Building applications with spreadsheets

Jesús Sánchez Cuadrado, Gonzalo Manzanares

Universidad de Murcia

LangDev CON 2024

Seville 17-19 October, 2024

https://langdevcon.org

7IOH

ProxyHands

jesusc@um.es

@sanchezcuadrado

http://github.com/jesusc

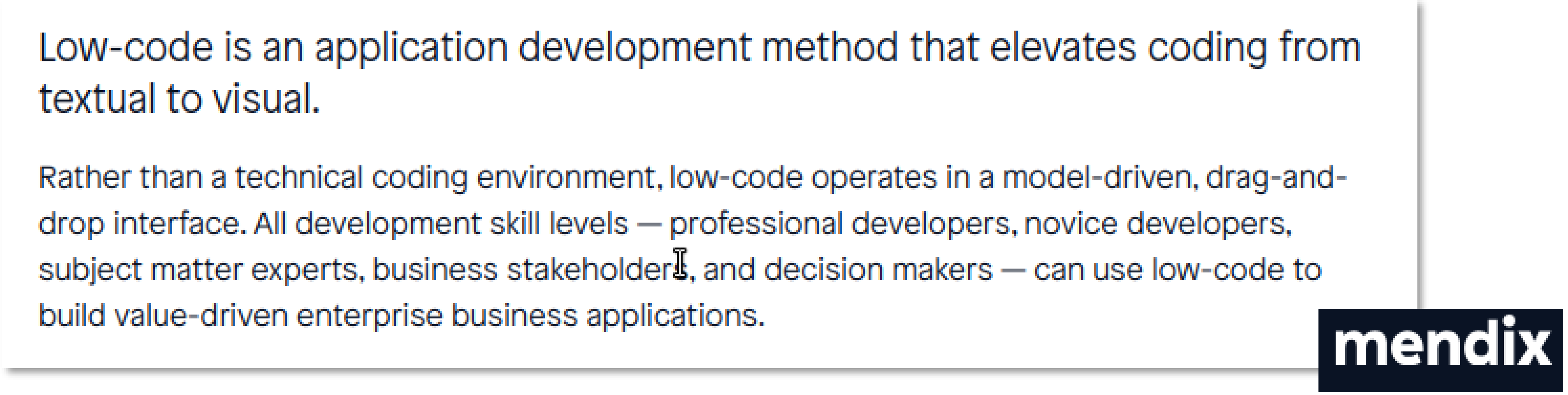
http://sanchezcuadrado.es

Motivation – **Spreadsheets are applications**

- This work originates from the following observations:
  - Excel is the most widely used programming language
  - Why non-programmers are able to use Excel but they can’t program?
    - Because they manipulate values (e.g., numbers) directly
  - Spreadsheets work as “poor man’s” business applications
  - Limitations:
    - The data is coupled with the code
      - Reusing requires deleting the data
      - Error-prone
    - The spreadsheet can’t be scaled/shared
    - Adding a visual interface is not within a normal user’s reach
    - There may be performance issues
    - Spreadsheets can have errors too

Low-code platforms

- What is low-code?



Low-code is an application development method that elevates coding from textual to visual.

Rather than a technical coding environment, low-code operates in a model-driven, drag-and-drop interface. All development skill levels — professional developers, novice developers, subject matter experts, business stakeholders, and decision makers — can use low-code to build value-driven enterprise business applications.

  - Many times a rebranding of MDE
  - Many others a graphical DSL (+ tool) for a domain (e.g., business applications)
- The overarching goal of low-code is to let end-users build applications
  - Is this possible with current systems?

Low-code vs Spreadsheets

- Low-code
  - They assume that users have abstraction abilities
  - The data or domain model has to be designed up front
  - The concepts are unconnected to “normal people”

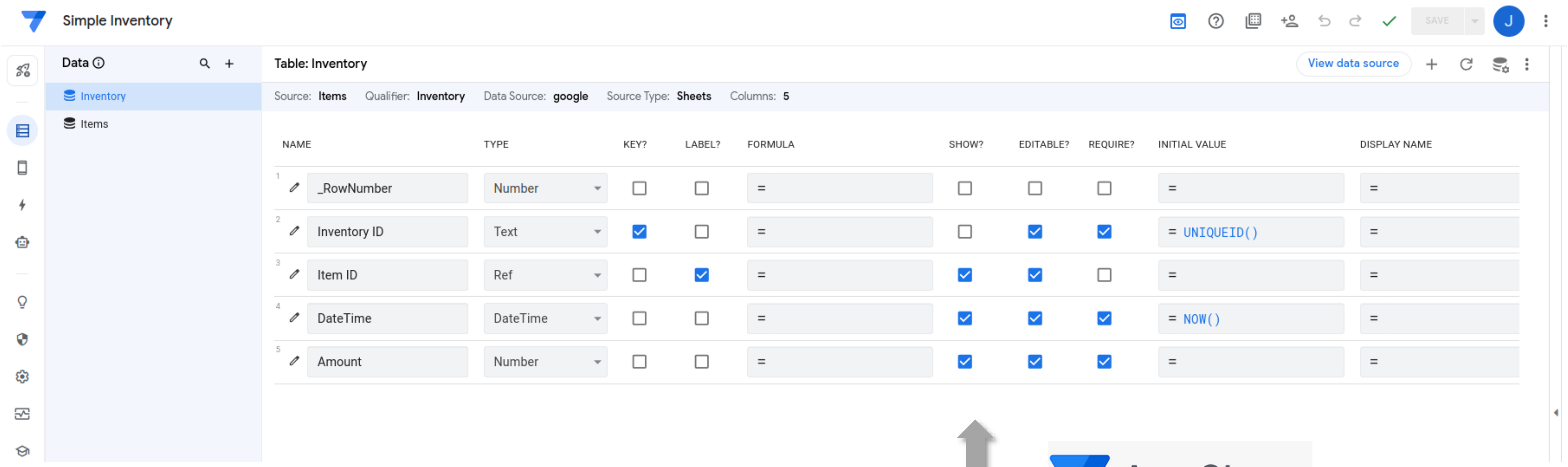


Table: Inventory

Source: **Items** Qualifier: **Inventory** Data Source: **google** Source Type: **Sheets** Columns: **5**

NAME	TYPE	KEY?	LABEL?	FORMULA	SHOW?	EDITABLE?	REQUIRE?	INITIAL VALUE	DISPLAY NAME
_RowNumber	Number	□	□	=	□	□	□	=	=
Inventory ID	Text	☑	□	=	□	☑	☑	= UNIQUEID()	=
Item ID	Ref	□	☑	=	☑	☑	□	=	=
DateTime	DateTime	□	□	=	☑	☑	☑	= NOW()	=
Amount	Number	□	□	=	☑	☑	☑	=	=

AppSheet

Low-code vs Spreadsheets

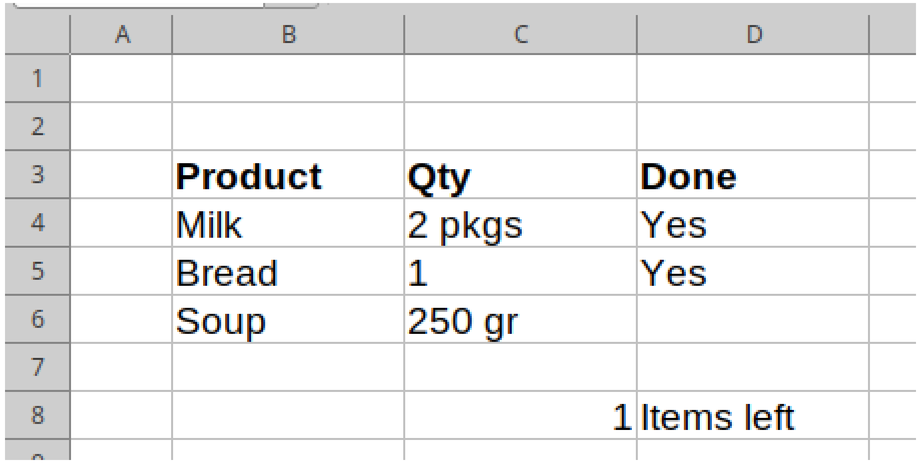
- Spreadsheets
  - Direct manipulation
  - Live programming
    - Results appears as soon as the user writes a formula
  - Gradual learning curve
    - You can always start with simple arithmetic

Hypothesis

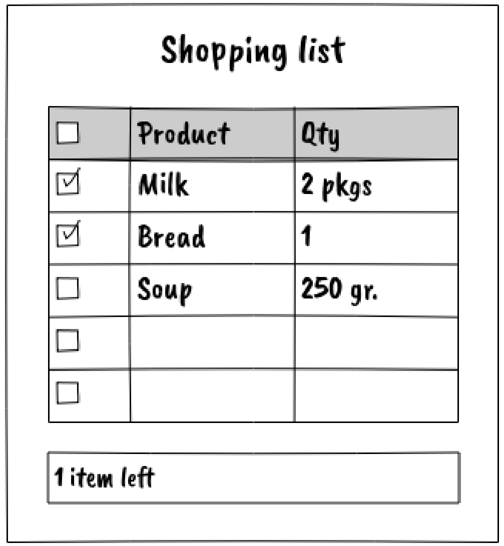
- If we want to build a low-code platform that normal users (aka citizen developers) can use, it should:
  - Be designed around the spreadsheet metaphor
  - Based on direct manipulation of concrete values,
  - Plus an application building environment based on live programming techniques.

Application examples – **Shopping list**

- Use and throw app



	A	B	C	D
1				
2				
3		**Product**	**Qty**	**Done**
4		Milk	2 pkgs	Yes
5		Bread	1	Yes
6		Soup	250 gr	
7				
8			1	Items left



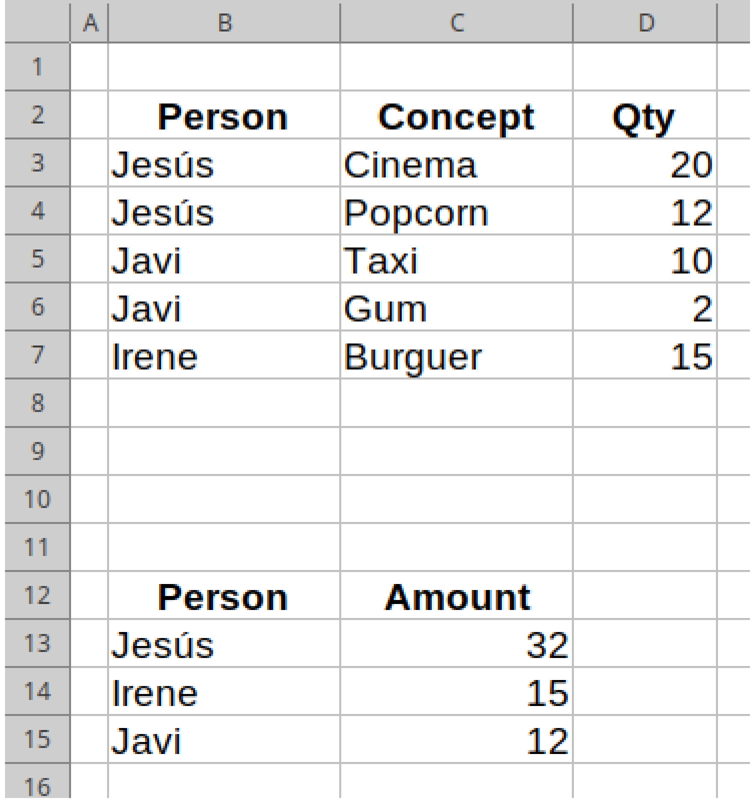
Shopping list

□	Product	Qty
☑	Milk	2 pkgs
☑	Bread	1
□	Soup	250 gr.
□		
□		

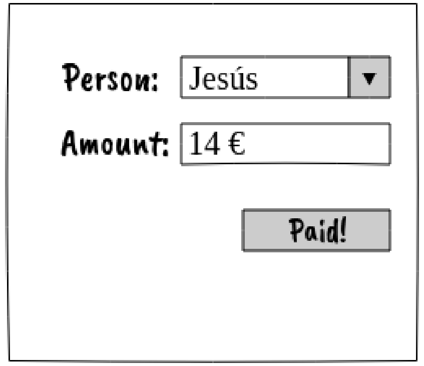
1 item left

Spreadsheets – **Sharing expenses**

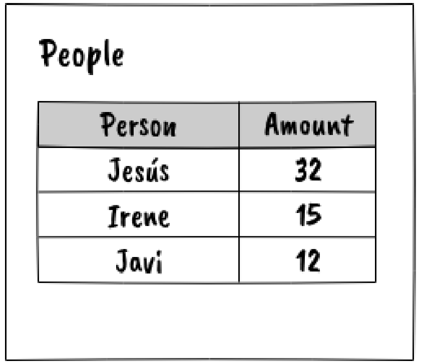
- Table 1: Register expenses
- Table 2: Summarize expenses



	A	B	C	D
1				
2		**Person**	**Concept**	**Qty**
3		Jesús	Cinema	20
4		Jesús	Popcorn	12
5		Javi	Taxi	10
6		Javi	Gum	2
7		Irene	Burguer	15
8				
9				
10				
11				
12		**Person**	**Amount**	
13		Jesús	32	
14		Irene	15	
15		Javi	12	
16				

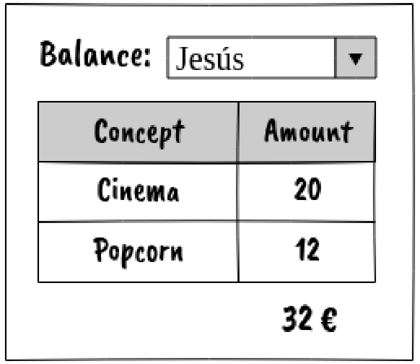


Person:	Jesús ▼
Amount:	14 €



People

Person	Amount
Jesús	32
Irene	15
Javi	12



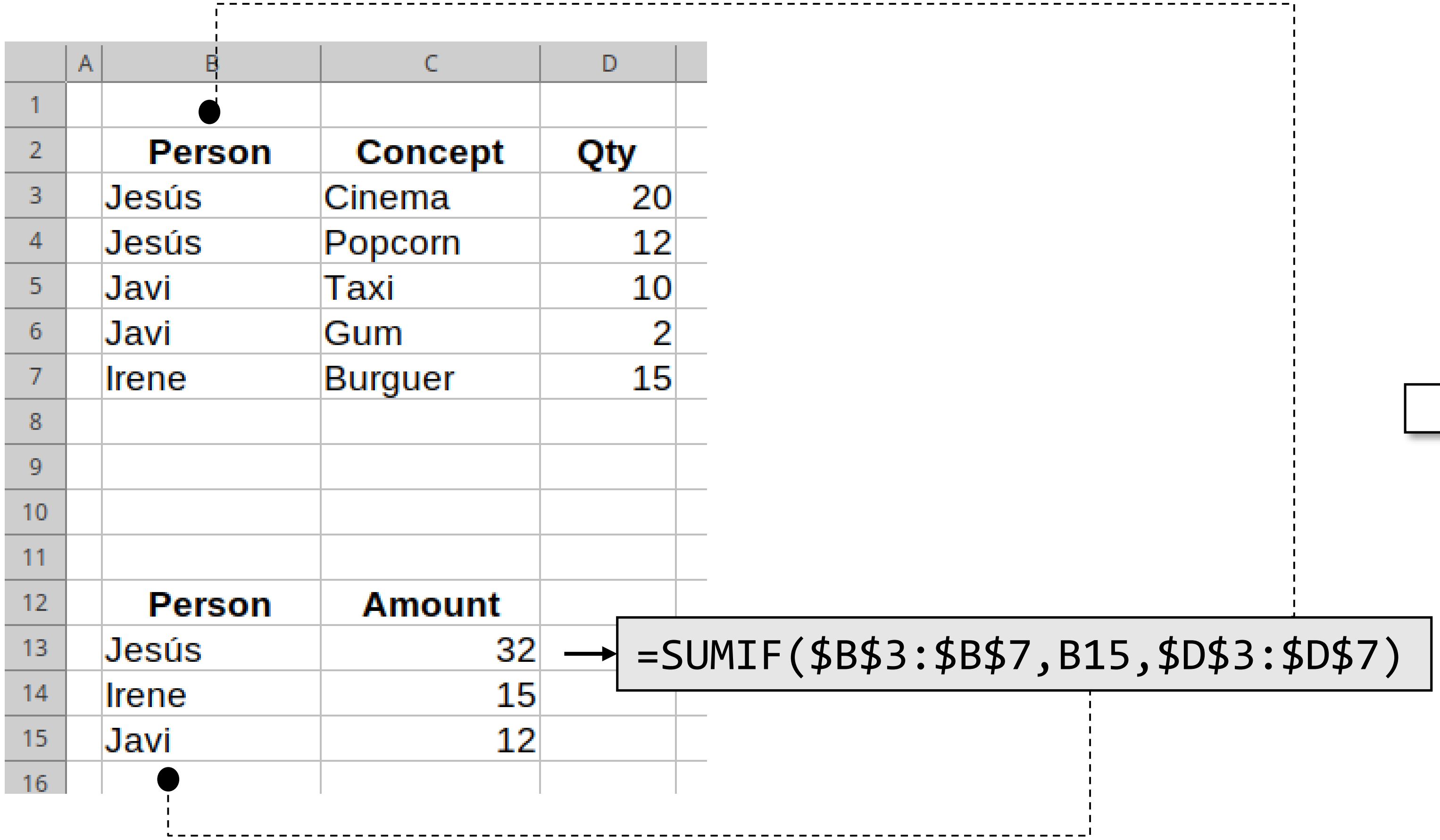
Balance: Jesús ▼

Concept	Amount
Cinema	20
Popcorn	12

32 €

Spreadsheets – **Sharing expenses**

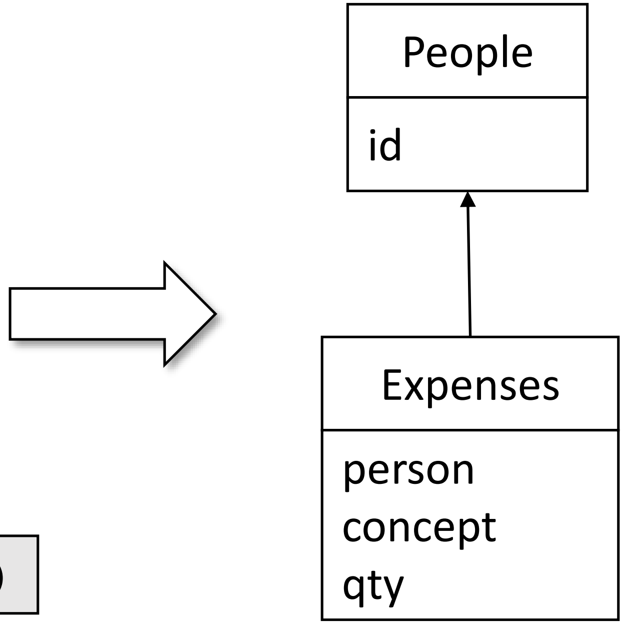
- There is a lot of implicit information



	A	B	C	D
1				
2		**Person**	**Concept**	**Qty**
3		Jesús	Cinema	20
4		Jesús	Popcorn	12
5		Javi	Taxi	10
6		Javi	Gum	2
7		Irene	Burguer	15
8				
9				
10				
11				
12		**Person**	**Amount**	
13		Jesús	32	
14		Irene	15	
15		Javi	12	
16				

```
=SUMIF($B$3:$B$7,B15,$D$3:$D$7)
```

Primary key



People

id

Expenses

person
concept
qty

Application examples – **Data analysis**

- Excel is used many times for data analysis
- Spreadsheets with large amounts of data are slow
- Reloading new data on top of an existing spreadsheet can be error prone
- Solution:
  - Work and polish with small data using the spreadsheet
  - Compile the spreadsheet to a script (e.g., Python/Pandas)

LowSheets

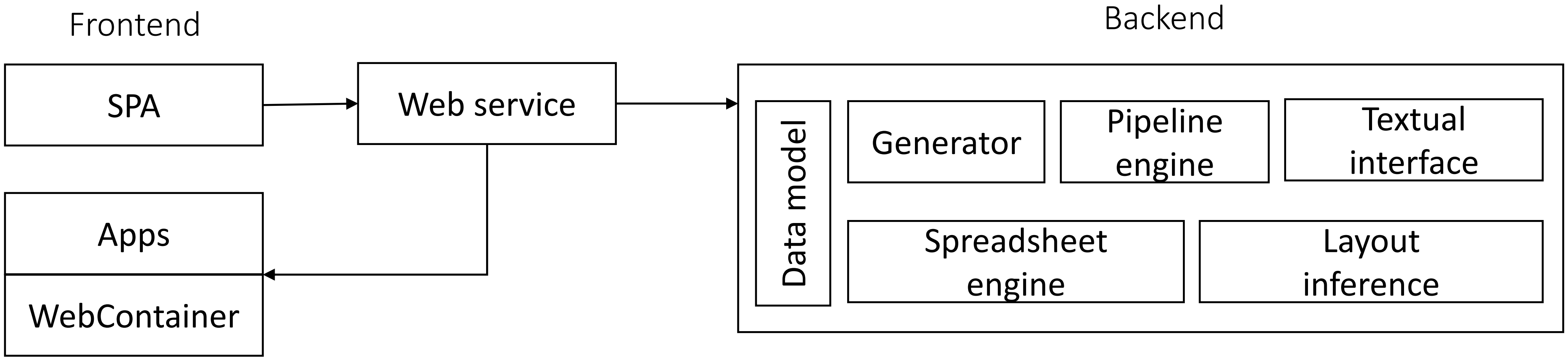
- A programming platform based on the spreadsheet metaphor
- Concepts:
  - Users write formulas “freely”
  - The system attempts to infer typical structures
    - No up-front design, the design emerges from the examples
    - Inference algorithm plus typing rules
  - A better Excel
    - There are tasks very difficult to achieve with Excel
    - Excel is about data analysis, but doesn’t excel at data manipulation
    - LowSheets introduces pipelines for data manipulation
  - Link the data to a user interface
  - Code generators and instant results

LowSheets

Demo time!

Architecture

- Monolithic web application
  - Frontend – Built with Svelte
  - Backend – Built with Spring



Conclusions

- Main take away:
    - **Spreadsheets can be used as the basis for building applications**
- **LowSheets**
    - A platform to build applications based on spreadsheets
    - So far:
        - Simple formula engine
        - Initial inference algorithm
        - Simple pipelines
        - Some generators
- Future:
    - Support more formulas
    - Take advantage of pipelines
    - Fix things!

Thanks!

Any questions?

7IOH

ProxyHands

MODELS & LANGUAGES LAB

http://models-lab.github.io

jesusc@um.es

@sanchezcuadrado

http://github.com/jesusc

http://sanchezcuadrado.es