, du antwortest
- 1 x Zufallsantwort
- 1 x Information speichern
- 2 x Alternativen
 - > 1 x State-Übergang

Werdet kreativ

Und bastelt weiter an eure App!