NWO ICT.OPEN 2026 — THE FUTURE OF ICT EDUCATION

A Student Developer Platform for IT Classrooms

Continuous delivery as a feedback mechanism

Lennart Tange, MSc · **Christian Salz, MSc** · **Martijn Bonajo, MSc**
Fontys University of Applied Sciences, Venlo

The Problem

A 2024/2025 study of informatics students at Fontys Venlo observed that during their 7th semester projects, students spent a disproportionate amount of time on analysis and design artefacts, mostly because they thought that was expected of them. However, until there is an implementation, its value remains theoretical, delaying feedback and reducing the effectiveness of the learning process.
Furthermore, they lacked supporting infrastructure to create feedback opportunities from (non-IT) customers during their project.

Observed: Theoretical project value

Analysis & Design docs
Coding
Deploy?

Delivery postponed or omitted entirely.

Goal: Continuous Delivery

build feedback

Break down complexity into manageable pieces. Iterative feedback from day one.

- Limited working software was delivered during the project;
- Integration, iterative feedback, and customer involvement were largely missed;
- Delivery and operation postponed until late or omitted entirely;
- Reduced exposure to real-world quality and deployment challenges.

The Approach

To address this, we built an **Internal Developer Platform**, inspired by the industry pattern where a platform team[3] provides capabilities and guardrails that reduce cognitive load for product development teams.
The platform serves educational roles:

Continuous Delivery as Feedback

Every code change (`git push`) triggers build, test, and deploy, creating mechanical feedback loops from day one, allowing mistakes and stimulating rapid iteration.

Authentic Learning

Industry practices (**Agile, DevOps, CI/CD**) brought directly into the classroom.
Students experience what continuous delivery means before studying the underlying theory and technology.
In our educational framework **HBO-i**[1], this helps fill current gaps in exposure to ‘Manage & Control’ and ‘Infrastructure’ activities.

We are piloting it in **PRJ2** (project 2, 1st year, 2nd semester) as a first deployment, which consists of predefined architecture with a backend, frontend, and database.

The Platform

Students deploy from day one with a single `git push`. No Kubernetes knowledge required. Each team gets an isolated namespace, database, public URL, and Grafana dashboard.

Code Change
Student pushes to `main` branch
GitHub

↓

Test & Build
Automated build creates a container image
GitHub Actions

↓

Image Stored
Versioned container image is stored
Harbor

↓

Version Updated
New version detected in infra repository
GitHub + ArgoCD Image Updater

↓

Deployment
ArgoCD syncs new version to the cluster
ArgoCD (GitOps) Kubernetes

↓

Your App is Live!
Accessible at *.prod.fontysvenlo.dev

Supporting Tools

Dashboards
Per-team health, resources, and errors
Grafana

Documentation
Student-facing and admin runbooks
MkDocs .md

Infrastructure-as-Code
Provisioning, configuration, and deployment
Kubernetes CloudStack Bash

Identity Management
Authorization and authentication, outside of the cluster
Keycloak

Access dashboards, documentation and applications yourselves!
grafana.prod.fontysvenlo.dev

Student Onboarding

1. Accept GitHub Classroom link from Canvas
2. On the student dashboard, use your team name to find your links
3. ***Create beautiful software!***

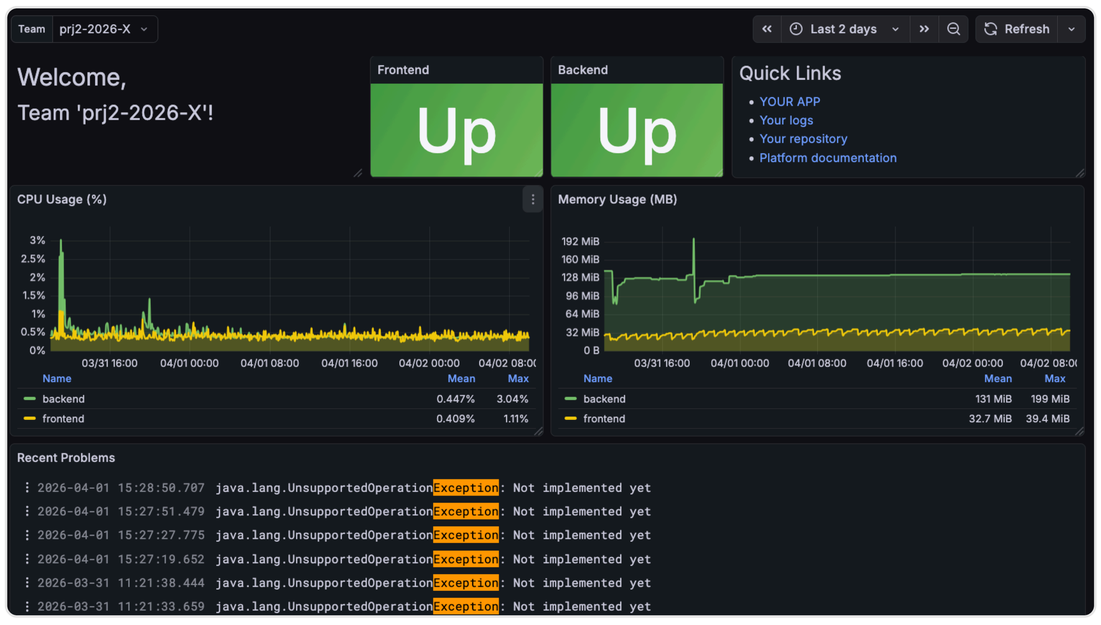

Fig. 1: *Each team gets a live dashboard with application health, resource usage, and recent errors.*

Three Feedback Loops

Every deployment becomes a learning opportunity:

Stakeholder

Coaches and users evaluate working software early and often.

“Am I building the right thing?”

Technical

Automated builds, tests, and deployments surface defects immediately.

“Does my code actually work?”

Process

Integration frequency and collaboration patterns become visible.

“Are we working effectively?”

Grounded in **Minimum Continuous Delivery**[5] principles: trunk-based development, automated builds, and deployment on every push.

From Industry to Education

The platform is designed using a **Developer Experience** (DevEx)[2] framework: optimizing **feedback loops**, minimizing extraneous **cognitive load**, and supporting **flow state**. In an educational context, good developer experience *is* good student experience: the same conditions that help developers do their best work help students learn most effectively.

Where industry optimizes **productivity**, we optimize **learning**:

- The “golden path” is not just for efficiency: it acts as **scaffolding**[6], supporting students at the boundary of what they can do independently;
- DORA[4] metrics like deployment frequency become a **proxy for feedback** rather than a performance target: a signal for coaching intervention;
- The platform hides infrastructure complexity so students focus on **the learning goals**, not on figuring out how to deploy.

Early Observations

As the semester is still running, we are comparing the first 7 weeks of the project to last year’s cohort:

736
Total deployments across all **14 teams** (75 students).
The spread between teams is 26 and 89 total deployments.

+15%
students contributing code to ‘main’.
More students are actively and visibly participating in the development process.

x5
more code reviews.
Based on the number of (merged) pull requests, integration happens more often, branches have a shorter lifecycle.

From coaching, we observed a natural progression towards minimal continuous delivery practices[5]:

- A shift towards more collaboration: planning, integrating, and discussing quality of work;
- Smaller batches of work are being planned;
- Live applications used in demos make the ‘definition of done’ more tangible;
- Students are not yet using their dashboards and environments often themselves.

While intended for education, the platform has also onboarded an applied research project, being able to continuously deliver our research applications with project partners.

Conclusions & Future Work

> *The challenges observed in semester 7 suggest these skills need to be embedded earlier. By introducing Continuous Delivery in year 1, students develop iterative working habits from the start. The early data supports this.*

In informal feedback, both lecturers and students from previous cohorts indicated they would have benefited from such a platform during their own projects.

As a strategic curricular decision, this platform will be embedded across all four 1st-year projects in the next academic year.

To support different learning journeys, also after the first year, we plan to evolve the platform:

Tier 1 — Golden Path (now)

Zero-config deployment. Platform handles infrastructure and guardrails.

- Pre-provisioned pipeline, namespace, database, URL and ready-made Grafana Dashboards;
- Desired extensions: *feedback capabilities from code quality and security scanning tools, DORA[4]-inspired scorecards.*

Tier 2 — Student-Managed (future)

Progressive handover of **Manage & Control** and **Infrastructure** responsibilities (HBO-i[1]):

- Define own delivery pipelines and quality gates;
- Configure monitoring & alerts;
- Choose frameworks, runtime, and/or database.
- Requires more access to the different tools (authentication and authorization).

Fontys University of Applied Sciences | Venlo, The Netherlands
Contact: **l.tange@fontys.nl**

[1] HBO-i. (2024). *HBO-i Domain Model for ICT.* hbo-i.nl
[2] Noda, M. et al. (2023). DevEx: What actually drives productivity. *ACM Queue, 21*(2).
[3] Skelton, M. & Pais, M. (2019). *Team Topologies.* IT Revolution Press.
[4] Forsgren, N. et al. (2018). *Accelerate.* IT Revolution Press.
[5] Finster, B. et al. *Minimum Viable Continuous Delivery.* minimumcd.org (accessed April 2026).
[6] Vygotsky, L. S. (1978). *Mind in Society.* Harvard University Press.