Università degli Studi di Messina, Italia
Dipartimento di scienze matematiche e informatiche,
scienze fisiche e scienze della terra

Soggetto: proposta di tirocinio

ID	PTI_IT_distefano salvatore_21/02/2026 11.21.32
Data	21/02/2026 11.21.32

Supervisore del progetto

Cognome	distefano
Nome	salvatore
Dipartimento	mift
Laboratorio	HPCA
E-mail	sdistefano@unime.it
Numero di telefono	

Co-Supervisore del progetto

Cognome	
Nome	
Posizione	
Dipartimento	

Laboratorio	
E-mail	
Numero di telefono	

Dettagli del progetto

Titolo	Antifragility & Chaos Engineering for Resilient Socio-Technical Systems

Descrizione dettagliata:
Executive Summary

Build systems that don’t just survive turbulence—they *learn* from it and get better. In this internship you’ll practice **chaos engineering** (safe, controlled failure injection) and **SRE** techniques (SLOs, error budgets, golden signals) to transform a cloud application from merely *resilient* to **antifragile**. You’ll design experiments, run them on Kubernetes with open-source tools, read the data, and then harden the system using real reliability patterns—circuit breakers, adaptive timeouts, and even **request hedging** to cut tail latency. Expect hands-on impact, publishable results, and skills that are immediately valuable in modern engineering teams.
[\[arxiv.org\]](https://arxiv.org/pdf/1702.05843), [\[sre.google\]](https://sre.google/books/), [\[en.wikipedia.org\]](https://en.wikipedia.org/wiki/Antifragile_%28book%29)

***

i) Problem, Context & Scenario

The problem. Modern services run on distributed, cloud-native stacks where **rare, long-tail slowdowns** and partial failures dominate user experience at scale; traditional tests miss these behaviors. We need disciplined ways to expose weaknesses *before* users do, and to turn incidents into lasting improvements.
[\[barroso.org\]](https://www.barroso.org/publications/TheTailAtScale.pdf), [\[research.google\]](https://research.google/pubs/the-tail-at-scale/)

Context.

* **Chaos engineering** formalizes *experimentation under failure* to build confidence in production behavior, a practice pioneered and documented by Netflix and now broadly adopted. [\[arxiv.org\]](https://arxiv.org/pdf/1702.05843), [\[api.pageplace.de\]](https://api.pageplace.de/preview/DT0400.9781492043836_A49444509/preview-9781492043836_A49444509.pdf)
* **SRE** reframes reliability as an engineering problem using **SLIs/SLOs**, **error budgets**, and **golden signals** to guide decisions and measure user-visible health. [\[sre.google\]](https://sre.google/books/), [\[sre.google\]](https://sre.google/sre-book/monitoring-distributed-systems/)
* **Antifragility** provides the design lens: create **convex payoffs** to variability so shocks produce learning and capability gains over time. [\[en.wikipedia.org\]](https://en.wikipedia.org/wiki/Antifragile_%28book%29), [\[fooledbyra...omness.com\]](https://fooledbyrandomness.com/ConvexityScience.pdf)

Scenario (what you'll work on). You'll instrument a small microservice app (e.g., "catalog–orders–payments") on Kubernetes, define SLOs for a critical journey (checkout), and run **controlled chaos** (latency injection, pod kills, partitions). You'll then **measure**, **fix**, and **re-test** until the system demonstrably handles turbulence better than before. [\[docs.litmuschaos.io\]](https://docs.litmuschaos.io/), [\[chaos-mesh.org\]](https://chaos-mesh.org/docs/)

***

ii) Aims & Goals

1. **Make reliability measurable.** Define SLIs/SLOs, dashboards, and an **error-budget policy** that governs when to slow changes versus when to ship. [\[sre.google\]](https://sre.google/books/), [\[sre.google\]](https://sre.google/workbook/error-budget-policy/)
2. **Practice safe-to-fail experiments.** Design and execute 3–5 chaos experiments with explicit hypotheses, blast-radius limits, and abort conditions. [\[api.pageplace.de\]](https://api.pageplace.de/preview/DT0400.9781492043836_A49444509/preview-9781492043836_A49444509.pdf), [\[docs.litmuschaos.io\]](https://docs.litmuschaos.io/)
3. **Engineer resilience patterns.** Add **circuit breakers, retries with jitter, bulkheads, adaptive timeouts** (e.g., with Resilience4j) and validate the effect on SLOs. [\[resilience....readme.io\]](https://resilience4j.readme.io/docs/circuitbreaker)
4. **Tame tail latency.** Apply **request hedging** and adaptive thresholds to reduce p95/p99 without over-provisioning, grounded in *The Tail at Scale*. [\[barroso.org\]](https://www.barroso.org/publications/TheTailAtScale.pdf)

5. **Close the learning loop.** Convert findings into code/config changes and a repeatable “game day” plan—building the habit of antifragile improvement. [\[api.pageplace.de\]](https://api.pageplace.de/preview/DT0400.9781492043836_A49444509/preview-9781492043836_A49444509.pdf)

***

iii) Learning Outcomes

By the end, you will be able to:

* **Explain and apply** *robust vs. resilient vs. antifragile* with the math intuition of convex responses (Jensen’s inequality). [\[en.wikipedia.org\]](https://en.wikipedia.org/wiki/Antifragile_%28book%29), [\[fooledbyra...omness.com\]](https://fooledbyrandomness.com/ConvexityScience.pdf)
* **Design SLOs** tied to user journeys and manage **error budgets** to balance feature velocity with stability. [\[sre.google\]](https://sre.google/books/), [\[sre.google\]](https://sre.google/workbook/error-budget-policy/)
* **Run chaos safely** on Kubernetes with **LitmusChaos** or **Chaos Mesh** and interpret the impact using golden signals. [\[docs.litmuschaos.io\]](https://docs.litmuschaos.io/), [\[chaos-mesh.org\]](https://chaos-mesh.org/docs/)
* **Implement resilience** with **Resilience4j** (circuit breakers, retries, timeouts, bulkheads) and verify improvements via re-experimentation. [\[resilience....readme.io\]](https://resilience4j.readme.io/docs/circuitbreaker)
* **Reduce tail latency** using **request hedging** and adaptive timeouts, justifying trade-offs with data. [\[barroso.org\]](https://www.barroso.org/publications/TheTailAtScale.pdf)

***

iv) Roadmap (100–150 hours)

> *Paced for \~5–8 weeks part-time; adjust depth by adding experiments or extending analysis.*

Phase 1 — Foundations & Setup (15–25h).
Short seminars on antifragility, chaos engineering and SRE; set up a Kubernetes sandbox with Prometheus/Grafana; deploy the demo app. [\[en.wikipedia.org\]](https://en.wikipedia.org/wiki/Antifragile_%28book%29),

[\[arxiv.org\]](https://arxiv.org/pdf/1702.05843), [\[sre.google\]](https://sre.google/books/)

Phase 2 — Make reliability visible (15–20h).
Define **SLIs/SLOs** and an **error-budget policy**; build dashboards for the **four golden signals** (latency, traffic, errors, saturation).
[\[sre.google\]](https://sre.google/workbook/error-budget-policy/), [\[sre.google\]](https://sre.google/sre-book/monitoring-distributed-systems/)

Phase 3 — Design experiments (20–25h).
Write 3–5 experiment specs (hypothesis, blast radius, abort rules): dependency **latency**, **network partition**, **pod kill/CPU stress**, **time skew**.
[\[docs.litmuschaos.io\]](https://docs.litmuschaos.io/), [\[chaos-mesh.org\]](https://chaos-mesh.org/docs/)

Phase 4 — Run & harden (35–55h).
Execute experiments, analyze SLO impact; add **Resilience4j** patterns (timeouts, RBJ retries, circuit breakers, bulkheads) and re-run tests to verify improvement.
[\[resilience....readme.io\]](https://resilience4j.readme.io/docs/circuitbreaker)

Phase 5 — Tackle tail latency (15–25h).
Introduce **request hedging** (duplicate after p95-based delay; cancel the loser) and adaptive timeouts; quantify p95/p99 gains and cost.
[\[barroso.org\]](https://www.barroso.org/publications/TheTailAtScale.pdf)

Deliverables: SLO pack (dashboards + policy), chaos portfolio (specs + results), resilience diffs (code/config), tail-latency brief, 10–15 page report + short talk.
[\[sre.google\]](https://sre.google/workbook/error-budget-policy/), [\[docs.litmuschaos.io\]](https://docs.litmuschaos.io/)

***

v) Scientific References (publications & tools)

Core publications (theory & practice)

* **Antifragility**: *Antifragile: Things That Gain from Disorder* — concept & convexity intuition for engineering.
[\[en.wikipedia.org\]](https://en.wikipedia.org/wiki/Antifragile_%28book%29)
* **Convexity formalizations** (Taleb et al.) — mathematical framing of antifragility and dose/response convexity.

[\[fooledbyra...omness.com\]](https://fooledbyrandomness.com/ConvexityScience.pdf), [\[mdpi.com\]](https://www.mdpi.com/1099-4300/25/2/343)
* **Chaos engineering (Netflix, discipline)**: Basiri et al., *IEEE Software*; Rosenthal & Jones, *O'Reilly* — principles, governance, case studies. [\[arxiv.org\]](https://arxiv.org/pdf/1702.05843), [\[api.pageplace.de\]](https://api.pageplace.de/preview/DT0400.9781492043836_A49444509/preview-9781492043836_A49444509.pdf)
* **SRE canon** (free online): **SRE books & workbook** (Google) — SLOs, monitoring, incident response; **error budget policy** exemplar; **Golden Signals** chapter. [\[sre.google\]](https://sre.google/books/), [\[sre.google\]](https://sre.google/workbook/error-budget-policy/), [\[sre.google\]](https://sre.google/sre-book/monitoring-distributed-systems/)
* **Tail latency**: Dean & Barroso, "**The Tail at Scale**" — why p99 matters, hedged requests. [\[barroso.org\]](https://www.barroso.org/publications/TheTailAtScale.pdf), [\[research.google\]](https://research.google/pubs/the-tail-at-scale/)

Tools & documentation (hands-on)

* **LitmusChaos** — CNCF chaos platform with ready-made K8s experiments & workflows. [\[docs.litmuschaos.io\]](https://docs.litmuschaos.io/)
* **Chaos Mesh** — CNCF platform for pod/network/IO/time faults with dashboard & CRDs. [\[chaos-mesh.org\]](https://chaos-mesh.org/docs/)
* **Resilience4j** — Java resilience patterns (circuit breaker, retry, rate limiter, bulkhead, time limiter). [\[resilience....readme.io\]](https://resilience4j.readme.io/docs/circuitbreaker)

***

Why students should join

You'll learn to **think like a reliability engineer**, perform **publishable experiments**, and ship **practical improvements** to a real system. It's an ideal bridge from coursework to the problems engineers solve every day in cloud, data, fintech, and beyond. [\[arxiv.org\]](https://arxiv.org/pdf/1702.05843), [\[sre.google\]](https://sre.google/books/)

Durata (mesi – max 12)	2

Durata (ore)	100
Numero di posizioni aperte	6

Competenze richieste dal tirocinio

Requisiti tecnici:	
Altri requisiti	