SOFTWAREENGINEERING FOR PHYSICISTS

Polysim

IDEA

Working on a **realistic-ish** project

Encounter irreducible **complexity**

Learn how to become a **team**

OUR PROJECT

Provenance tracking for scientific simulations

First, we need to understand

Provenance

FAIR PRINCIPLES + DATA PROVENANCE

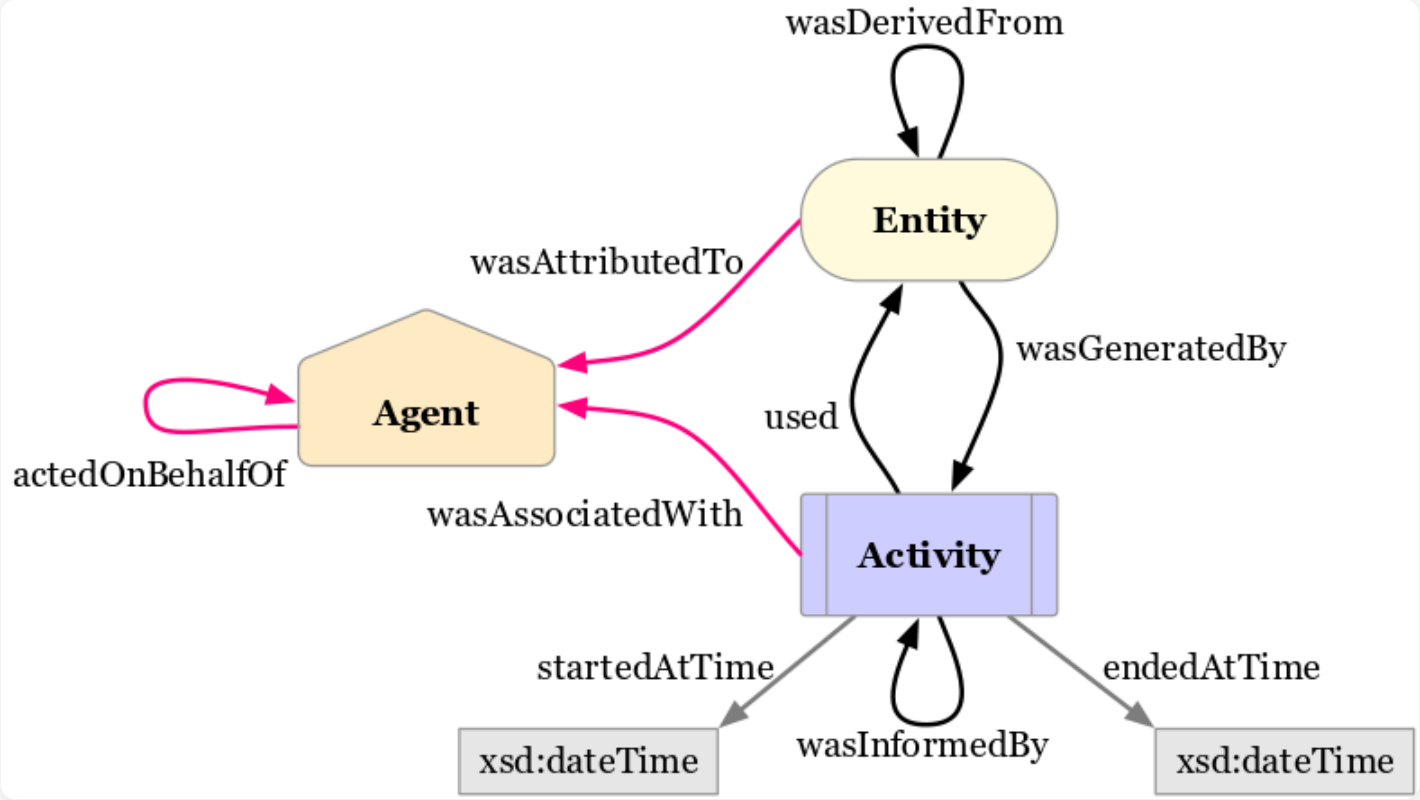
FAIR PRINCIPLES

- F – Findable (metadata, identifiers)
- A – Accessible (standards, protocols)
- I – Interoperable (tools, system)
- R – Reusable (structured)

WHAT IS DATA PROVENANCE?

- Where data comes from
- How it was produced
- Who produced it
- What transformations were applied

W3C PROV CONCEPTUAL MODEL

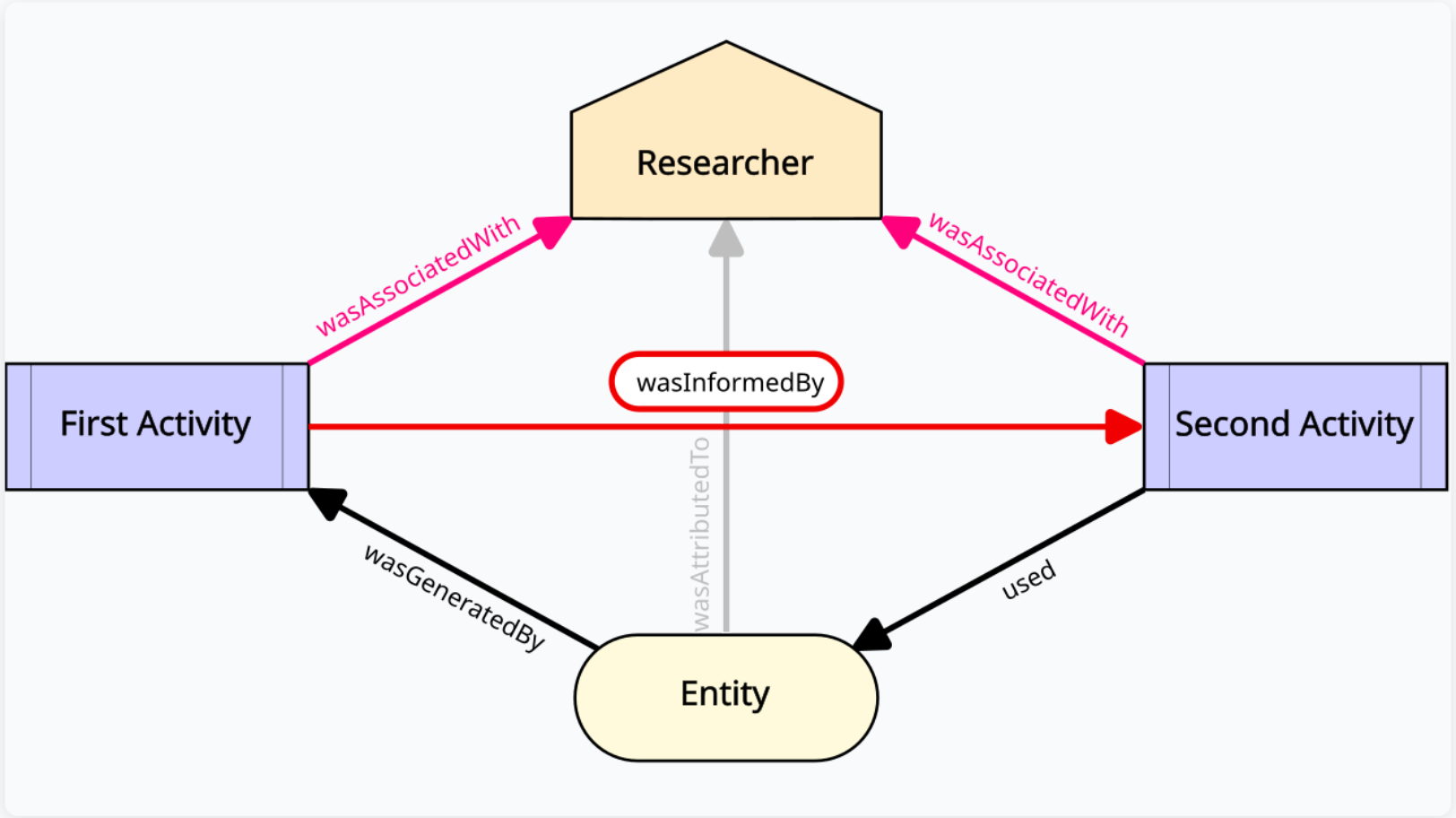


IMPORTANCE OF DATA PROVENANCE IN SCIENTIFIC SIMULATIONS

- Reproducibility
- Debugging & Error Tracking
- Collaboration & Sharing
- Data Reuse & FAIR Compliance
- Scientific Integrity & Validation
- Automation & Workflow Management

TRACKING DATA PROVENANCE IN SIMULATIONS

- Inputs & Outputs
- Processes
- Agents
- Environment


Researcher
wasAssociatedWith
wasAssociatedWith
wasInformedBy
First Activity
Second Activity
wasAttributedTo
wasGeneratedBy
used
Entity

Our answer to **FAIR**

POLYSIM

web application + CLI tool

to track provenance data for scientific simulations

HOW TO BUILD A WEB APP?

BY USING OTHER PEOPLE'S STUFF

Whether **technical**...

- **Docker** containerization
- **Caddy** reverse proxy, ssl
- **Next.js** frontend+backend via trpc
- **PostgreSQL** database
- **Keycloak** authentication

... or **organizational**

- **git, CI/CD** GitHub Actions
- **best practices** code reviews, testing
- **agile methodologies** Scrum/Kanban

AGILE, SCRUM & KANBAN

1. WHAT IMPLIES "AGILE" REALLY?

Agile is **not a tool**. It is a **mindset**.

> *"Responding to change over following a plan." -- Agile Manifesto (2001)*

1. WHAT IMPLIES "AGILE" REALLY?

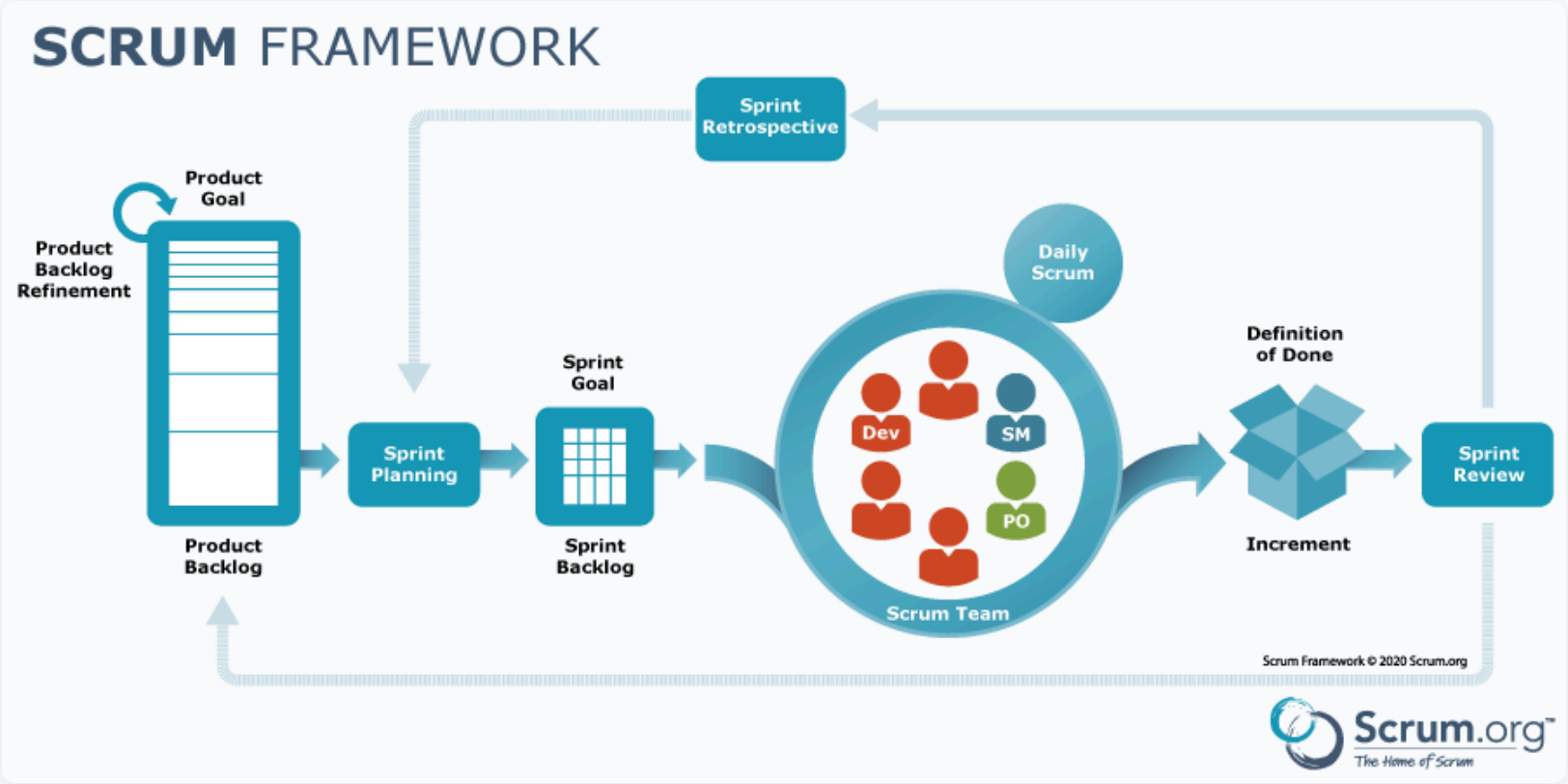
- **Uncertainty**
- **Increments**
- **People** Collaboration > Rigid Processes

2. THE TOOLS

SCRUM (STRUCTURE)

Rhythm & Roles

- **Time-boxed:** Fixed Sprints
- **Roles:** PO, Scrum Master, Developers
- **Loop:** Plan → Work → Review → Retro
- **Goal:** Alignment & Inspect/Adapt

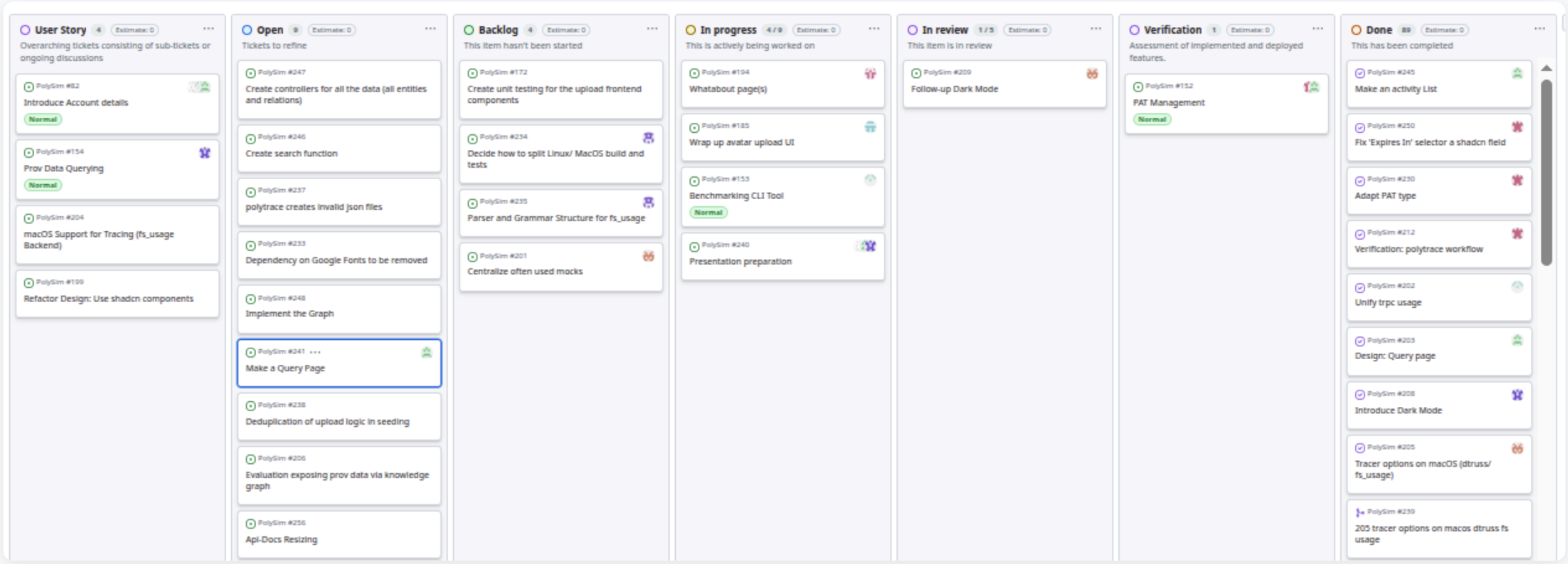

SCRUM FRAMEWORK
Sprint Retrospective
Product Goal
Product Backlog Refinement
Product Backlog
Sprint Planning
Sprint Goal
Sprint Backlog
Daily Scrum
Dev
SM
PO
Scrum Team
Definition of Done
Increment
Sprint Review
Scrum Framework © 2020 Scrum.org
Scrum.org
The Home of Scrum

KANBAN (FLOW)

Visualizing Work

- **Visual Board:** To Do → Doing → Done
- **Pull Principle:** No pushing tasks onto people
- **WIP Limits:** Stop starting, start finishing
- **Goal:** Optimize flow, reduce bottlenecks

OUR BOARD



Polysim

3. WHY DO WE DO THIS?

- Large Work Packages
- Team Coordination
- Fighting Complexity

4. HOW DID THIS PAN OUT FOR US?

FROM THEORY TO REALITY

THE REALITY CHECK

We started with strict **Scrum**, but...

- **Constraint 1:** Unplanned work
- **Constraint 2:** Time constraints
- **Constraint 3:** Part-time nature

Result: Mismatch between process and reality.

HOW WE REACTED

Jugaad

"Overcoming harsh constraints by improvising an effective solution using limited resources."

OUR SOLUTION: SCRUMBAN

Flexibility & Adaptability

WE COMBINED THE BEST OF BOTH WORLDS

- **From Scrum (Structure):**
  - "Daily" syncs & Retrospectives (Alignment)
  - Shared Responsibility
- **From Kanban (Flow):**
  - Continuous refinement
  - **Pull-based** work (when time allows)
  - Flexible prioritizations

COMMUNICATION IS KEY

Also for machine *-to→* human communication

```
    component: "uploadActivity"
[21:20:42.806] DEBUG (515918): Inserting activity: {"id":"1307c813a68ff65fc52b278a891e90789ccce0a7a08a547dd32f119bf8ff9
c36","label":"New Activity Informed","started_at":"2026-01-29T20:20:42.790Z","ended_at":"2026-01-29T20:20:42.790Z","met
adata":{}}
    component: "uploadActivity"
[21:20:42.807] DEBUG (515918): Upserting agent association for user c39da29a-3402-476e-8e52-a11356363bf4
    component: "uploadActivity"
[21:20:42.810] DEBUG (515918): Existing entity IDs: 18476adf36adf358cf264c60fa3e58aa957aa8f2891cc9ffaa6a9b5f879e8fda
    component: "uploadActivity"
[21:20:42.810] DEBUG (515918): Inserting 2 new entities.
    component: "uploadActivity"
[21:20:42.811] DEBUG (515918): Inserted 2 new entities. Found 1 existing entities.
    component: "uploadActivity"
[21:20:42.813] DEBUG (515918): Some entities were not inserted because they already exist.
                           we need to find if some of these entities are used in wasInformedBy relations

    expected: 1
    inserted: 2
[21:20:42.814] DEBUG (515918): wasGeneratedBy rows: [{"entity_id":"18476adf36adf358cf264c60fa3e58aa957aa8f2891cc9ffaa6a
9b5f879e8fda","activity_id":"846a6dd9cfbf18e9d8471cdac8aa39493b69a82ff4a87370cf79c7a75cbeee0e"}]
    component: "uploadActivity"
[21:20:42.815] DEBUG (515918): wasUsed rows: [{"activity_id":"f98bde4965b63be2168da16d4440f36137bbbc19e8cb81b7e4beb35d7
1abf2b1","entity_id":"18476adf36adf358cf264c60fa3e58aa957aa8f2891cc9ffaa6a9b5f879e8fda","role":"input"}]
    component: "uploadActivity"
```

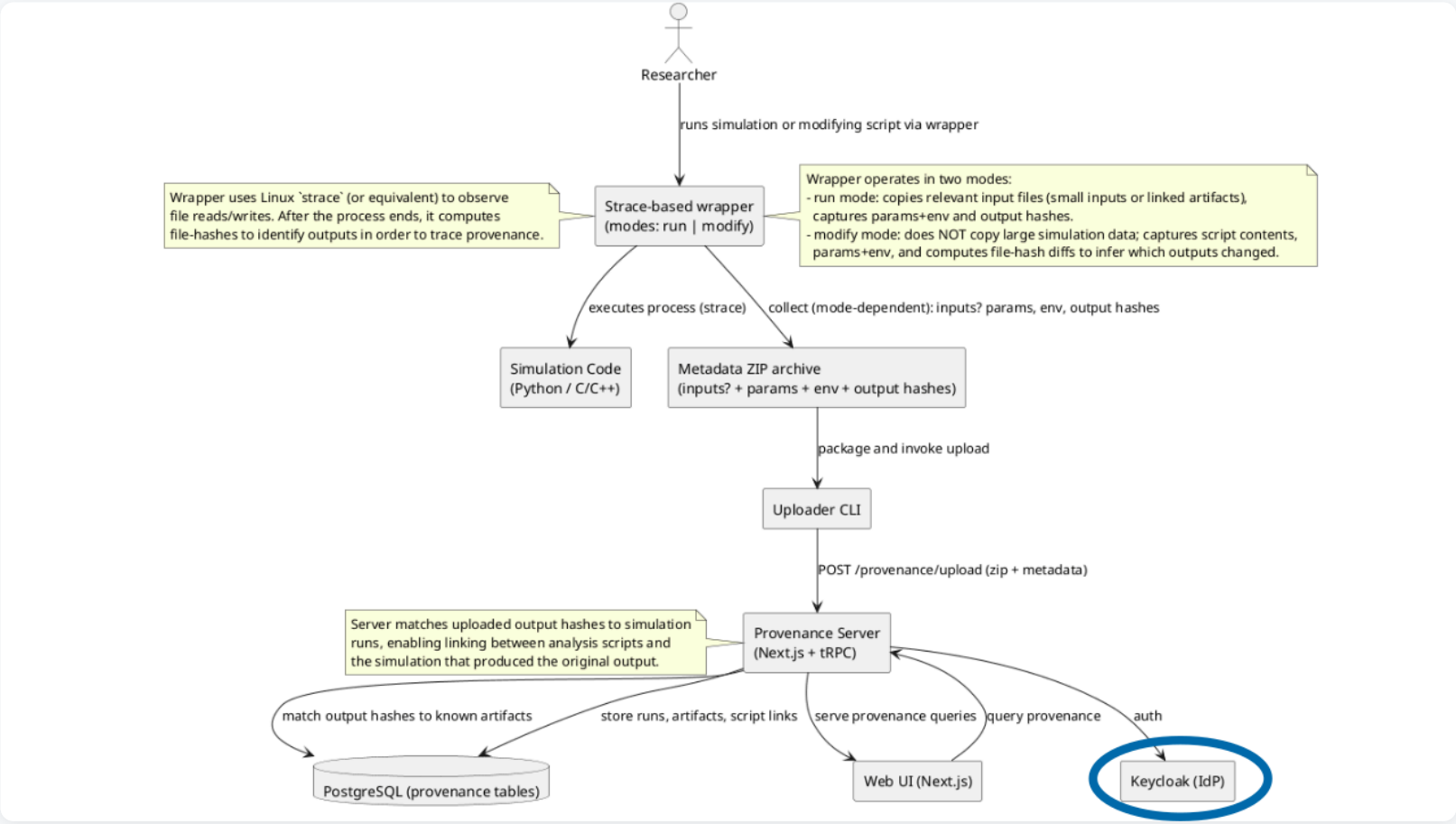
aka **Logging**

WHAT NEXT?

We **know** what we want to build

We **know** how we organize ourselves.

Now we need to **plan** the app.


Researcher
runs simulation or modifying script via wrapper
Wrapper uses Linux `strace` (or equivalent) to observe file reads/writes. After the process ends, it computes file-hashes to identify outputs in order to trace provenance.
Strace-based wrapper (modes: run | modify)
Wrapper operates in two modes:
- run mode: copies relevant input files (small inputs or linked artifacts), captures params+env and output hashes.
- modify mode: does NOT copy large simulation data; captures script contents, params+env, and computes file-hash diffs to infer which outputs changed.
executes process (strace)
collect (mode-dependent): inputs? params, env, output hashes
Simulation Code (Python / C/C++)
Metadata ZIP archive (inputs? + params + env + output hashes)
package and invoke upload
Uploader CLI
POST /provenance/upload (zip + metadata)
Server matches uploaded output hashes to simulation runs, enabling linking between analysis scripts and the simulation that produced the original output.
Provenance Server (Next.js + tRPC)
match output hashes to known artifacts
store runs, artifacts, script links
serve provenance queries
query provenance
auth
PostgreSQL (provenance tables)
Web UI (Next.js)
Keycloak (IdP)

WHY KEYCLOAK?

- **Trust the Pros:** Industry standard (BMW, Cisco, CERN)
- **Open Source:** Audited by thousands of developers
- **Focus on Features:** Skip weeks of auth work, build PolySim instead

SETTING UP KEYCLOAK WITH NEXT.JS

The Infrastructure

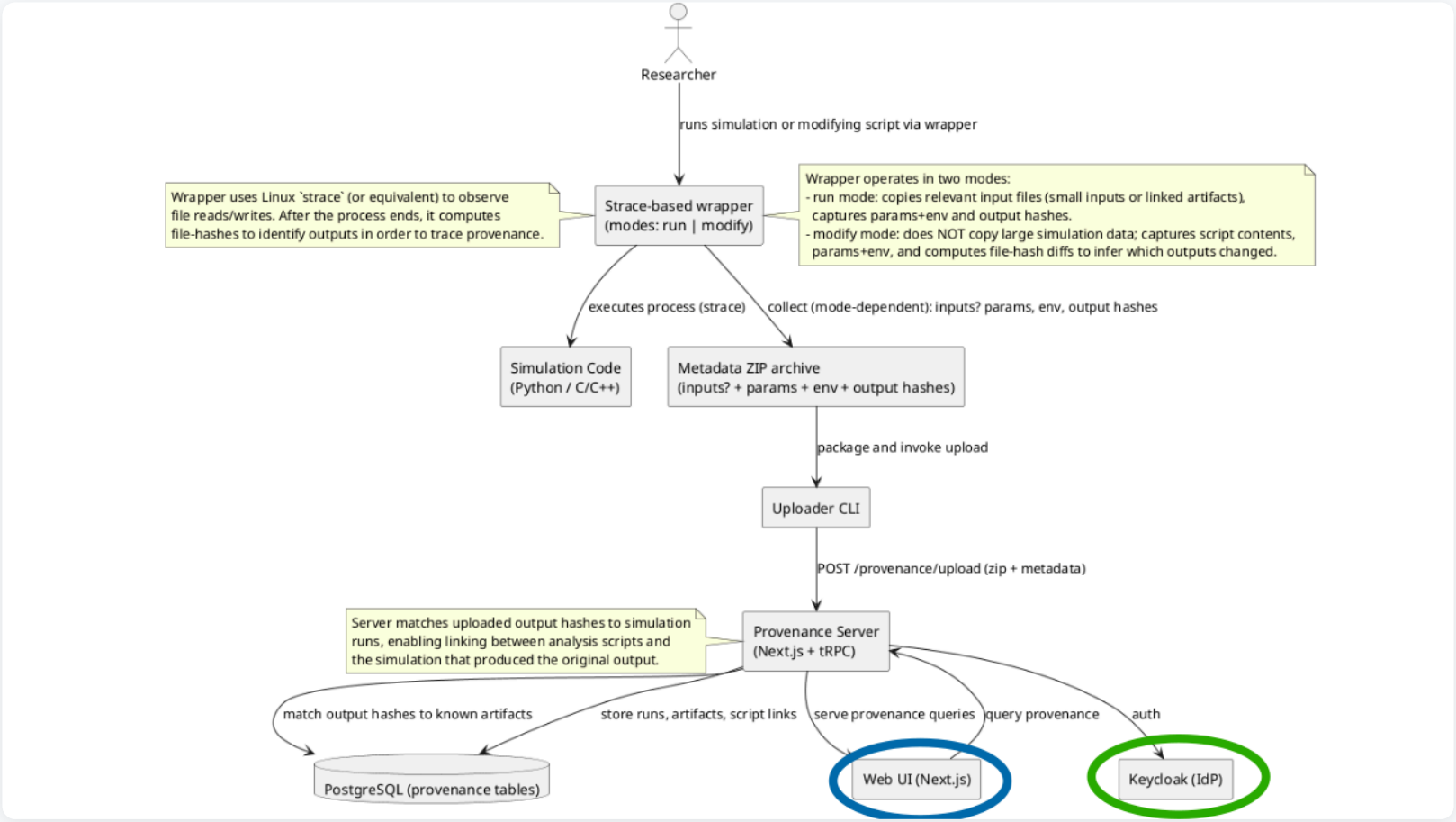
- Keycloak runs in its own Docker container
- Separate from the web app
- Port `:8080`

The Bridge

- `next-auth` connects Next.js to Keycloak
- Acts like a secure "translator"
- Handles redirects and

THE LOGIN FLOW

Step	Action
1️⃣	User clicks "Login"
2️⃣	Redirected to Keycloak login page
3️⃣	Keycloak verifies credentials
4️⃣	JWT Token returned
5️⃣	PolySim grants access ✅


Researcher
runs simulation or modifying script via wrapper
Wrapper uses Linux `strace` (or equivalent) to observe file reads/writes. After the process ends, it computes file-hashes to identify outputs in order to trace provenance.
Strace-based wrapper (modes: run | modify)
Wrapper operates in two modes:
- run mode: copies relevant input files (small inputs or linked artifacts), captures params+env and output hashes.
- modify mode: does NOT copy large simulation data; captures script contents, params+env, and computes file-hash diffs to infer which outputs changed.
executes process (strace)
collect (mode-dependent): inputs? params, env, output hashes
Simulation Code (Python / C/C++)
Metadata ZIP archive (inputs? + params + env + output hashes)
package and invoke upload
Uploader CLI
POST /provenance/upload (zip + metadata)
Server matches uploaded output hashes to simulation runs, enabling linking between analysis scripts and the simulation that produced the original output.
Provenance Server (Next.js + tRPC)
match output hashes to known artifacts
store runs, artifacts, script links
serve provenance queries
query provenance
auth
PostgreSQL (provenance tables)
Web UI (Next.js)
Keycloak (IdP)

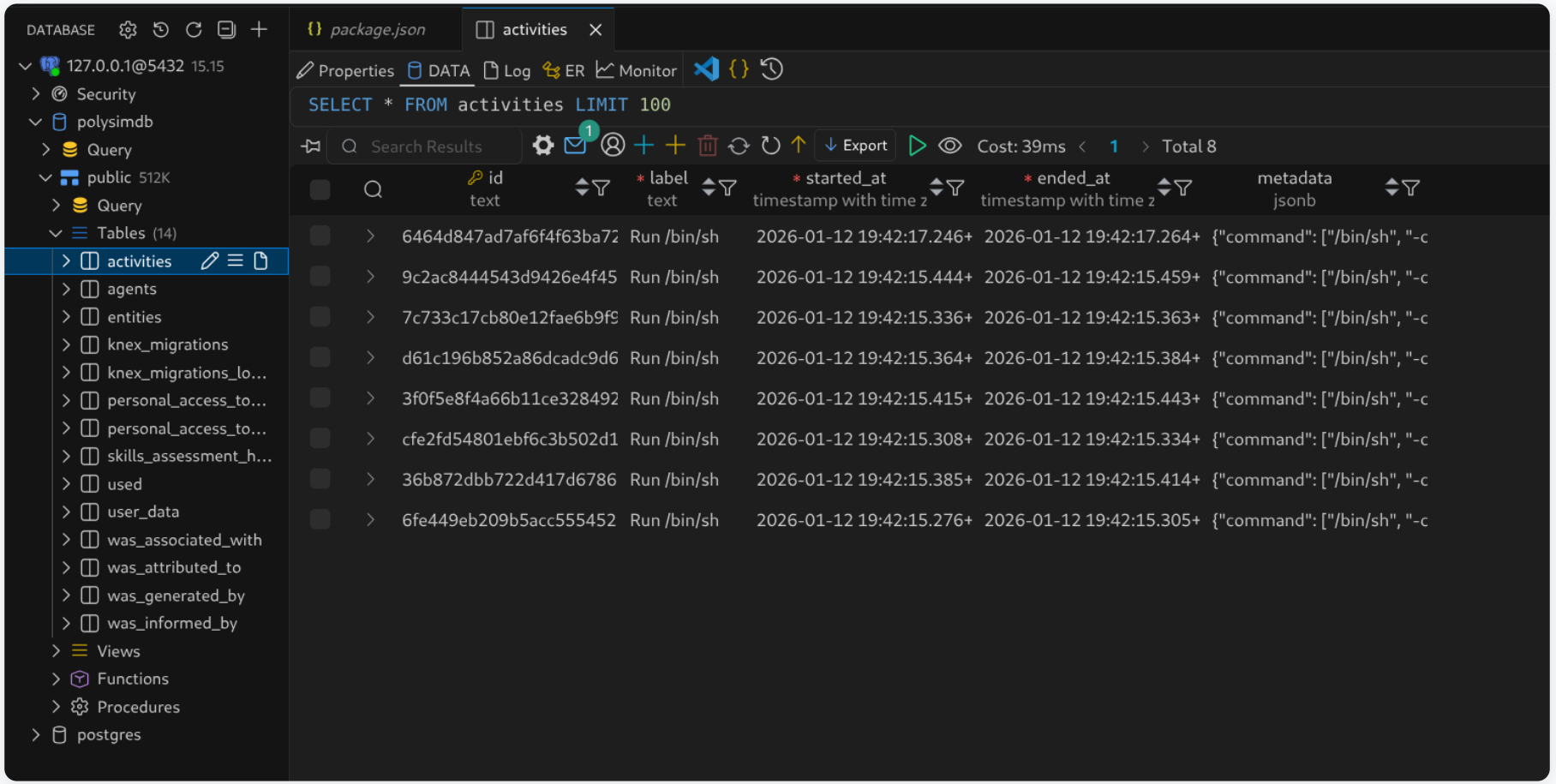
REACT

Some things are easier clicked than typed

VISUALIZATION

- Good UI Matters
- Data without good Visuals is a bit useless
- The website should look nice and be easy to use

```
{
  "activity": {
    "endedAt": 1769462822274,
    "id": "556e0c71d1bd84c1ba3097b1d10a6ae70c72e23a4893cc2c9d8d3aaade76fd8b",
    "label": "Run make",
    "metadata": {
      "command": [
        "make"
      ]
    },
    "startedAt": 1769462808653
  },
  "entities": [
    {
      "createdAt": 1769462812831,
      "id": "52e83be3b5f9568ee29ebd6e8dd870cd88dee01514eda57b4750fc359125b85d",
      "label": "lmroman9-regular.lua",
      "metadata": {
        "accesses": [
          {
            "metadata": {},
            "pid": 204677,
            "role": "output"
          },
          {
            "metadata": {},
            "pid": 204677,
            "role": "input"
          }
        ],
        "path": "/home/enno/.local/texlive/2025/texmf-var/luatex-cache/generic/fonts/otl/lmroman9-regular.lua"
      },
      "role": "output"
    },
    {
      "createdAt": 1769462812539,
      "id": "47ce72619b7be7ebe6554e7619b524288b78ddc4768f4e4632cc99a6f321577f",
      "label": "lmroman5-regular.lua",
      "metadata": {
        "accesses": [
          {
            "metadata": {},
            "pid": 204677,
            "role": "output"
          },
          {
            "metadata": {},
            "pid": 204677,
            "role": "input"
          }
        ],
        "path": "/home/enno/.local/texlive/2025/texmf-var/luatex-cache/generic/fonts/otl/lmroman5-regular.lua"
      },
      "role": "output"
    },
    {
      "createdAt": 1769462812483,
      "id": "261a511244475d9051223424aeba888db784c522ecf2ec4aeedc0b8681fcde5b",
      "label": "lmroman8-regular.lua",
      "metadata": {
        "accesses": [
          {
            "metadata": {},
            "pid": 204677,
            "role": "output"
          },
```


DATABASE
127.0.0.1@5432 15.15
Security
polysimdb
Query
public 512K
Query
Tables (14)
activities
agents
entities
knex_migrations
knex_migrations_lo...
personal_access_to...
personal_access_to...
skills_assessment_h...
used
user_data
was_associated_with
was_attributed_to
was_generated_by
was_informed_by
Views
Functions
Procedures
postgres
package.json
activities
Properties
DATA
Log
ER
Monitor
SELECT * FROM activities LIMIT 100
Search Results
Export
Cost: 39ms
1
Total 8
id text
label text
started_at timestamp with time z
ended_at timestamp with time z
metadata jsonb
6464d847ad7af6f4f63ba72 Run /bin/sh 2026-01-12 19:42:17.246+ 2026-01-12 19:42:17.264+ {"command": ["/bin/sh", "-c
9c2ac8444543d9426e4f45 Run /bin/sh 2026-01-12 19:42:15.444+ 2026-01-12 19:42:15.459+ {"command": ["/bin/sh", "-c
7c733c17cb80e12fae6b9f9 Run /bin/sh 2026-01-12 19:42:15.336+ 2026-01-12 19:42:15.363+ {"command": ["/bin/sh", "-c
d61c196b852a86dcadc9d6 Run /bin/sh 2026-01-12 19:42:15.364+ 2026-01-12 19:42:15.384+ {"command": ["/bin/sh", "-c
3f0f5e8f4a66b11ce328492 Run /bin/sh 2026-01-12 19:42:15.415+ 2026-01-12 19:42:15.443+ {"command": ["/bin/sh", "-c
cfe2fd54801ebf6c3b502d1 Run /bin/sh 2026-01-12 19:42:15.308+ 2026-01-12 19:42:15.334+ {"command": ["/bin/sh", "-c
36b872dbb722d417d6786 Run /bin/sh 2026-01-12 19:42:15.385+ 2026-01-12 19:42:15.414+ {"command": ["/bin/sh", "-c
6fe449eb209b5acc555452 Run /bin/sh 2026-01-12 19:42:15.276+ 2026-01-12 19:42:15.305+ {"command": ["/bin/sh", "-c

WE NEED TO LOOK AT THE WEBSITE!

HOW DO WE VISUALIZE THINGS IN OUR WEBSITE?

- shadcn components based on React
  - Responsive
  - Design toolbox
- Tables
- Graph view of Provenance

CHALLENGES

- Dark mode
- Different component sources

IMPLEMENTING AVATAR UPLOAD

The Evolution: From File Input → Dialog

Before	After
Generic Shadcn file component	Dedicated Dialog modal
No preview	Live preview before upload
Confusing UX	Clear, focused workflow

AVATAR UPLOAD WORKFLOW

Click avatar (sidebar) → Account Page → "Upload Avatar" button → Dialog opens → Preview image → Confirm upload

STORING AVATARS IN THE DATABASE

Image → Base64 String → Database

- Convert image to **Base64** text format
- Send via **tRPC mutation**
- Store directly in PostgreSQL
- **No external cloud storage needed**

RETRIEVING AVATARS

Database → Base64 String → Browser Display

- tRPC query fetches the Base64 string
- Browser converts it back to an image
- Shown in sidebar and profile page
- **Simple and efficient ✅**

EXPERIENCE: COMPLEXITY IN PRACTICE

What looks simple becomes surprisingly tricky

Avatar upload seemed straightforward, but required:

- Base64 encoding/decoding
- Real-time UI updates
- Proper error handling

Our solution: Iterate, test, refine

BRIDGING FRONTEND & BACKEND

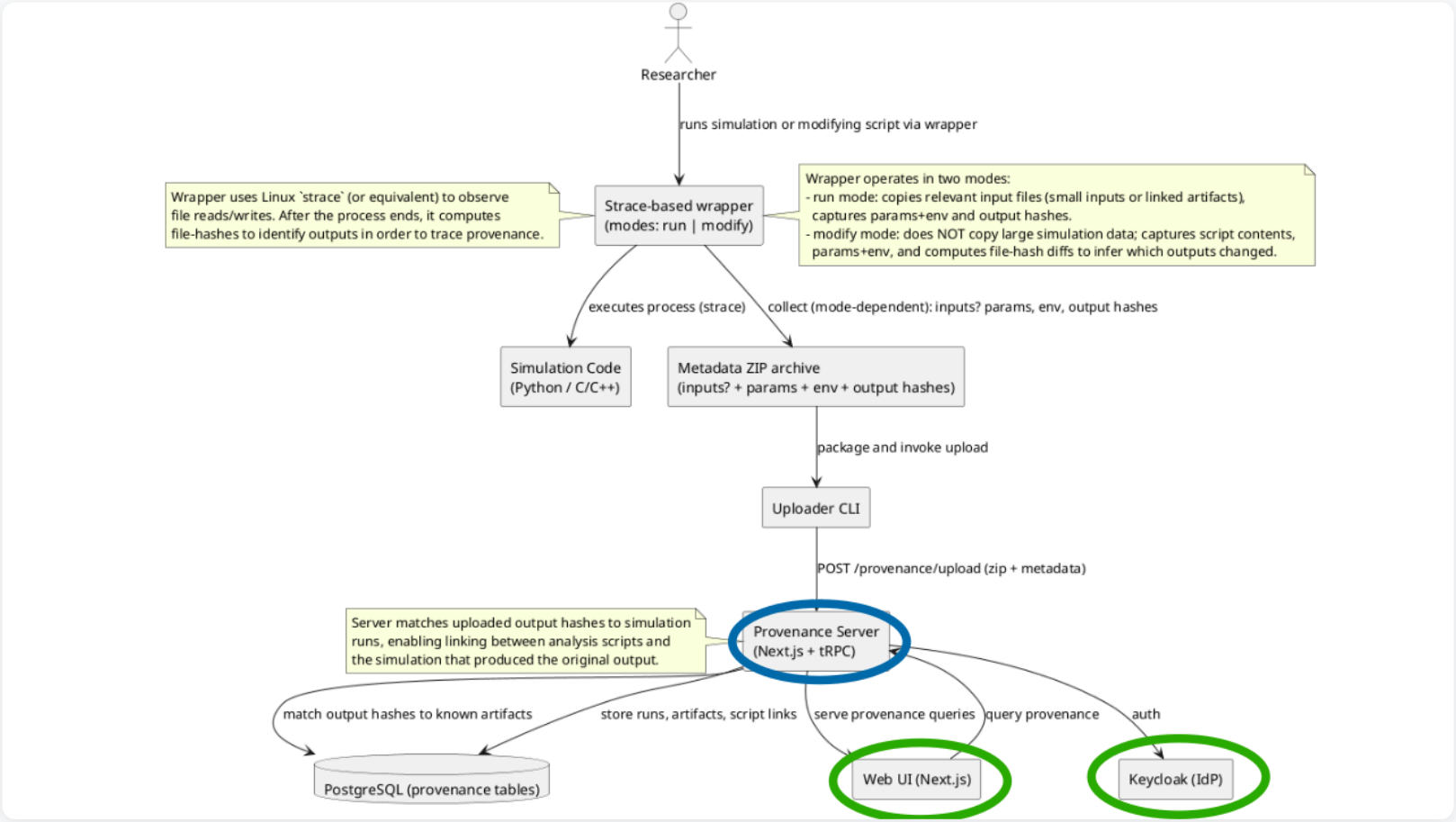
NEXT.JS + TRPC

Type-Safe Full-Stack

Next.js = Folder → Route mapping

trpc = frontend↔backend as simple as (typed) function

easy internal and public API development


Researcher
runs simulation or modifying script via wrapper
Wrapper uses Linux `strace` (or equivalent) to observe file reads/writes. After the process ends, it computes file-hashes to identify outputs in order to trace provenance.
Strace-based wrapper (modes: run | modify)
Wrapper operates in two modes:
- run mode: copies relevant input files (small inputs or linked artifacts), captures params+env and output hashes.
- modify mode: does NOT copy large simulation data; captures script contents, params+env, and computes file-hash diffs to infer which outputs changed.
executes process (strace)
collect (mode-dependent): inputs? params, env, output hashes
Simulation Code (Python / C/C++)
Metadata ZIP archive (inputs? + params + env + output hashes)
package and invoke upload
Uploader CLI
POST /provenance/upload (zip + metadata)
Server matches uploaded output hashes to simulation runs, enabling linking between analysis scripts and the simulation that produced the original output.
Provenance Server (Next.js + tRPC)
match output hashes to known artifacts
store runs, artifacts, script links
serve provenance queries
query provenance
auth
PostgreSQL (provenance tables)
Web UI (Next.js)
Keycloak (IdP)

PERSONAL ACCESS TOKENS (PATS)

AUTHENTICATION FOR THE PUBLIC API

For CLI tools like **PolyTrace**, we need a secure authentication method:

- No interactive login possible
- Classic solution: **Personal Access Tokens**

WHAT ARE PERSONAL ACCESS TOKENS?

A **PAT** is a long-lived token for API authentication.

Key characteristics:

- Generated by backend, shown **once**
- Stored **hashed** on server
- Sent via `Authorization` header
- Alternative to username/password

WHY USE PATS?

NON-INTERACTIVE AUTHENTICATION

Automated systems (CLI tools, scripts) cannot perform interactive logins.

LEAST-PRIVILEGE ACCESS

PATs can be scoped with specific permissions.

SAFE REVOCATION

Compromising a PAT doesn't require resetting the user's password.

TOKEN GENERATION WORKFLOW

TOKEN GENERATION WORKFLOW

1. Random bytes

64 bytes → hex

TOKEN GENERATION WORKFLOW

1. Random bytes

64 bytes → hex

↑

→

2. Token

`a3f2...8d1c`

TOKEN GENERATION WORKFLOW

1. Random bytes

64 bytes → hex

↑

→

2. Token

`a3f2...8d1c`

↓

3. HMAC

+ secret "blablabla"

→ `7c8e...`

TOKEN GENERATION WORKFLOW

1. Random bytes
64 bytes → hex

→

2. Token
`a3f2...8d1c`

↓

3. HMAC
+ secret "blablabla"
→ `7c8e...`

←

4. SHA-256
`2f1a...` (stored in DB)

↑

SECURITY: DOUBLE HASHING

Why HMAC + SHA-256?

HMAC with server secret

- Secret in env vars only
- Never exposed to clients/DB
- Prevents rainbow tables

SHA-256 hash

- Stored in PostgreSQL
- One-way, cannot recover token

CODE IMPLEMENTATION

```
const token = crypto.randomBytes(64).toString('hex');
const hmac = crypto.createHmac('sha256', process.env.SERVER_SECRET);
const salted = hmac.update(token).digest('hex');
const hash = crypto.createHash('sha256').update(salted).digest('hex');
// Store hash in DB, show token to user ONCE
```

HOW WE USE PATS IN POLYSIM

1. User generates PAT via web interface
2. Token shown **once** (store locally)
3. PolyTrace sends PAT in requests
4. Server validates and grants access

Use case: Upload provenance from PolyTrace

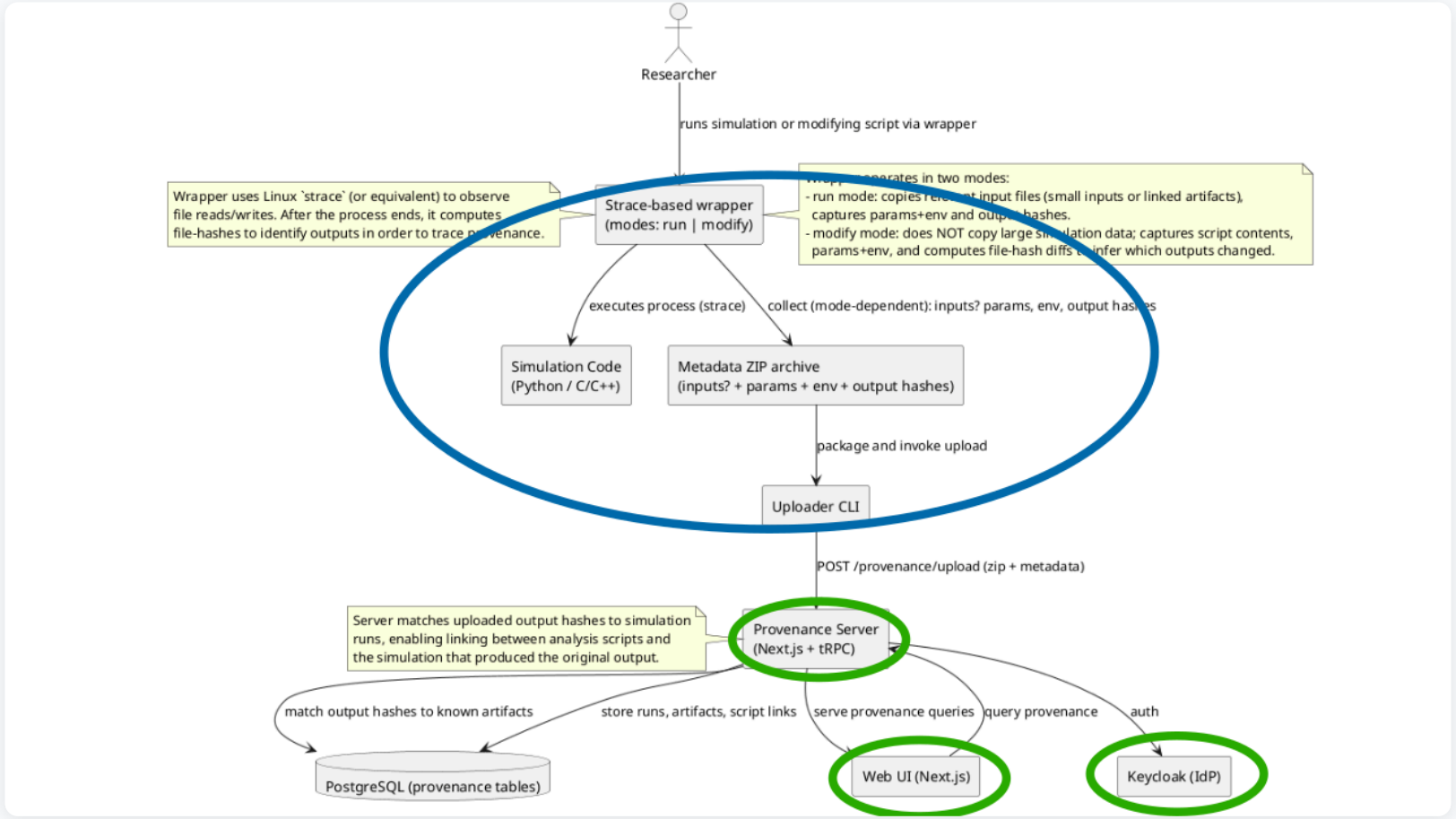
BACKEND AUTHENTICATION FLOW

1. Client sends PAT in `Authorization` header
2. Server applies HMAC + SHA-256
3. Compare with stored hash in PostgreSQL
4. If valid: Inject user into tRPC context
5. Procedures use context for auth

```
const userId = getUserIdFromContext(ctx);
```

public API secured with **PATs**!

→ http://localhost:3000/api-doc


Researcher
runs simulation or modifying script via wrapper
Wrapper uses Linux `strace` (or equivalent) to observe file reads/writes. After the process ends, it computes file-hashes to identify outputs in order to trace provenance.
Strace-based wrapper (modes: run | modify)
- run mode: copies relevant input files (small inputs or linked artifacts), captures params+env and output hashes.
- modify mode: does NOT copy large simulation data; captures script contents, params+env, and computes file-hash diffs to infer which outputs changed.
executes process (strace)
collect (mode-dependent): inputs? params, env, output hashes
Simulation Code (Python / C/C++)
Metadata ZIP archive (inputs? + params + env + output hashes)
package and invoke upload
Uploader CLI
POST /provenance/upload (zip + metadata)
Server matches uploaded output hashes to simulation runs, enabling linking between analysis scripts and the simulation that produced the original output.
Provenance Server (Next.js + tRPC)
match output hashes to known artifacts
store runs, artifacts, script links
serve provenance queries
query provenance
auth
PostgreSQL (provenance tables)
Web UI (Next.js)
Keycloak (IdP)

POLYTRACE AND UPLOADER

WHAT IS POLYTRACE?

A CLI tool to record filesystem activity

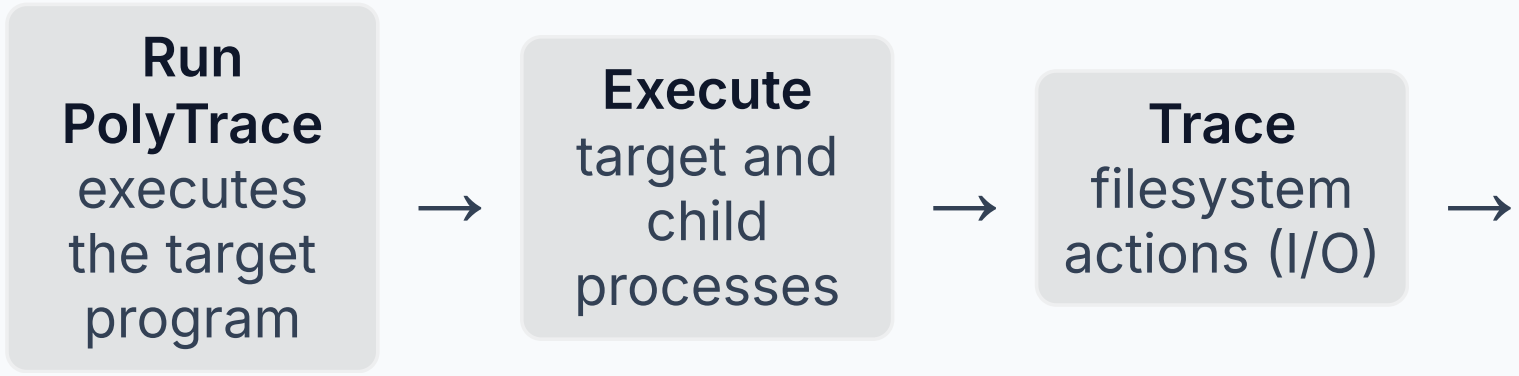
Run PolyTrace executes the target program

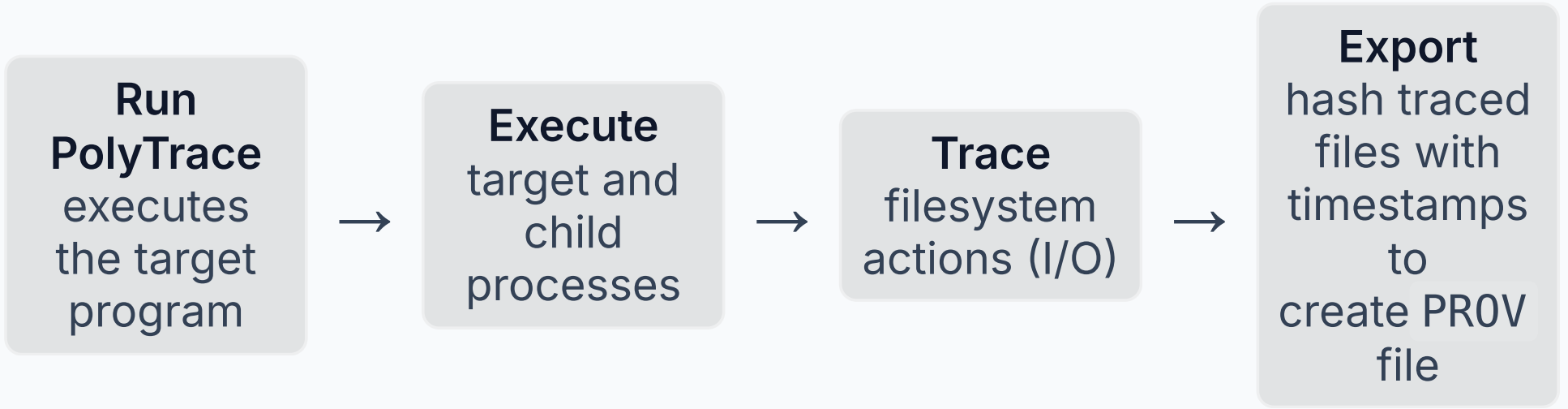
Run PolyTrace executes the target program →

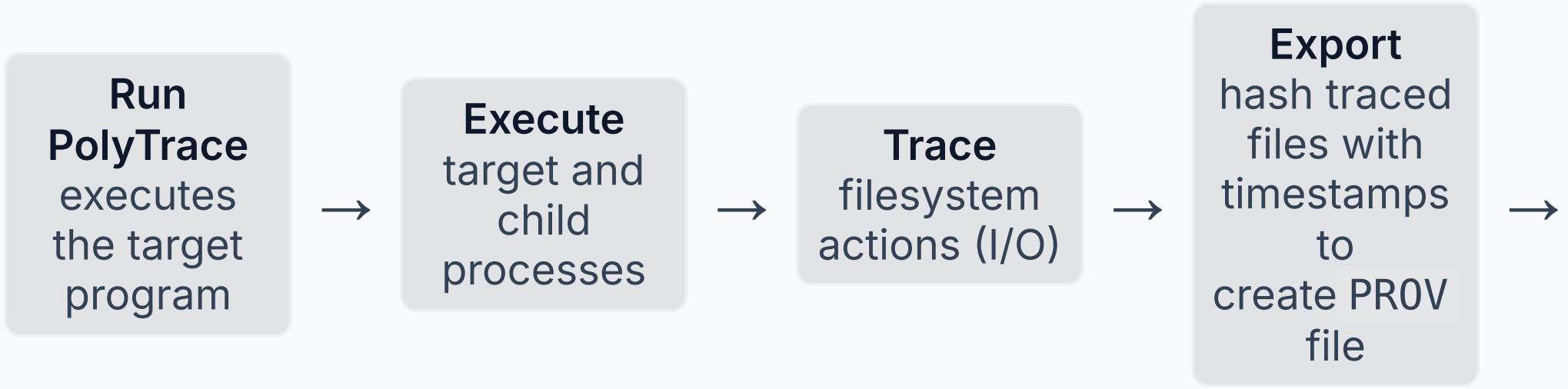
Run PolyTrace executes the target program → **Execute** target and child processes

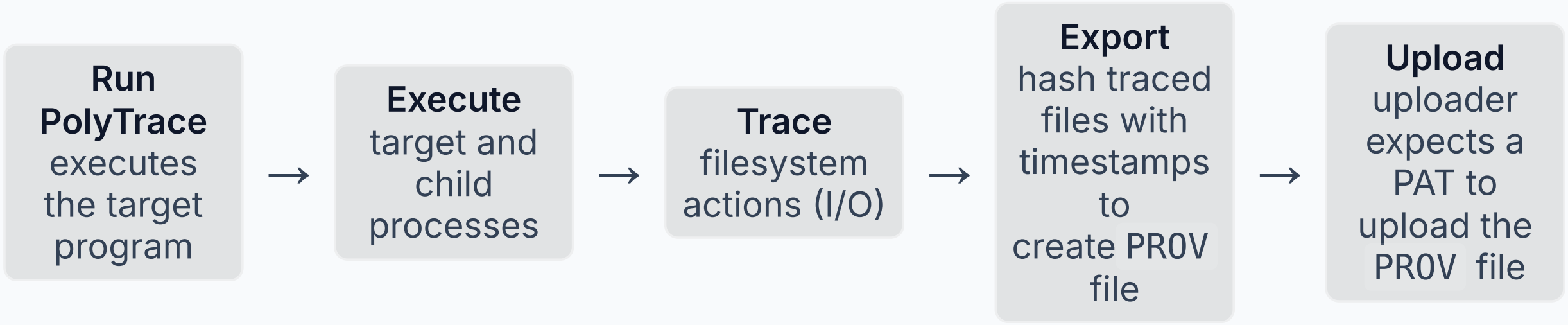
Run PolyTrace executes the target program → **Execute** target and child processes →


Run PolyTrace executes the target program
Execute target and child processes
Trace filesystem actions (I/O)


Run PolyTrace executes the target program
Execute target and child processes
Trace filesystem actions (I/O)


Run PolyTrace executes the target program
Execute target and child processes
Trace filesystem actions (I/O)
Export hash traced files with timestamps to create PR0V file


Run PolyTrace executes the target program
Execute target and child processes
Trace filesystem actions (I/O)
Export hash traced files with timestamps to create PR0V file


Run PolyTrace executes the target program
Execute target and child processes
Trace filesystem actions (I/O)
Export hash traced files with timestamps to create PR0V file
Upload uploader expects a PAT to upload the PR0V file

PLATFORM SUPPORT

OS	How PolyTrace traces filesystem activity
Linux	Uses `strace`
Windows	Uses `strace` via **WSL**
macOS	Uses `fs_usage` with some limitations

POLYTRACE PIPELINE

POLYTRACE PIPELINE

PolyTrace executes target program

```
#!/bin/sh
BASE_DIR=$(cd "$(dirname "$0")" && pwd)
OUT="$BASE_DIR/tmp/simple_run_out"
rm -f "$OUT"
cat /etc/ld.so.cache > /dev/null 2>/dev/null || true
echo hello > "$OUT"
sleep 0.01
exit 0
```

POLYTRACE PIPELINE

PolyTrace executes target program → **strace output**

```
#!/bin/sh
BASE_DIR=$(cd "$(dirname "$0")" && pwd)
OUT="$BASE_DIR/tmp/simple_run_out"
rm -f "$OUT"
cat /etc/ld.so.cache > /dev/null 2>/dev/null || true
echo hello > "$OUT"
sleep 0.01
exit 0
```

```
1769772754.266682 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=960733, si_uid=1000
1769772754.266720 chdir("/home/tobias/Projekte/PolySim/tools/polytrace/test/fixtures") = 0
1769772754.266818 +++ exited with 0 +++
1769772754.265826 newfstatat(AT_FDCWD, "/home/tobias/.sdkman/candidates/java/current/bin/dirnam
1769772754.265864 newfstatat(AT_FDCWD, "/home/tobias/.nvm/versions/node/v24.9.0/bin/dirname", 0
1769772754.265880 newfstatat(AT_FDCWD, "/home/tobias/.local/bin/dirname", 0x7ffc1d955610, 0) =
1769772754.265894 newfstatat(AT_FDCWD, "/usr/local/sbin/dirname", 0x7ffc1d955610, 0) = -1 ENOEN
1769772754.265908 newfstatat(AT_FDCWD, "/usr/local/bin/dirname", 0x7ffc1d955610, 0) = -1 ENOENT
1769772754.265922 newfstatat(AT_FDCWD, "/usr/sbin/dirname", 0x7ffc1d955610, 0) = -1 ENOENT (Dat
1769772754.265935 newfstatat(AT_FDCWD, "/usr/bin/dirname", {st_mode=S_IFREG|0755, st_size=35208
1769772754.265955 execve("/usr/bin/dirname", ["dirname", "test/fixtures/simple_run.sh"], 0x5cd2
1769772754.266179 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Datei oder Verzeichnis nicht
1769772754.266194 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
1769772754.266242 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
1769772754.266515 openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
```

POLYTRACE PIPELINE

PolyTrace executes target program → **fs_usage output**

```
#!/bin/sh
BASE_DIR=$(cd "$(dirname "$0")" && pwd)
OUT="$BASE_DIR/tmp/simple_run_out"
rm -f "$OUT"
cat /etc/ld.so.cache > /dev/null 2>/dev/null || true
echo hello > "$OUT"
sleep 0.01
exit 0
```

```
17:31:54.814998 fsgetpath /usr/lib/dyld 0.000038 simple_run.6475663
17:31:54.815776 fsgetpath /usr/lib/dyld 0.000019 simple_run.6475663
17:31:54.816178 open F=3 (R******\_\_******) . 0.000098 simple_run.6475663
17:31:54.816182 fcntl F=3 <getpath> 0.000004 simple_run.6475663
17:31:54.816187 close F=3 0.000005 simple_run.6475663
17:31:54.816197 fsgetpath /Users/efemyuksel/PolySim/tools/polytrace/test/fixtures/simple_run 0.
17:31:54.816213 fsgetpath /usr/lib/dyld 0.000015 simple_run.6475663
17:31:54.816345 open F=3 (R****\_\_\_****f**) 0.000030 simple_run.6475663
17:31:54.816396 openat F=4 (R****\_\_\_\_******) [3]/../../System/Volumes/Preboot/Cryptexes/OS
17:31:54.816415 fstatat64 [4]/System/Library/dyld 0.000012 simple_run.6475663
17:31:54.816440 openat F=6 (R******\_\_******) [4]/System/Library/dyld 0.000024 simple_run.6475
17:31:54.816444 fcntl F=6 <getpath> 0.000003 simple_run.6475663
17:31:54.816451 close F=3 0.000003 simple_run.6475663
17:31:54.816452 close F=5 0.000001 simple_run.6475663
17:31:54.816454 close F=4 0.000001 simple_run.6475663
```

POLYTRACE PIPELINE

→ **fs_usage output** → **PROV output**

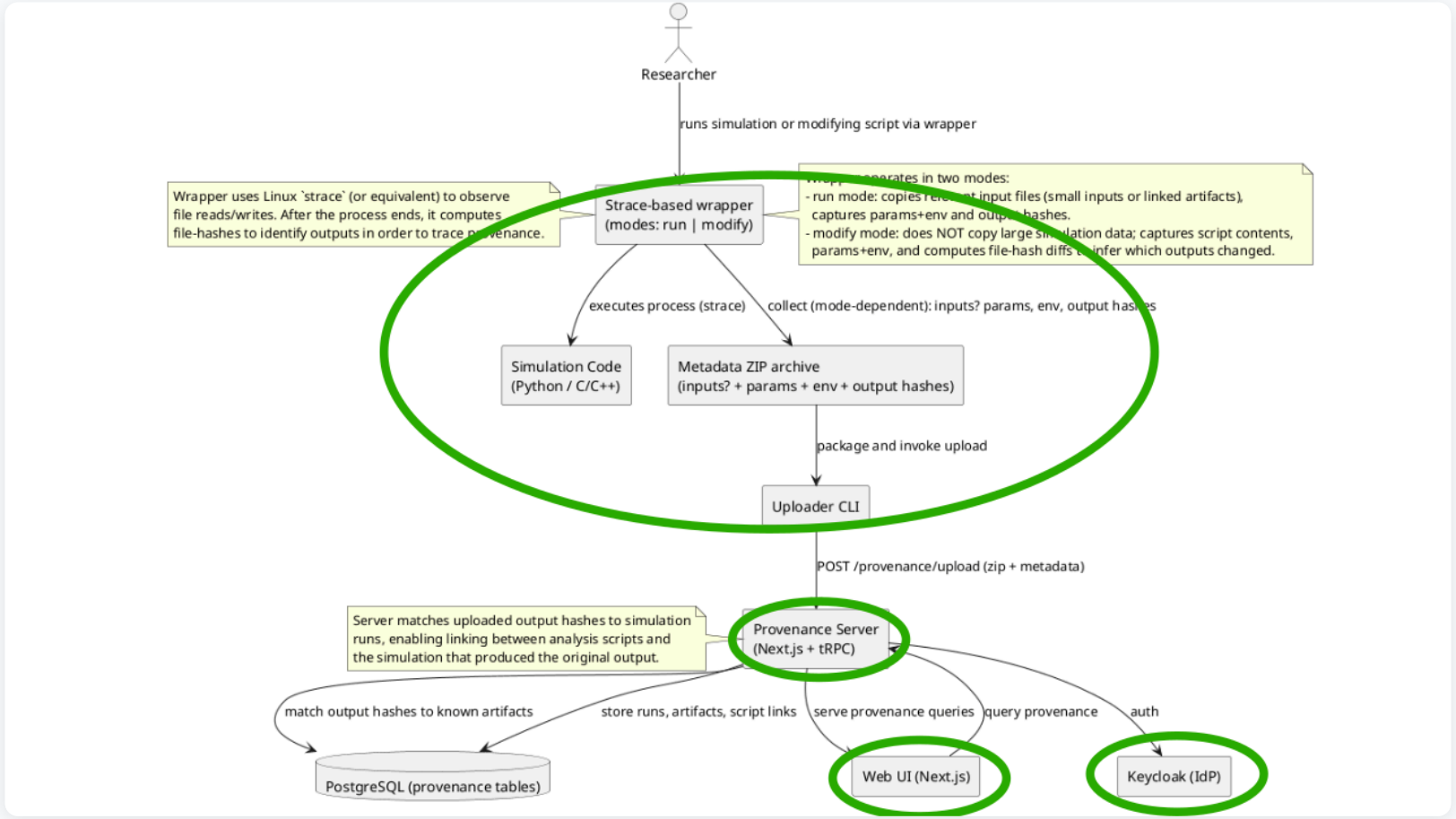
```
17:31:54.814998 fsgetpath /usr/lib/dyld 0.000038 simple_run.6475663
17:31:54.815776 fsgetpath /usr/lib/dyld 0.000019 simple_run.6475663
17:31:54.816178 open F=3 (R******\_\_******) . 0.000098 simple_run.6475663
17:31:54.816182 fcntl F=3 <getpath> 0.000004 simple_run.6475663
17:31:54.816187 close F=3 0.000005 simple_run.6475663
17:31:54.816197 fsgetpath /Users/efemyuksel/PolySim/tools/polytrace/test/fixtures/simple_run 0.
17:31:54.816213 fsgetpath /usr/lib/dyld 0.000015 simple_run.6475663
17:31:54.816345 open F=3 (R****\_\_\_****f**) 0.000030 simple_run.6475663
17:31:54.816396 openat F=4 (R****\_\_\_\_******) [3]/../../System/Volumes/Preboot/Cryptexes/OS
17:31:54.816415 fstatat64 [4]/System/Library/dyld 0.000012 simple_run.6475663
17:31:54.816440 openat F=6 (R******\_\_******) [4]/System/Library/dyld 0.000024 simple_run.6475
17:31:54.816444 fcntl F=6 <getpath> 0.000003 simple_run.6475663
17:31:54.816451 close F=3 0.000003 simple_run.6475663
17:31:54.816452 close F=5 0.000001 simple_run.6475663
17:31:54.816454 close F=4 0.000001 simple_run.6475663
```


```
{
  "activity": {
    "endedAt": 1769725453183,
    "id": "cf843303e1d1269de4c3155fae1b8d1fee2bf84f4bde84b1f004e4784a7b9458",
    "label": "Run /bin/sh",
    "metadata": {
      "command": [
        "/bin/sh",
        "-c",
        "/home/tobias/Projekte/PolySim/tools/polytrace/test/fixtures/simple_run.sh"
      ]
    },
    "startedAt": 1769725453141
  },
```


Polysim

UPLOADER

```
➜  prov_upload_input git:(240-presentation-preparation) ../../../../build/bin/upload --setup
PolySim Upload Tool

=== Configuring PolySim Upload ===

=== PolySim Upload Configuration Setup ===
This wizard will help you configure the upload settings.

Enter upload URL. Skip it to set default URL. (default: https://polysim.work):

Default URL has been set.

Enter Personal Access Token (PAT):


Configuration saved to: "/home/tobias/.config/polysim/config.json" (user read/write only)
➜  prov_upload_input git:(240-presentation-preparation) ../../../../build/bin/upload fork_exec.json
PolySim Upload Tool

Using configuration:
  host: polysim.work
  port: 443
  basePath: /api/public
  use_ssl: true
Using payload argument: fork_exec.json
Uploading activity to server...
Success: {
  "counts": {
    "activities": 1,
    "entities": 3,
    "used": 1,
    "wasAssociatedWith": 0,
    "wasGeneratedBy": 1,
```


Researcher
runs simulation or modifying script via wrapper
Wrapper uses Linux `strace` (or equivalent) to observe file reads/writes. After the process ends, it computes file-hashes to identify outputs in order to trace provenance.
Strace-based wrapper (modes: run | modify)
- run mode: copies relevant input files (small inputs or linked artifacts), captures params+env and output hashes.
- modify mode: does NOT copy large simulation data; captures script contents, params+env, and computes file-hash diffs to infer which outputs changed.
executes process (strace)
collect (mode-dependent): inputs? params, env, output hashes
Simulation Code (Python / C/C++)
Metadata ZIP archive (inputs? + params + env + output hashes)
package and invoke upload
Uploader CLI
POST /provenance/upload (zip + metadata)
Server matches uploaded output hashes to simulation runs, enabling linking between analysis scripts and the simulation that produced the original output.
Provenance Server (Next.js + tRPC)
match output hashes to known artifacts
store runs, artifacts, script links
serve provenance queries
query provenance
auth
PostgreSQL (provenance tables)
Web UI (Next.js)
Keycloak (IdP)

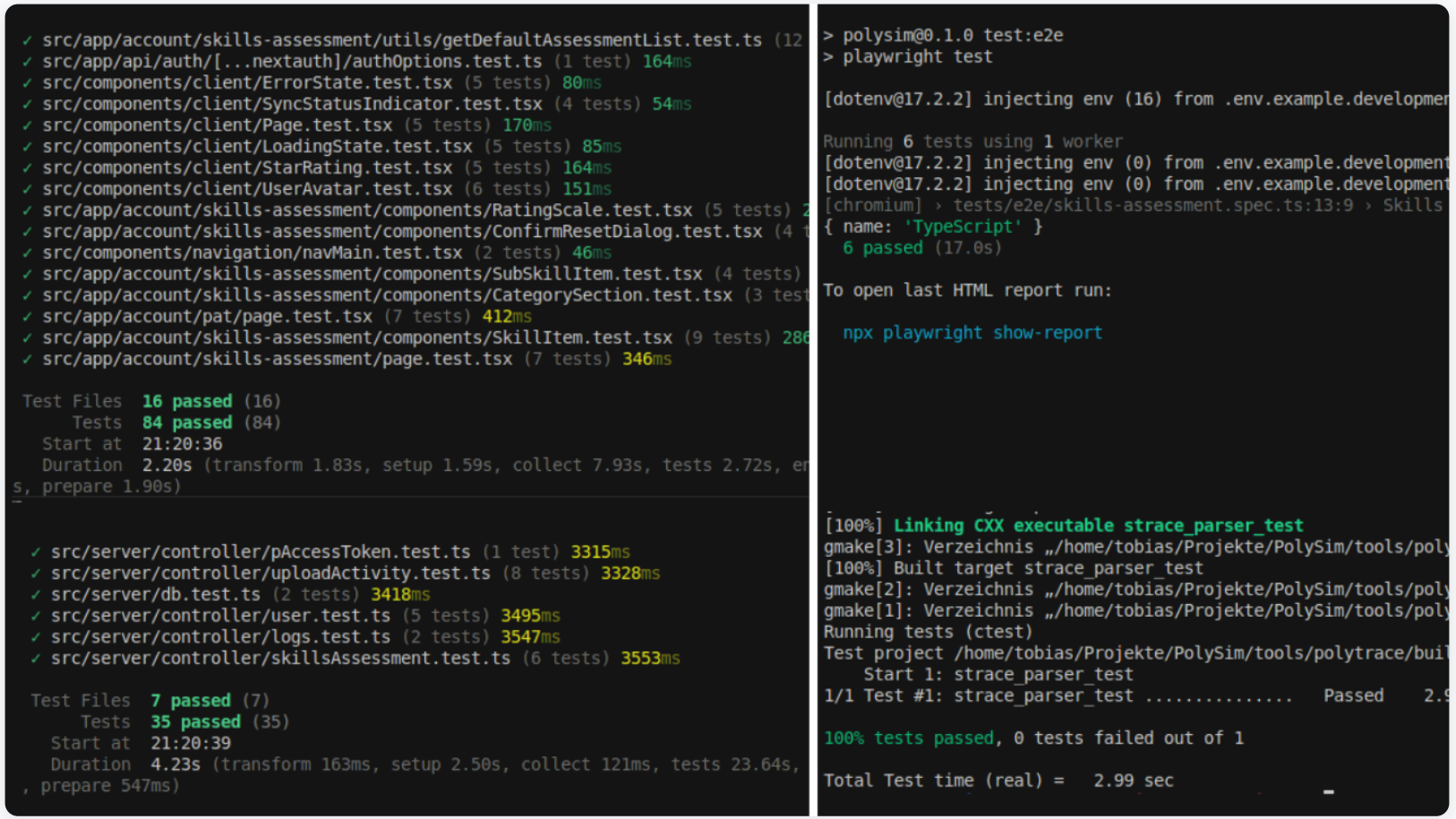
POLYSIM

- Full-stack web application (2 servers)
- CLI tracing tool (PolyTrace)
- integration with Keycloak Auth

GOOD SOFTWARE DEVELOPER PRACTICE

well **tested** code

Does it work as intended?
What if someone change something that they didnt wrote?
What if we break something unrelated with our changes?


✓ src/app/account/skills-assessment/utils/getDefaultAssessmentList.test.ts (12
✓ src/app/api/auth/[...nextauth]/authOptions.test.ts (1 test) 164ms
✓ src/components/client/ErrorState.test.tsx (5 tests) 80ms
✓ src/components/client/SyncStatusIndicator.test.tsx (4 tests) 54ms
✓ src/components/client/Page.test.tsx (5 tests) 170ms
✓ src/components/client/LoadingState.test.tsx (5 tests) 85ms
✓ src/components/client/StarRating.test.tsx (5 tests) 164ms
✓ src/components/client/UserAvatar.test.tsx (6 tests) 151ms
✓ src/app/account/skills-assessment/components/RatingScale.test.tsx (5 tests)
✓ src/app/account/skills-assessment/components/ConfirmResetDialog.test.tsx (4
✓ src/components/navigation/navMain.test.tsx (2 tests) 46ms
✓ src/app/account/skills-assessment/components/SubSkillItem.test.tsx (4 tests)
✓ src/app/account/skills-assessment/components/CategorySection.test.tsx (3 test
✓ src/app/account/pat/page.test.tsx (7 tests) 412ms
✓ src/app/account/skills-assessment/components/SkillItem.test.tsx (9 tests) 286
✓ src/app/account/skills-assessment/page.test.tsx (7 tests) 346ms
Test Files 16 passed (16)
Tests 84 passed (84)
Start at 21:20:36
Duration 2.20s (transform 1.83s, setup 1.59s, collect 7.93s, tests 2.72s, en
s, prepare 1.90s)
✓ src/server/controller/pAccessToken.test.ts (1 test) 3315ms
✓ src/server/controller/uploadActivity.test.ts (8 tests) 3328ms
✓ src/server/db.test.ts (2 tests) 3418ms
✓ src/server/controller/user.test.ts (5 tests) 3495ms
✓ src/server/controller/logs.test.ts (2 tests) 3547ms
✓ src/server/controller/skillsAssessment.test.ts (6 tests) 3553ms
Test Files 7 passed (7)
Tests 35 passed (35)
Start at 21:20:39
Duration 4.23s (transform 163ms, setup 2.50s, collect 121ms, tests 23.64s,
, prepare 547ms)
> polysim@0.1.0 test:e2e
> playwright test
[dotenv@17.2.2] injecting env (16) from .env.example.developmer
Running 6 tests using 1 worker
[dotenv@17.2.2] injecting env (0) from .env.example.development
[dotenv@17.2.2] injecting env (0) from .env.example.development
[chromium] › tests/e2e/skills-assessment.spec.ts:13:9 › Skills
{ name: 'TypeScript' }
6 passed (17.0s)
To open last HTML report run:
npx playwright show-report
[100%] Linking CXX executable strace_parser_test
gmake[3]: Verzeichnis „/home/tobias/Projekte/PolySim/tools/poly
[100%] Built target strace_parser_test
gmake[2]: Verzeichnis „/home/tobias/Projekte/PolySim/tools/poly
gmake[1]: Verzeichnis „/home/tobias/Projekte/PolySim/tools/poly
Running tests (ctest)
Test project /home/tobias/Projekte/PolySim/tools/polytrace/buil
Start 1: strace_parser_test
1/1 Test #1: strace_parser_test ..... Passed 2.9
100% tests passed, 0 tests failed out of 1
Total Test time (real) = 2.99 sec

CONCLUSION:

-Self-Organized Teamwork: Process optimization through regular Retrospectives (transition from Scrum to Scrumban).

-Industry Standards: Integrated Git Workflows, mandatory Code Reviews, and automated CI/CD pipelines.

-FAIR & Provenance: Built a system to track scientific data origin, ensuring reproducibility in research.

-Architectural Clarity: Managed high complexity using

CURRENT STATE OF THE PROJECT

Working PoC: Functional "Vertical Slice" from CLI-tracing to Database and UI-visualization

Infrastructure: Full-stack integration with Docker, Keycloak Auth, and tRPC type-safety

LESSONS LEARNED

Complexity Management: Balancing feature-richness with core logic

Team Coordination: Overcoming communication bottlenecks

Aligning work in a non-full-time environment.

Process Realism: Adapting workflows to fit actual student time-constraints

SUMMARY: A MULTI-LAYERED COURSE

Team Dynamics: Collaborative development on a complex, real-world project

Modern Tech Stack: Hands-on experience with Next.js, PostgreSQL, and Type-safe APIs

Scientific Value: Bridging the gap between software engineering and FAIR simulation principles

End - Questions?

Left intentionally blank

Currently deployed at

https://polysim.work

Username: demo.user

Password: tu-dortmund