Prototyping Workflow Automation with Snap*!*

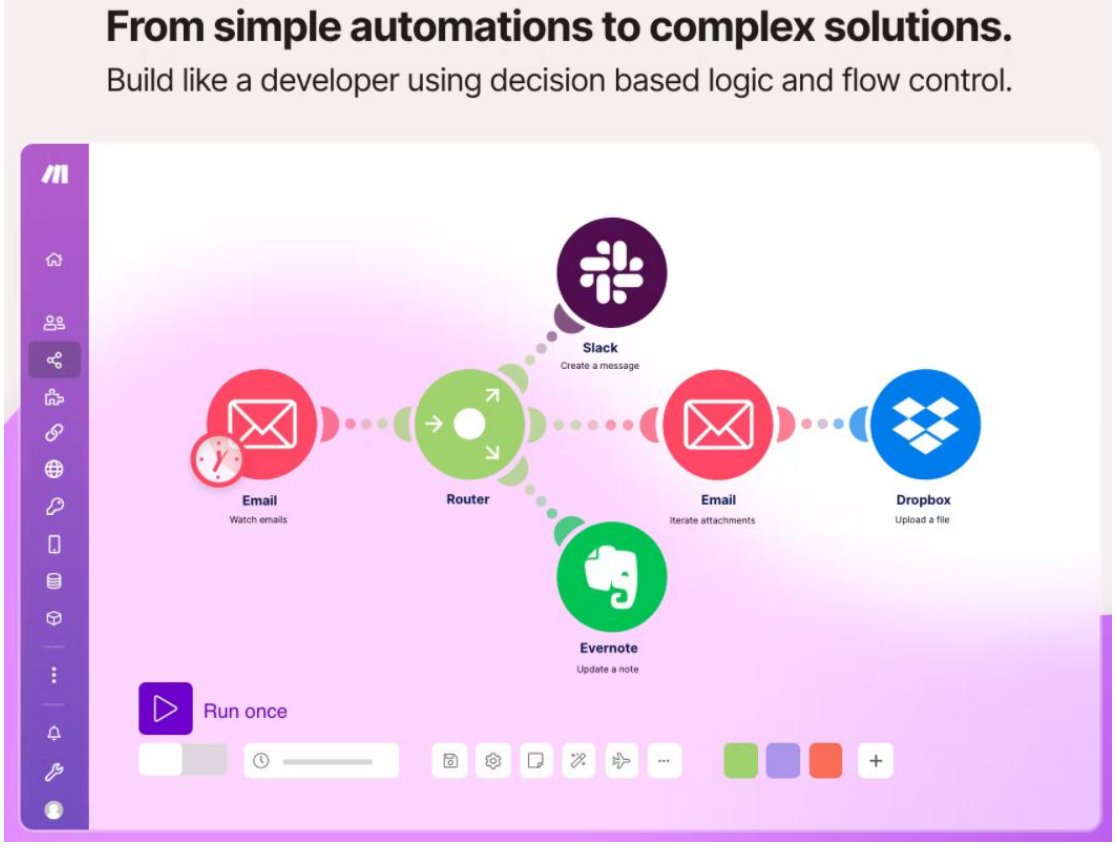
Xavier Pi

Snap*!*Con 2025

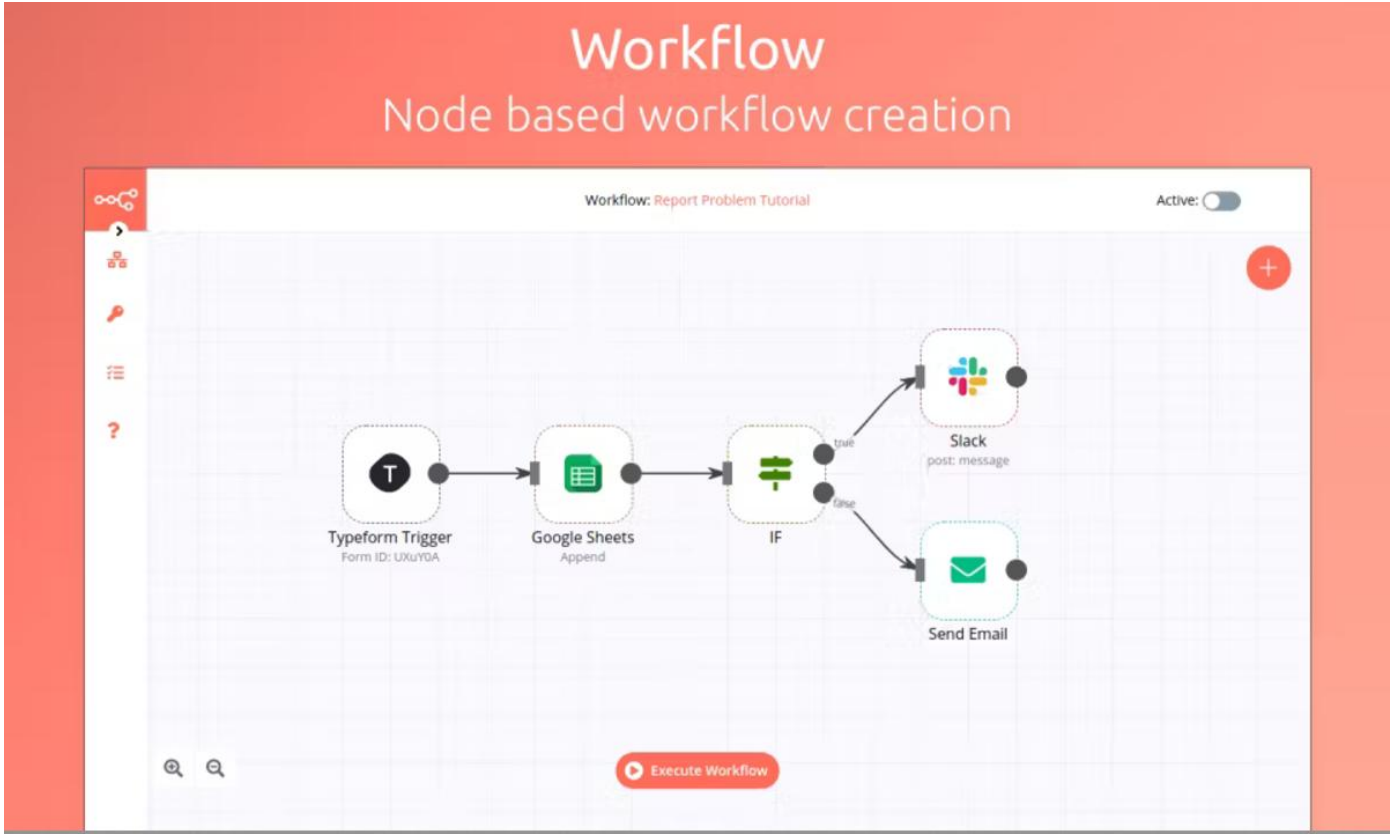
Popular Workflows Automation

- Popular workflow automation tools, such as Make, n8n, Zapier, Microsoft Power Automate, or SAP Build Process Automation, propose the use of "low-code" graphical languages to minimize the barrier to entry.
- Technically, they are based on flowcharts assisted with tools that facilitate their use.

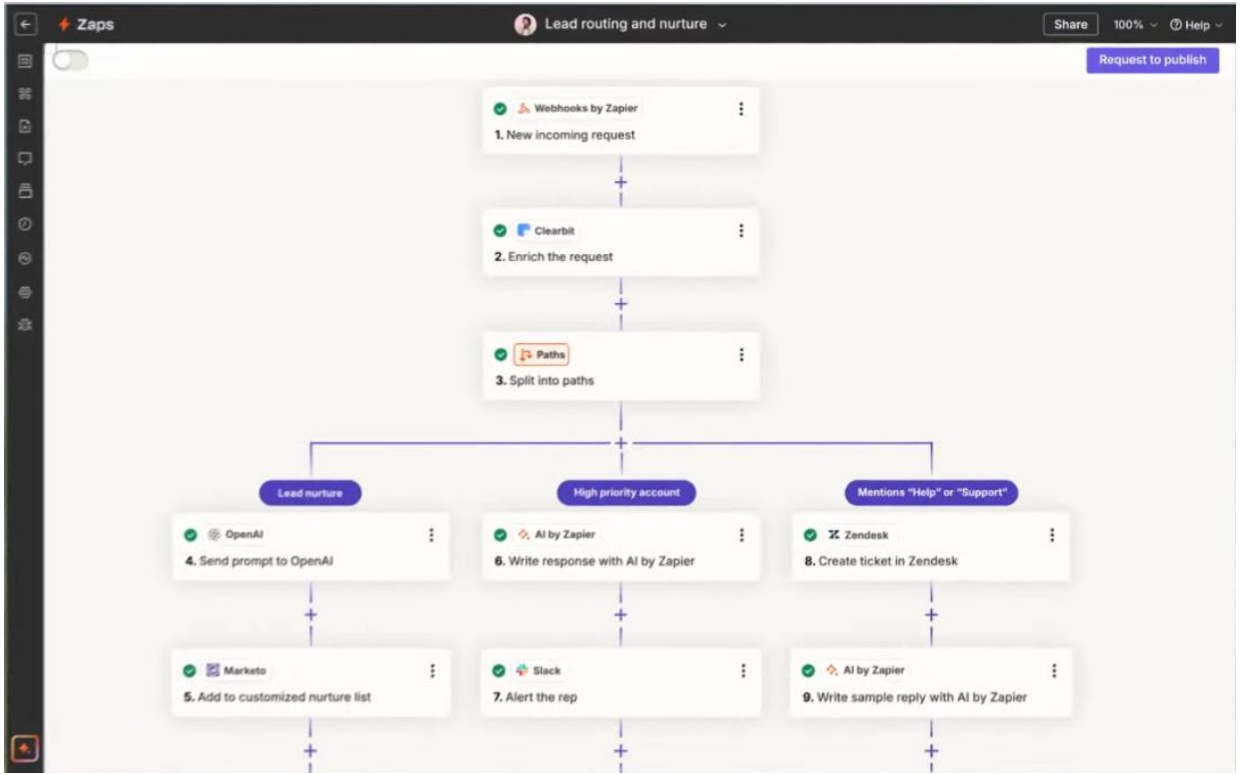
Make


From simple automations to complex solutions.
Build like a developer using decision based logic and flow control.
Slack
Create a message
Email
Watch emails
Router
Email
Iterate attachments
Dropbox
Upload a file
Evernote
Update a note
Run once

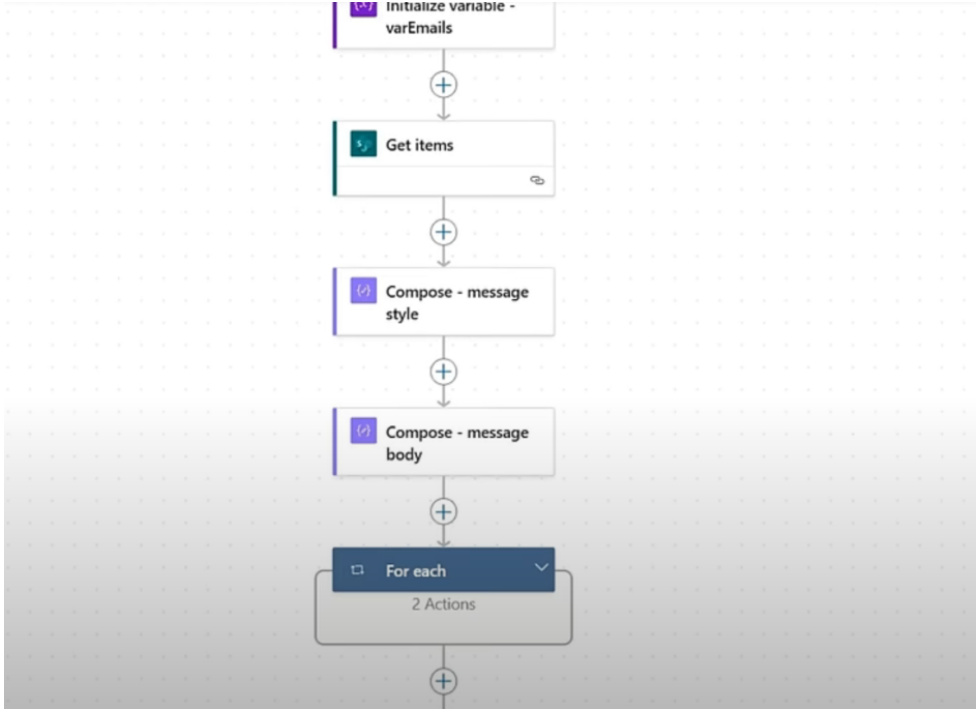
n8n


Workflow
Node based workflow creation
Workflow: Report Problem Tutorial
Active:
Typeform Trigger
Form ID: UXuY0A
Google Sheets
Append
IF
true
false
Slack
post: message
Send Email
Execute Workflow

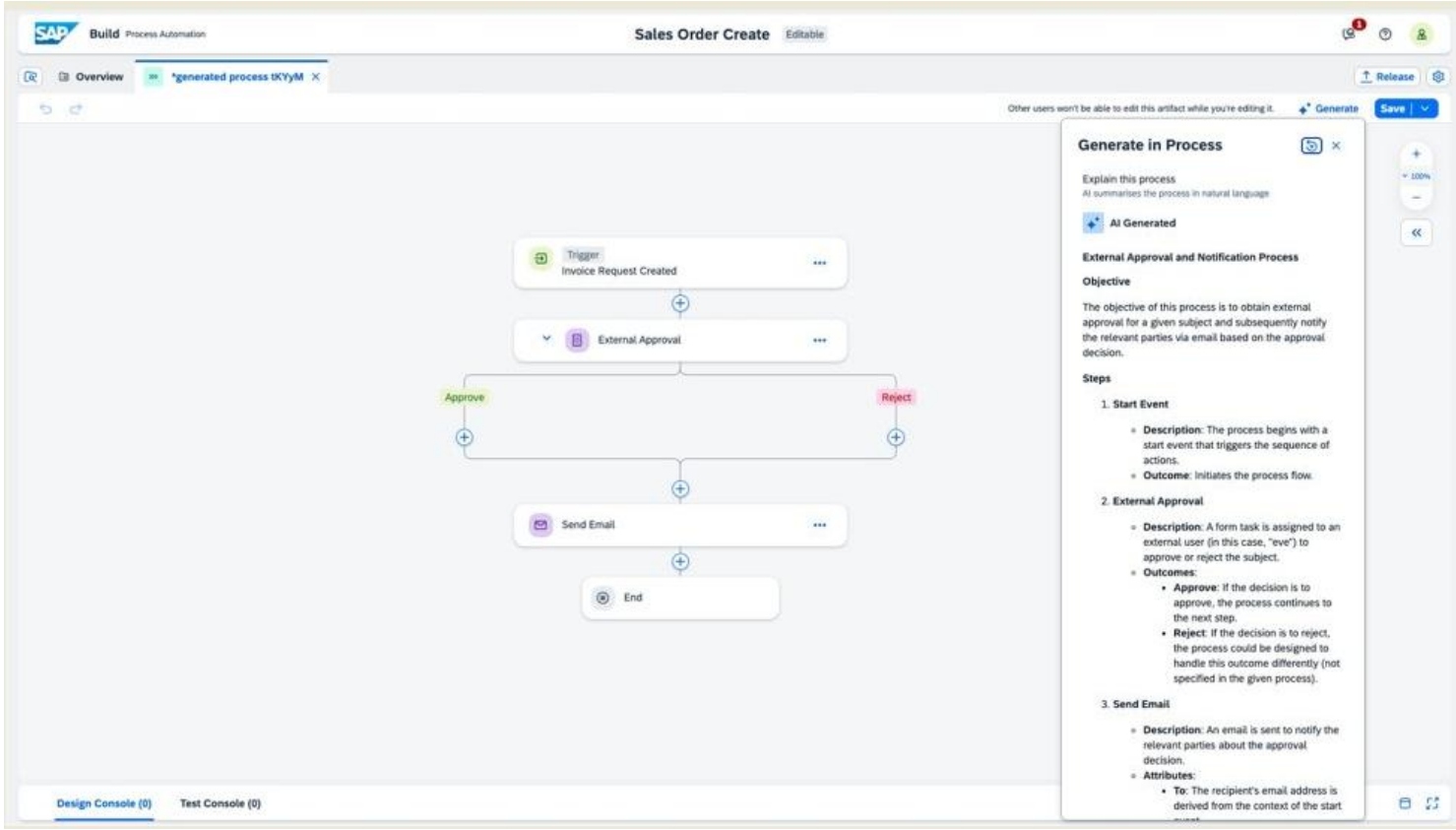
Zapier


Zaps
Lead routing and nurture
Share
100%
Help
Request to publish
Webhooks by Zapier
1. New incoming request
Clearbit
2. Enrich the request
Paths
3. Split into paths
Lead nurture
High priority account
Mentions "Help" or "Support"
OpenAI
4. Send prompt to OpenAI
AI by Zapier
6. Write response with AI by Zapier
Zendesk
8. Create ticket in Zendesk
Marketo
5. Add to customized nurture list
Slack
7. Alert the rep
AI by Zapier
9. Write sample reply with AI by Zapier

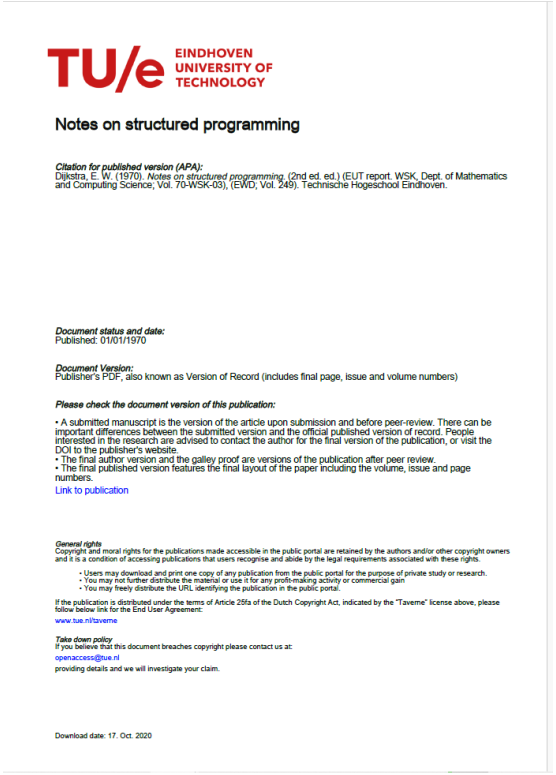
Microsoft Power Automate



SAP Build Process Automation



Algorithmic Foundations



TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Notes on structured programming

TECHNISCHE HOGESCHOOL EINDHOVEN
NEDERLAND
ONDERAFDELING DER WISKUNDE

TECHNOLOGICAL UNIVERSITY EINDHOVEN
THE NETHERLANDS
DEPARTMENT OF MATHEMATICS

NOTES ON STRUCTURED PROGRAMMING

by

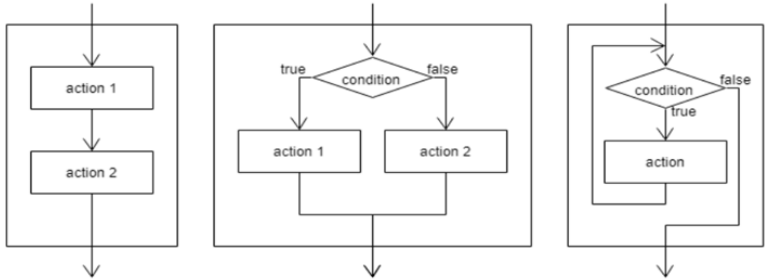
Prof.dr. Edsger W. Dijkstra

T.H.-Report 70-WSK-03

Second edition April 1970

Algorithmic Structures

Flowchart



Pseudocode

```
...
action 1
action 2
...
```

```
...
if condition then
   action 1
else
   action 2
endif
...
```

```
...
while condition do
   action
endwhile
...
```

Snap*!*

(from Iteration, composition library)

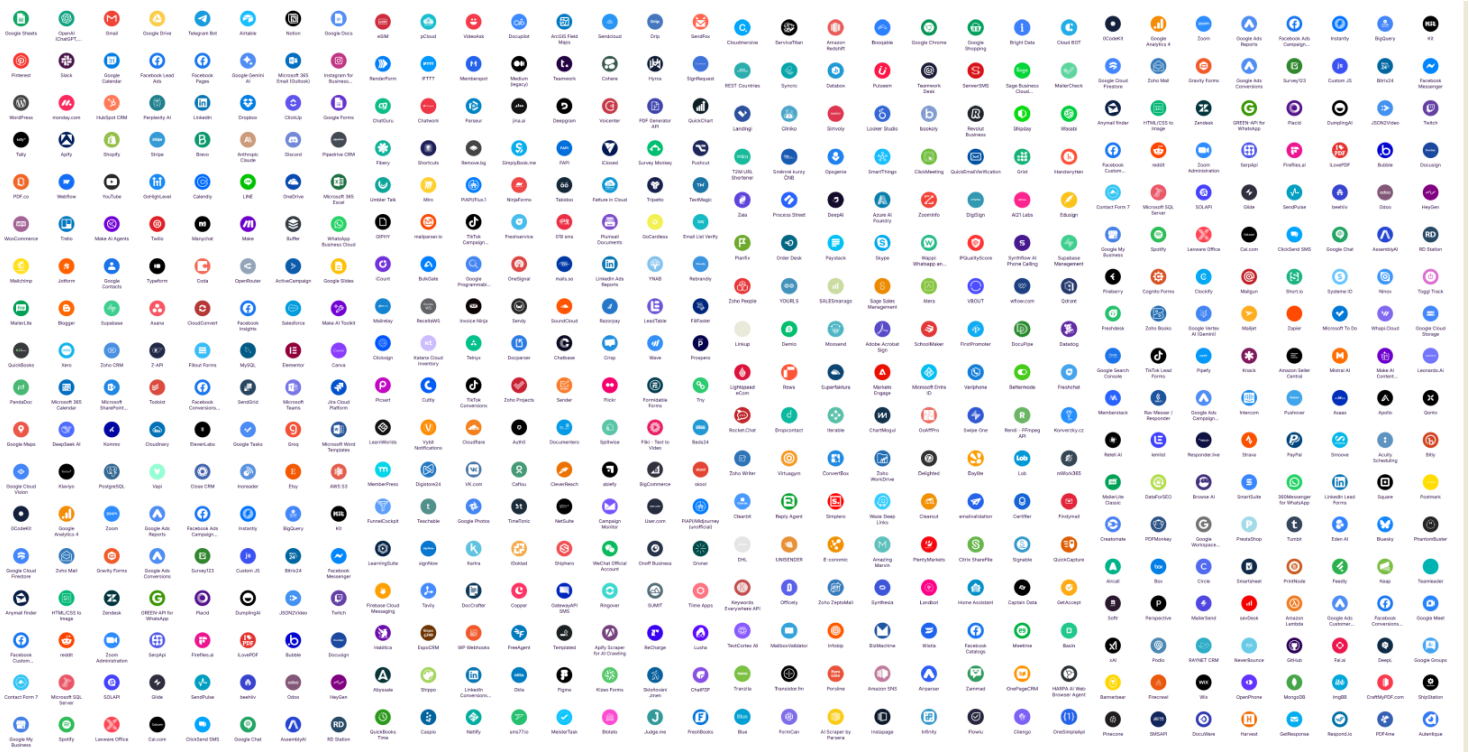
Basic Common Elements

- Actions or tasks
- Events or triggers
- Decisions and loops
- Connectors

Connectors

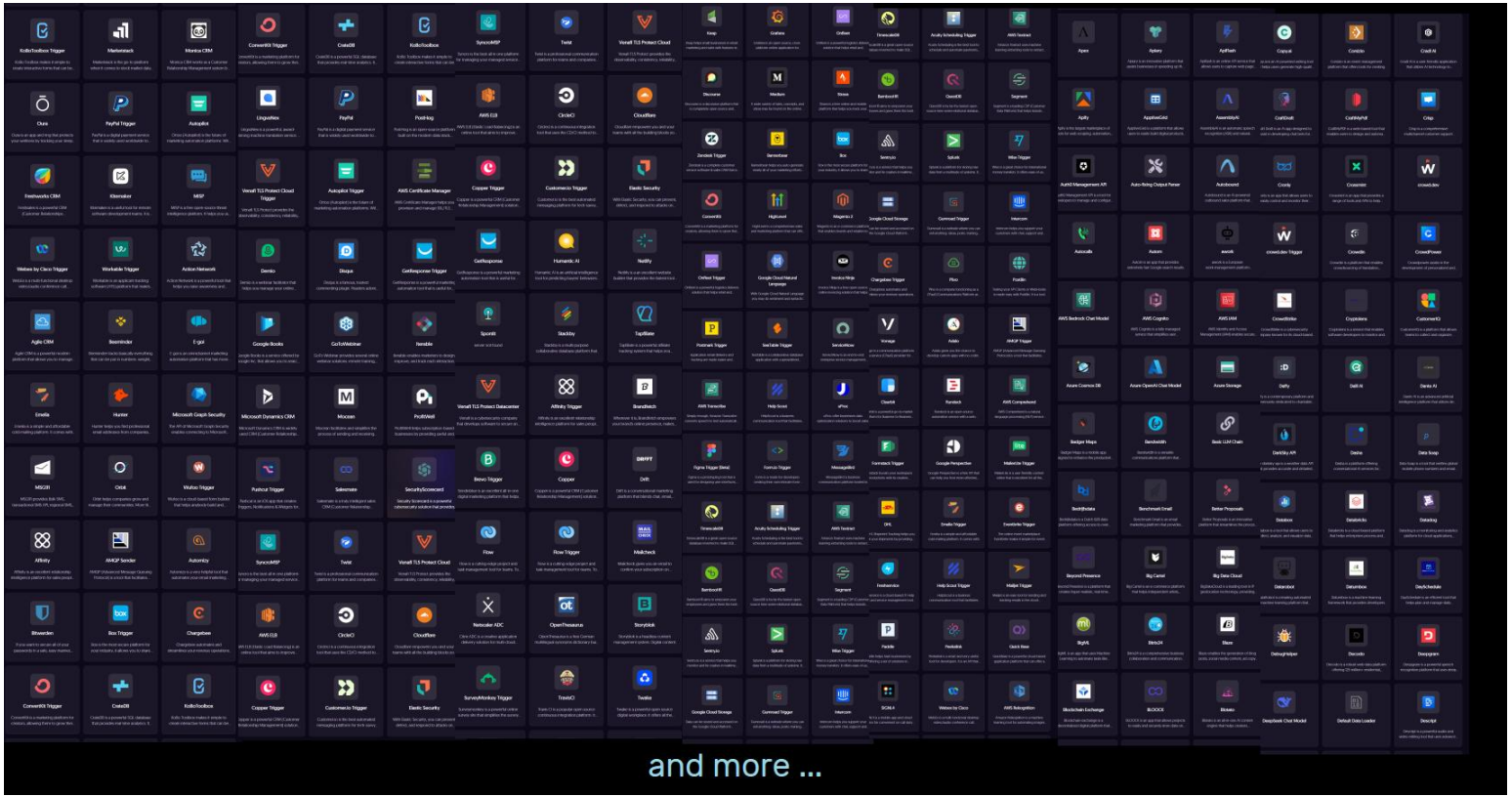

Gmail
Telegram Bot
Google Sheets
Google Drive
Google Calendar
SAP ECC Agent
OpenAI (ChatGPT, Whisper, DALL-E)
WhatsApp Business Cloud
WooCommerce
Microsoft 365 Calendar
Salesforce
Google Gemini AI
WordPress
Microsoft Dynamics 365 - CRM
PostgreSQL
and more than 2800 ...

Make Connectors

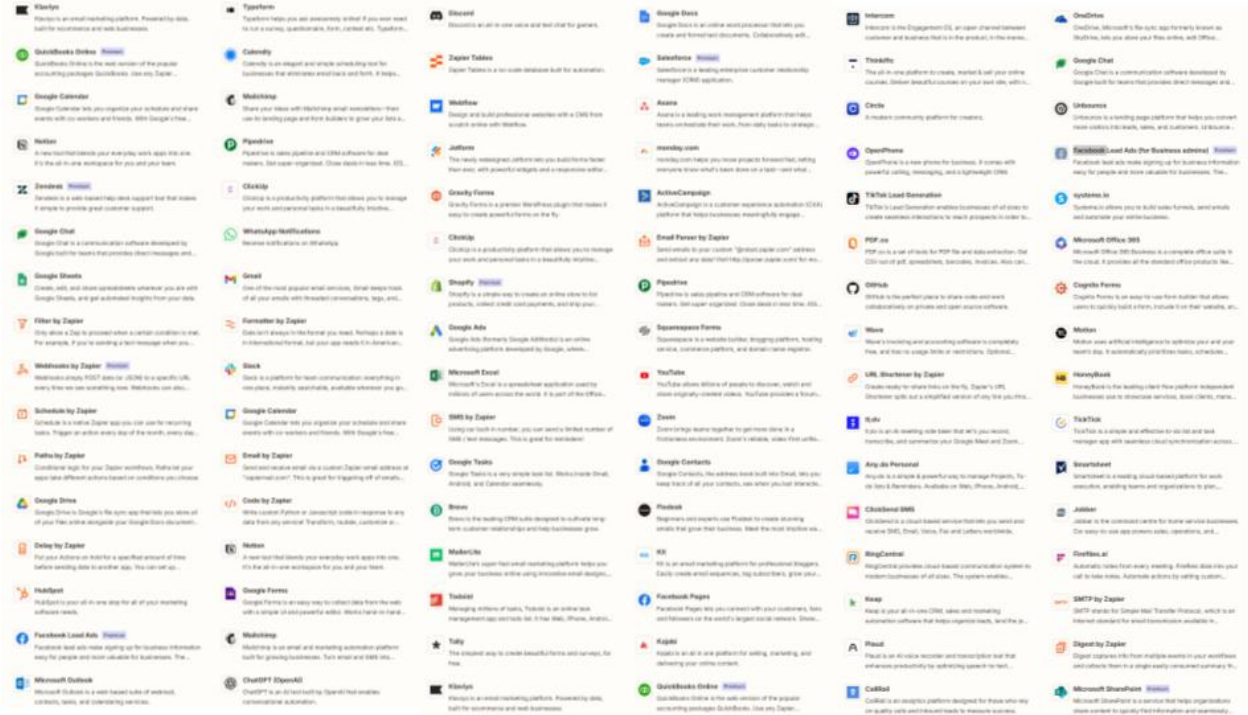


and more and growing ...

n8n Connectors

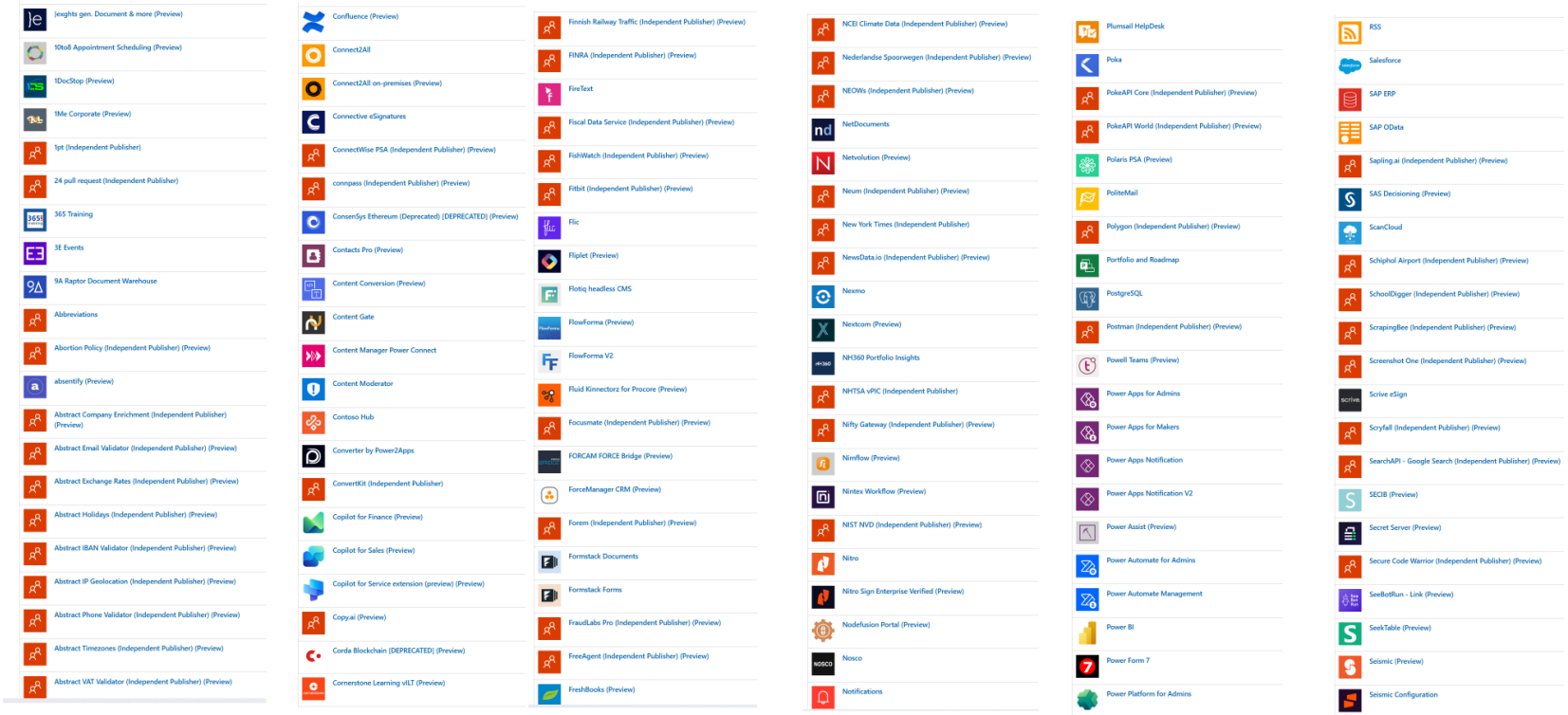


Zapier Connectors



and more ...

Microsoft Power Automate Connectors



and more and growing ...

A Basic Connector Example

Two Approaches

- Pure Snap*!* Implementation
  - No dependencies
  - Architecturally and technically limited
- Snap*!* + External bridge agent support
  - Depends on an external bridge agents that call existing APIs (there are thousands of them).
  - Architecturally unlimited and technically almost unlimited

Pure Snap*!* Implementation

- Using Snap*!* “Web services access (https)” only allows direct calls to some APIs, such as API-KEY-based ones.
- Security issue: API key is visible in the code.
- Many APIs cannot be called directly from browsers for security reasons.

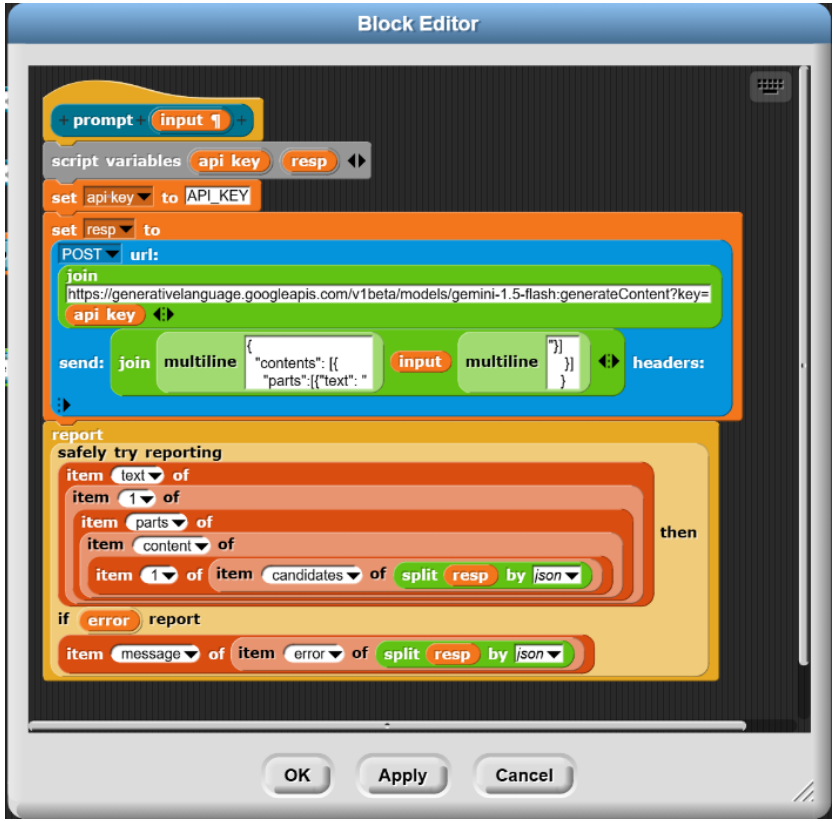
External Bridge Agents

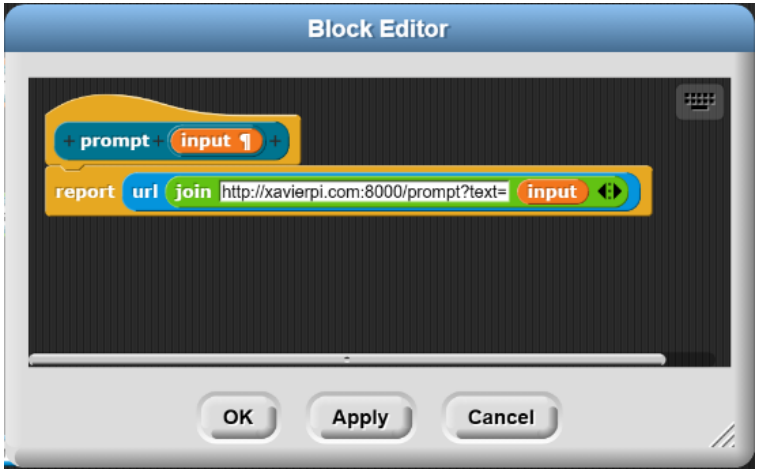
- Based on independent agent which uses existent implemented APIs.
- Example:
  - Flask Python process
  - Can access to any API (Key based, token based, Oauth 2, etc)
  - Accessed from Snap*!* via HTTP or MQTT
  - Access to Snap*!* via MQTT

External Bridge Agent Example

```python
import google.generativeai as genai
import os
from flask import Flask, request
from flask_cors import CORS
import urllib.parse

app = Flask(__name__)
CORS(app)

API_KEY = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
genai.configure(api_key=API_KEY)

def call_gemini_api(prompt):
    model = genai.GenerativeModel('gemini-1.5-flash')
    response = model.generate_content(prompt)
    return response.text


@app.route('/prompt', methods=['GET'])
def prompt():
    text_prompt = urllib.parse.unquote_plus(request.args.get('text', ''))
    return call_gemini_api(text_prompt)


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True, port=8000)
```

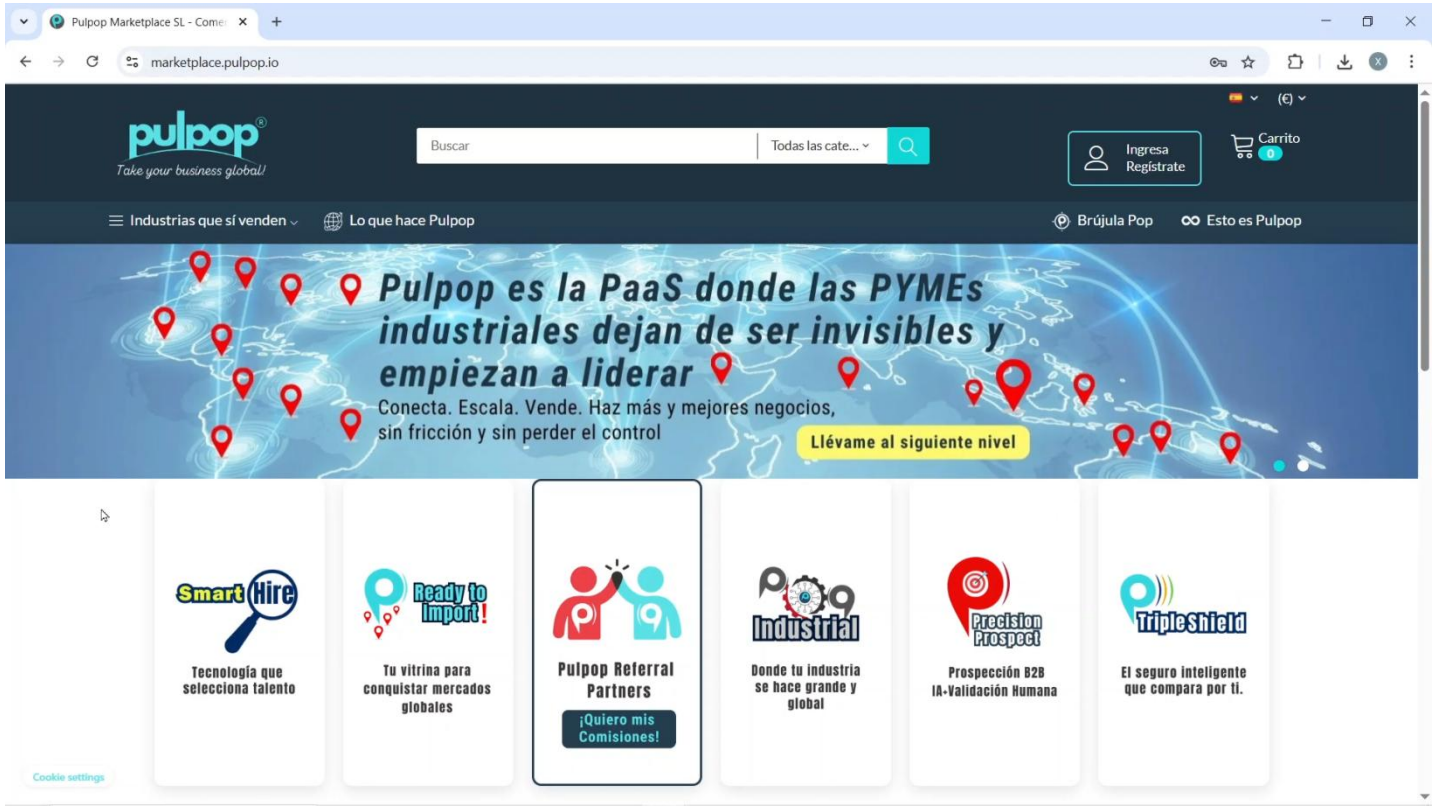
Pure Snap*!* vs External Bridge Agent



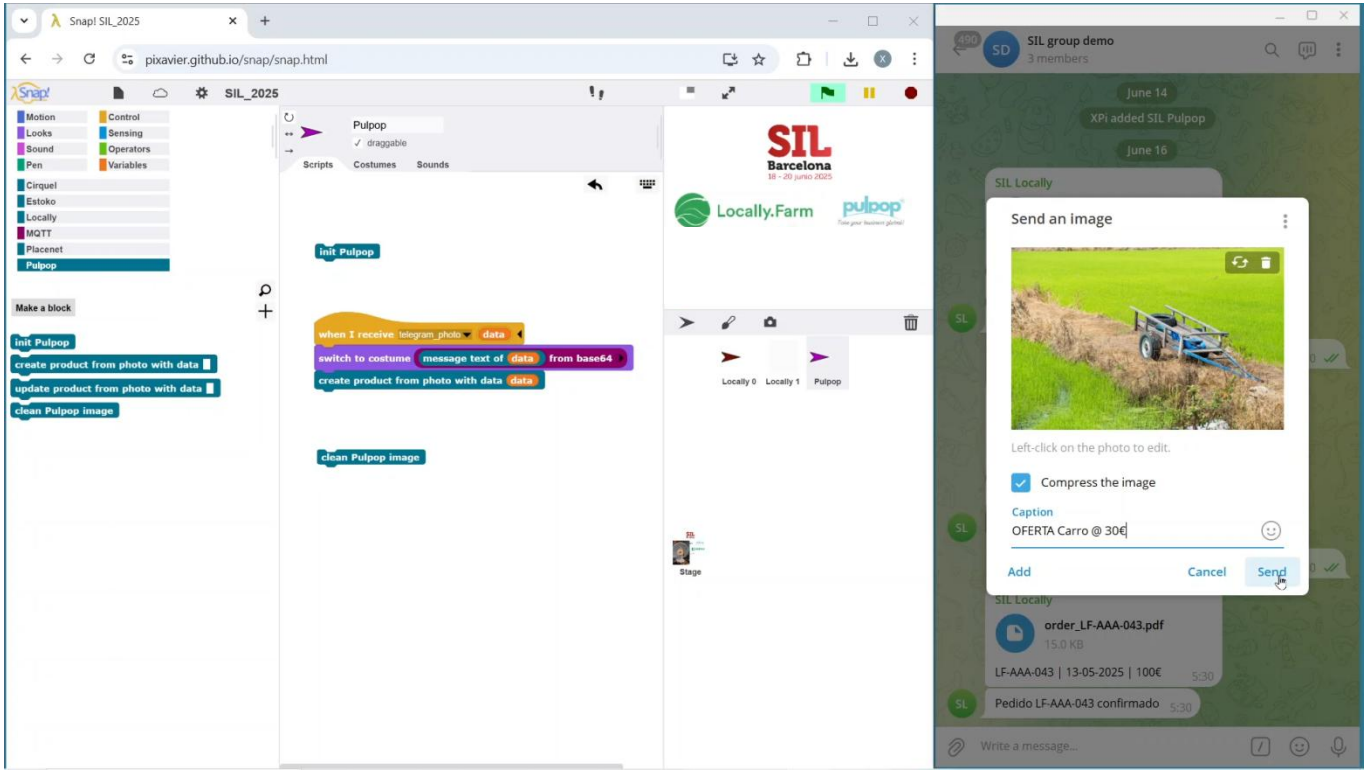


← https://xavierpi.com/ia

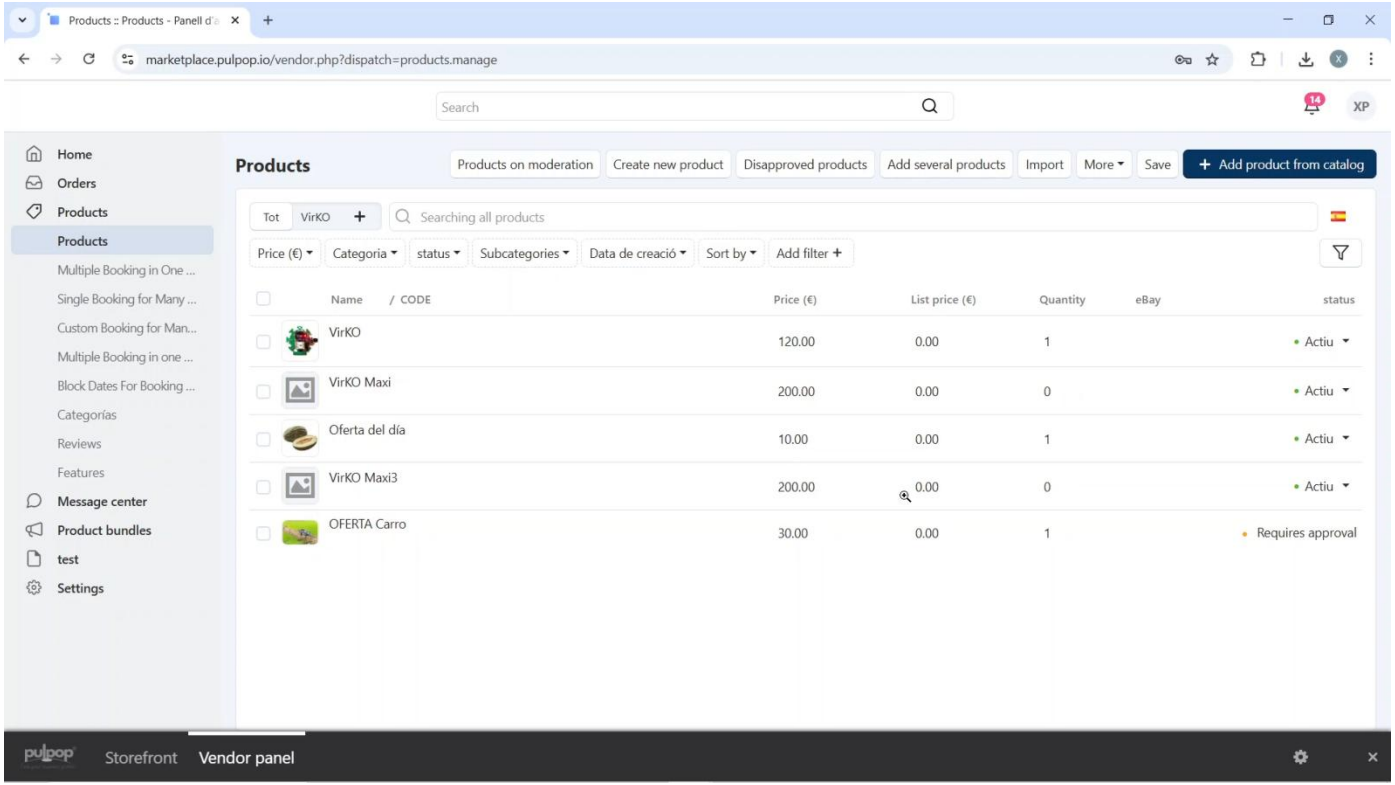
Experiences at Barcelona SIL 2025



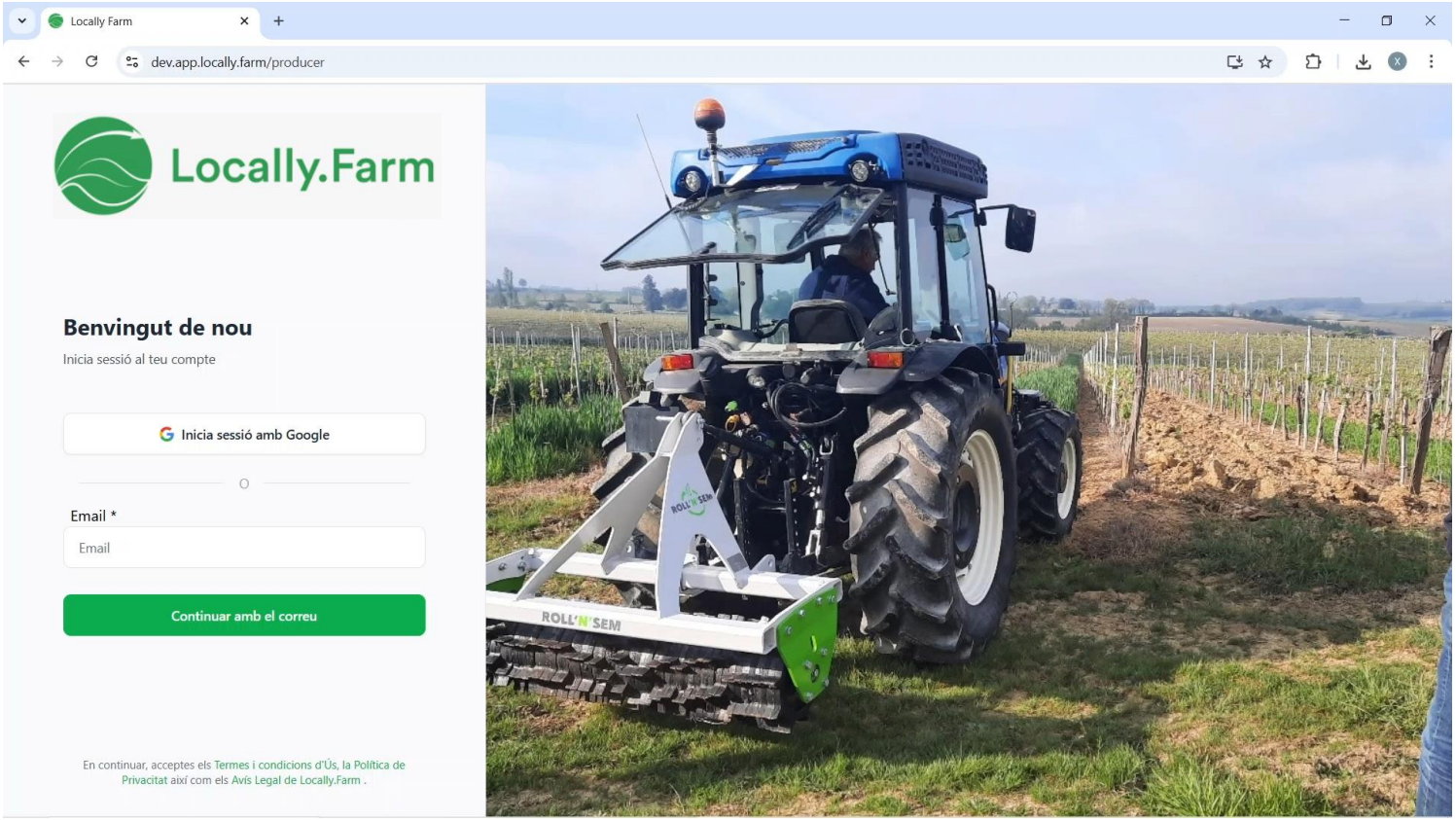
Experiences at SIL 2025



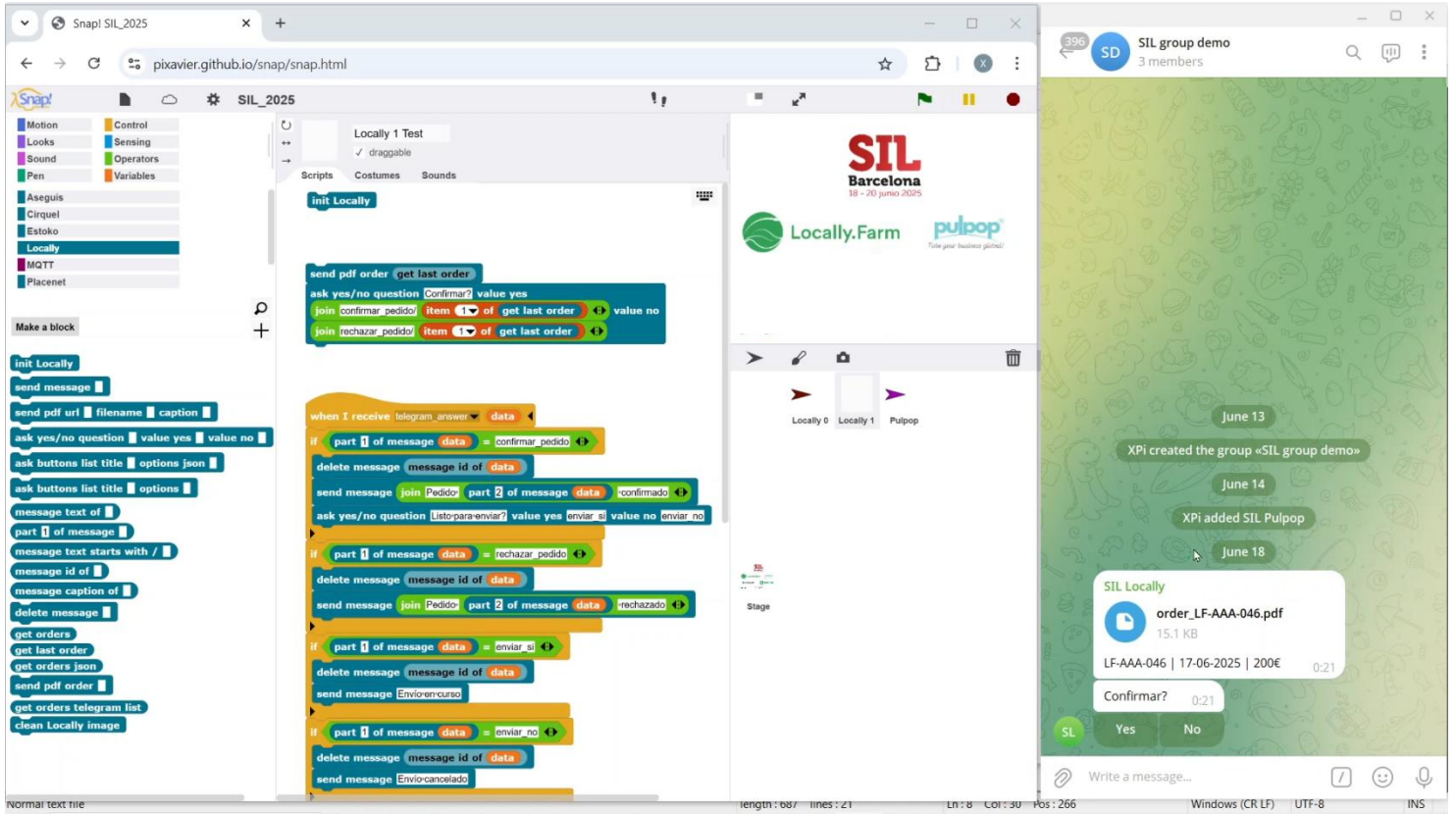
Experiences at SIL 2025



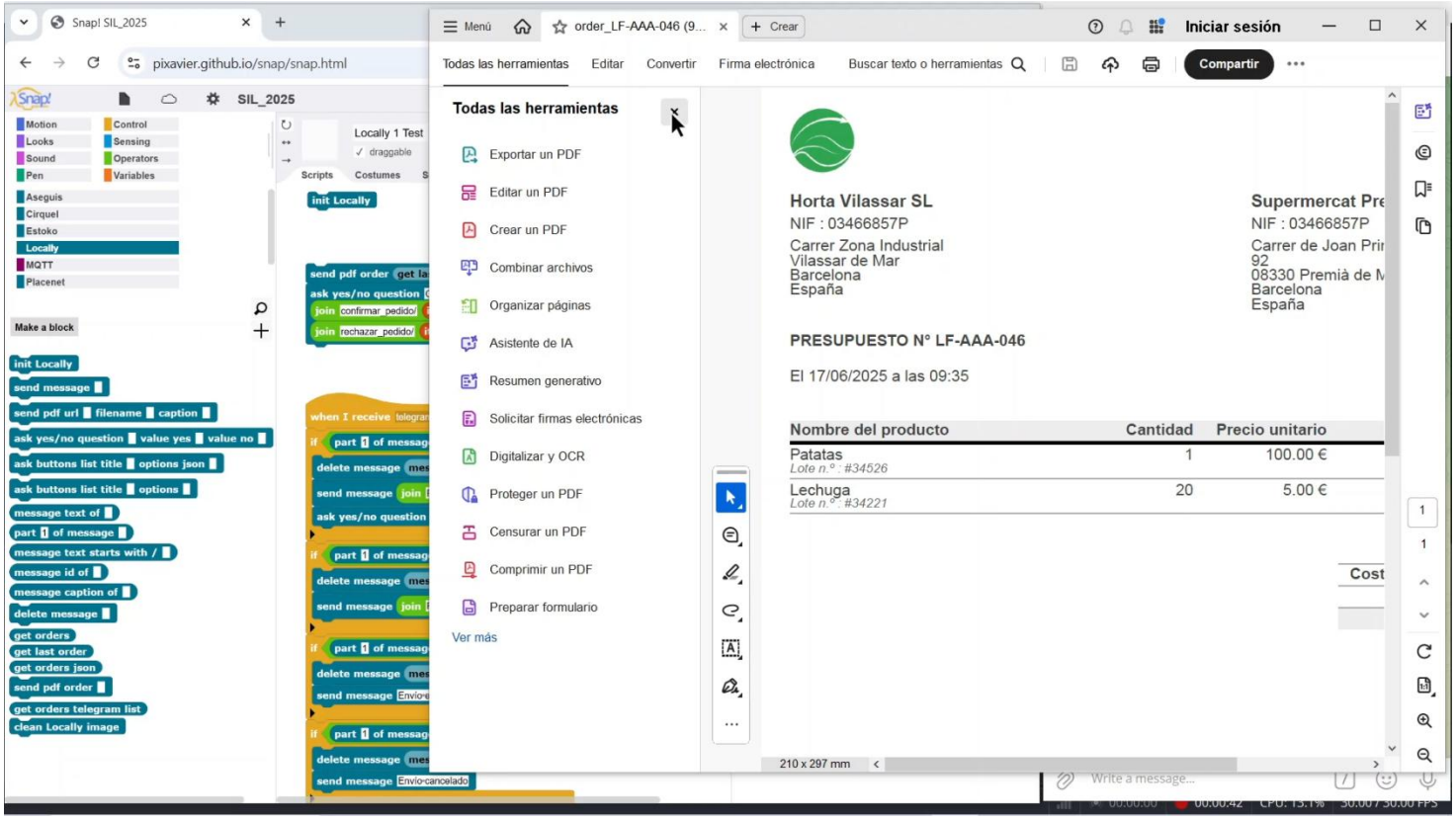
Experiences at SIL 2025



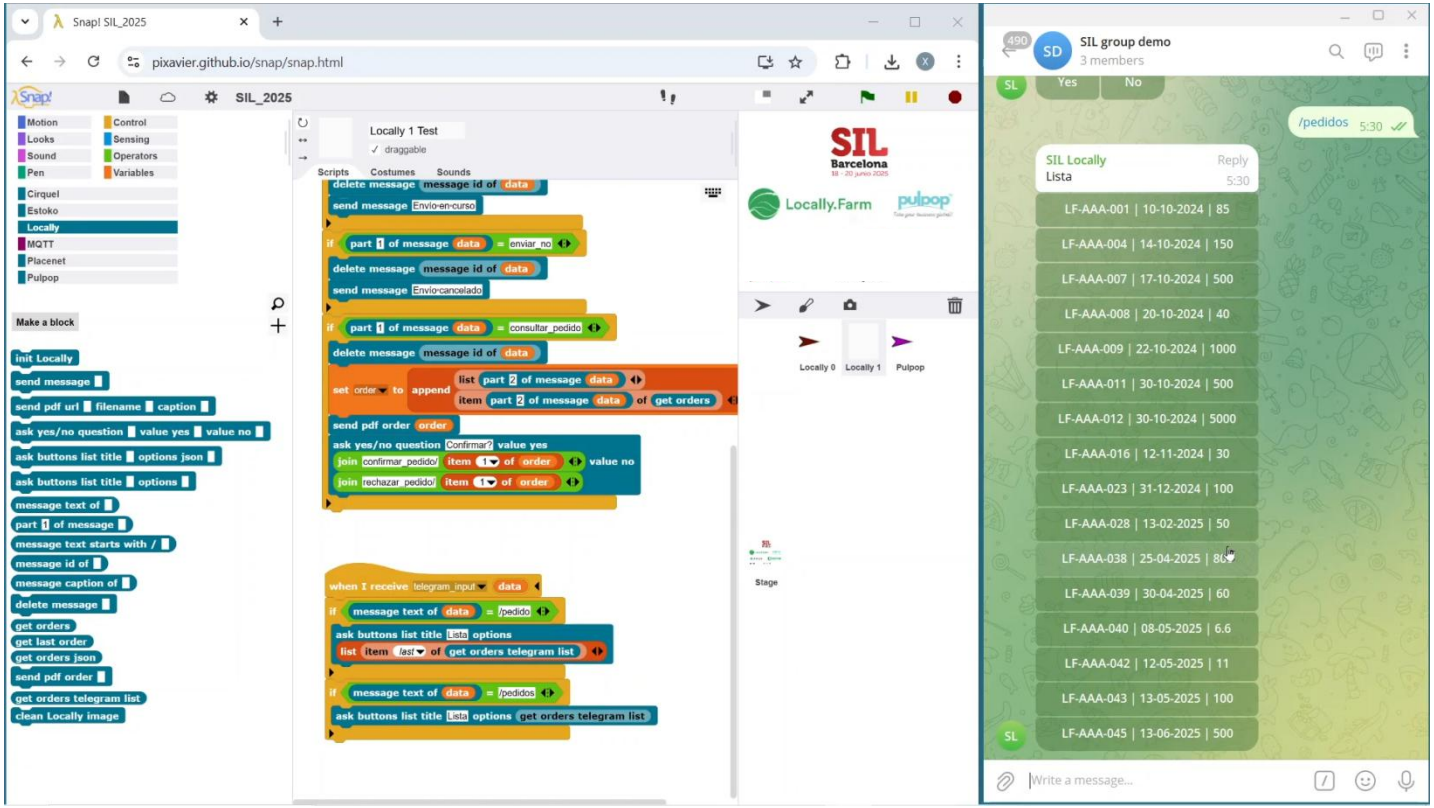
Experiences at SIL 2025



Experiences at SIL 2025



Experiences at SIL 2025



Recap

- Block-based languages like Snap*!* are lingua franca for almost the whole new generation; there is no need to create parallel discourses to explain computational thinking (algorithmic thinking, decomposition, abstraction, and pattern recognition).

- Undergraduate programmers can develop connectors based on external bridge agents, easily using AI in most cases, on a cheap VPS or cloud service.

- Snap*!* is architecturally complete (bidirectional sync and async agents interactions with client-server and PubSub architectures) thanks to “Web services access (https)” and “MQTT” libraries.

Thank you !

Xavier Pi
xpi@enginyers.net