

# **Rapport - Projet Triangulator**

Master 1 - Technique de Test

**Caroline HULOT**

17 décembre 2025

# 1 Introduction

Le projet **Triangulator** consiste en la réalisation d'un micro-service de calcul géométrique. Il y a eu la mise en place d'une approche *Test-First*.

## Fonctionnement du Triangulator

1. **Réception** : Récupère un ensemble de points (**PointSet**) identifié par un UUID.
2. **Traitement** : Effectue une triangulation.
3. **Réponse** : Retourne le résultat (**Triangles**) dans un format binaire.

## 2 Architecture du projet

### Structure du répertoire

```
TP/
|-- Triangulator/
|   |-- app.py           # API Flask
|   |-- triangulator.py  # Algorithme de triangulation
|   |-- binary_format.py # serialisation

|-- tests/
|   |-- TestsUnitaires.py
|   |-- TestsRoutes.py
|   |-- TestsPerformance.py
|   |-- TestsRobustesse.py
|   |-- TestsSecurite.py
|   |-- TestsQualite.py
|   |-- mocks.py
```

## 3 Réalisation

### 3.1 Implémentation du Triangulator (triangulator.py)

L'algorithme de triangulation vient d'internet car le but premier du projet étais les tests. Celui que j'ai choisis est performant et marche parfaitement.

## 4 Les Test

Au total, **78 tests** verifie le fonctionnement du Triangulator.

### 4.1 1. Tests Unitaires (24 tests)

Validation de la logique pure, répartie entre `TestsUnitaires.py` (22 tests) et `TestsAutres.py` (2 tests).

- **Algorithmique** : Vérification des cas standards, (1 triangle, carré,...) et des cas dégénérés (points alignés → 0 triangle, points dupliqués,...)

- **Binaire** : Vérification de la conversion (`PointSet ↔ Binaire`).
- **Limites** : Gestion des valeurs `Nan`, `Inf` et précision des coordonnées.

## 4.2 2. Tests API et Routes (11 tests)

Validation des routes HTTP dans `TestsRoutes.py`.

- Vérification des codes de retour (200, 201, 400, 404).
- Validation des types de contenu et des structures de réponse.

## 4.3 3. Robustesse et Sécurité (19 tests)

Tests offensifs pour vérifier la résilience du système.

- **Robustesse (14 tests)** : Résistance aux données corrompues, timeouts et pannes simulées.
- **Sécurité (5 tests)** : Protection contre les injections (SQL, Shell), DoS et ID malveillants.

## 4.4 4. Intégration et Système (16 tests)

Validation des interactions et du flux complet (`TestsIntegrations.py`, `TestsSysteme.py`, `test-triangulator-interactive.py`).

## 4.5 5. Performance (4 tests)

Validation des temps de réponse et benchmarks.

NB Points	Temps Moyen	NB Triangles
100	~ 0.0007 s	98
1 000	~ 0.0613 s	998
5 000	~ 1.8589 s	4 998
10 000	~ 7.2022 s	9 998
50 000	~ 218.6335 s	49 998
100 000	~ 957.3221 s	99 998

TABLE 1 – Benchmark de la triangulation

# 5 Résultats

## 5.1 Tableau de Synthèse

Composant	Statut	Détails
Tests	100% Passés	<b>78 tests</b> exécutés avec succès.
Qualité	Conforme	<code>ruff</code> (linting) et <code>coverage</code> (93%).
Documentation	Générée	HTML disponible via <code>make doc</code> .

## 5.2 Analyse des Écarts (Implémentation Test vs Réalité)

Il y a une légère différence entre les tests que je voulais faire et les tests finaux.

Section	Voulu	Exécuté	Justification de l'écart
Tests Unitaires	31	24	Regroupement de cas.
Tests API	16	11	Consolidation des tests de validation de payload.
Tests Robustesse	15	14	Fusion des tests sur les exceptions internes.
Tests Sécurité	5	5	Pareil.
Tests Intégration	9	9	Pareil.
Tests Système	7	7	Pareil.
Performance	4	4	Pareil.
Qualité	4	4	Pareil.
<b>TOTAL</b>	<b>91 (théorique)</b>	<b>78 (réel)</b>	Optimisation.

TABLE 2 – Comparatif Planification vs Exécution

## 6 Comparaison : Plan Initial vs l'Implémentation

Le tableau montre les zones où l'implémentation a changé par rapport au plan de départ.

Axe	Plan Initial (Théorique)	Rapport d'implémentation	Réalité (Implémenté)
<b>Nombres</b>	50 tests estimés	81 tests annoncés	<b>78 tests exécutés</b>
<b>Unitaires</b>	Algo de base	31 tests (avec doublons logiques)	<b>24 tests optimisés</b> (Regroupement des cas limites)
<b>Performance</b>	"Mesurer temps"	4 tests	<b>4 tests + Benchmark</b> séparé
<b>Sécurité</b>	"ID trafiqué"	5 tests (SQL, DoS)	<b>5 tests</b>
<b>API / Routes</b>	Codes HTTP classiques	16 tests	<b>16 tests</b>
<b>Couverture</b>	Obj. > 90%	100%	<b>93%</b> de couverture de code réelle

TABLE 3 – Évolution des test

## 7 Automatisation et Outilage (Makefile)

On peut tout faire avec le `Makefile`.

### 7.1 Commandes Principales

- `make test` : Lance tout les tests (Unitaires, Intégration, Performance,...).

```

TP/tests/TestEdgeCases.py::TestEdgeCases::test_single_point PASSED [ 1%]
TP/tests/TestEdgeCases.py::TestEdgeCases::test_two_points PASSED [ 2%]
TP/tests/TestIntegrations.py::TestIntegrationWithMocks::test_full_workflow_with_mock_pointset_manager PASSED [ 3%]
TP/tests/TestIntegrations.py::TestIntegrationWithMocks::test_triangulator_pointset_manager_integration PASSED [ 5%]
TP/tests/TestIntegrations.py::TestIntegrationWithMocks::test_integration_nonexistent_id_error PASSED [ 6%]
TP/tests/TestIntegrations.py::TestIntegrationWithMocks::test_integration_service_unavailable PASSED [ 7%]
TP/tests/TestIntegrations.py::TestIntegrationWithMocks::test_integration_invalid_format PASSED [ 8%]
TP/tests/TestIntegrations.py::TestIntegrationWithMocks::test_pointset_manager_returns_invalid_format PASSED [ 10%]
TP/tests/TestIntegrations.py::TestIntegrationWithMocks::test_pointset_manager_returns_no_data PASSED [ 13%]
TP/tests/TestIntegrations.py::TestIntegrationWithMocks::test_multiple_concurrent_triangulations PASSED [ 15%]
TP/tests/TestIntegrations.py::TestIntegrationWithMocks::test_integration_timeout_handling PASSED [ 16%]
TP/tests/TestPerformance.py::TestPerformance::test_larger_pointset_triangulation PASSED [ 15%]
TP/tests/TestPerformance.py::TestPerformance::test_repeated_requests_performance PASSED [ 16%]
TP/tests/TestPerformance.py::TestPerformance::test_memory_usage PASSED [ 17%]
TP/tests/TestPerformance.py::TestPerformance::test_binary_conversion_performance PASSED [ 19%]
TP/tests/TestQualite.py::TestCodeQuality::test_ruff_check_passes PASSED [ 20%]
TP/tests/TestQualite.py::TestCodeQuality::test_coverage_threshold PASSED [ 21%]
TP/tests/TestQualite.py::TestCodeQuality::test_documentation_threshold PASSED [ 23%]
TP/tests/TestRobustesse.py::TestRobustesse::test_all_make_targets PASSED [ 24%]
TP/tests/TestRobustesse.py::TestRobustesse::test_truncated_pointset PASSED [ 25%]
TP/tests/TestRobustesse.py::TestRobustesse::test_non_float_coordinates PASSED [ 26%]
TP/tests/TestRobustesse.py::TestRobustesse::test_unreachable_service PASSED [ 28%]
TP/tests/TestRobustesse.py::TestRobustesse::test_empty_result PASSED [ 29%]
TP/tests/TestRobustesse.py::TestRobustesse::test_internal_exception_handling PASSED [ 30%]
TP/tests/TestRobustesse.py::TestRobustesse::test_time_limit_handling PASSED [ 32%]
TP/tests/TestRobustesse.py::TestRobustesse::test_negative_parallelism_robustness PASSED [ 33%]
TP/tests/TestRobustesse.py::TestRobustesse::test_bound_float_limits PASSED [ 34%]
TP/tests/TestRobustesse.py::TestRobustesse::test_random_corrupted_data PASSED [ 35%]
TP/tests/TestRobustesse.py::TestRobustesse::test_pointset_with_single_point PASSED [ 37%]
TP/tests/TestRobustesse.py::TestRobustesse::test_pointset_with_two_points PASSED [ 38%]
TP/tests/TestRobustesse.py::TestRobustesse::test_database_connection_failure PASSED [ 39%]
TP/tests/TestRobustesse.py::TestRobustesse::test_pointset_format_mismatch PASSED [ 41%]
TP/tests/TestRobustesse.py::TestRobustesse::test_pointset_out_of_range_indices PASSED [ 42%]
TP/tests/TestRobustesse.py::TestRobustesse::test_pointset_endpoint_too_few_points_in_request PASSED [ 43%]
TP/tests/TestRoutes.py::TestTriangulationEndpoint::test_nonexistent_id PASSED [ 44%]
TP/tests/TestRoutes.py::TestTriangulationEndpoint::test_malformed_id PASSED [ 46%]
TP/tests/TestRoutes.py::TestTriangulationEndpoint::test_triangulation_collinear PASSED [ 47%]
TP/tests/TestRoutes.py::TestFromHandling::test_404_bad_request PASSED [ 48%]
TP/tests/TestRoutes.py::TestFromHandling::test_404_not_found PASSED [ 50%]
TP/tests/TestRoutes.py::TestPointSetRegistration::test_post_pointset_success PASSED [ 51%]
TP/tests/TestRoutes.py::TestPointSetRegistration::test_post_pointset_invalid_format PASSED [ 52%]
TP/tests/TestRoutes.py::TestPointSetRegistration::test_get_pointset_empty PASSED [ 53%]
TP/tests/TestRoutes.py::TestPointSetRegistration::test_get_pointset_success PASSED [ 55%]
TP/tests/TestRoutes.py::TestPointSetRegistration::test_get_pointset_not_found PASSED [ 56%]
TP/tests/TestSecurity.py::TestSecurity::test_tampered_id PASSED [ 57%]
TP/tests/TestSecurity.py::TestSecurity::test_oversized_request_dos PASSED [ 58%]
TP/tests/TestSecurity.py::TestSecurity::test_sql_injection_in_id PASSED [ 68%]
TP/tests/TestSecurity.py::TestSecurity::test_oversized_binary_pointset PASSED [ 61%]
TP/tests/TestSecurity.py::TestSecurity::test_command_injection_in_pointset_data PASSED [ 62%]
TP/tests/TestSystem.py::TestWorkflow::test_parallel_workflow PASSED [ 64%]
TP/tests/TestSystem.py::TestWorkflow::test_workflow_with_missing_service PASSED [ 65%]
TP/tests/TestSystem.py::TestSystemWorkflow::test_pointset_manager_fails_midway PASSED [ 66%]
TP/tests/TestSystem.py::TestSystemWorkflow::test_concurrent_triangulation_requests PASSED [ 67%]
TP/tests/TestSystem.py::TestSystemWorkflow::test_workflow_with_empty_pointset PASSED [ 69%]
TP/tests/TestSystem.py::TestSystemWorkflow::test_workflow_with_align_points PASSED [ 70%]
TP/tests/TestSystem.py::TestSystemWorkflow::test_workflow_large_dataset PASSED [ 71%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_three_points_one_triangle PASSED [ 73%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_three_points_no_triangles PASSED [ 74%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_colinear_points_no_triangle PASSED [ 75%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_duplicate_points_rejection PASSED [ 76%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_invalid_input_format PASSED [ 78%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_empty_pointset PASSED [ 79%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_points_with_nan PASSED [ 80%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_points_with_inf PASSED [ 82%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_very_close_points_precision PASSED [ 83%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_maximal_size_2000 PASSED [ 84%]
TP/tests/TestUnitaires.py::TestTriangulationAlgorithm::test_negative_coordinates PASSED [ 87%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_pointset_to_binary PASSED [ 88%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_binary_to_pointset PASSED [ 89%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_triangles_to_binary PASSED [ 91%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_binary_to_triangle PASSED [ 92%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_no_duplicate_vertex_in_triangle PASSED [ 93%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_binary_size_verification PASSED [ 94%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_binary_with_malformed_coordinates PASSED [ 95%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_binary_with_large_pointset PASSED [ 97%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_binary_corrupted_data PASSED [ 98%]
TP/tests/TestUnitaires.py::TestBinaryConversions::test_binary_zero_points PASSED [ 100%]

=====
78 passed in 3.46s =====

```

- **make unit\_test** : Exécute uniquement les tests funitaires.
- **make perf\_test** : Isole les tests de performance et de charge.
- **make benchmark** : Lance le script de benchmark dédié (**benchmark\_triangulation.py**) pour mesurer les temps d'exécution.
- **make lint** applique les règles de style via **ruff** (avec auto-fix).
- **make coverage** génère le rapport HTML complet dans :

TP/htmlcov/index.html.

Name	Stmts	Miss	Cover	Missing
TP/Triangulator/app.py	72	15	79%	56-58, 80, 82, 111, 149-150, 185, 203-204, 231, 243-244, 250
TP/Triangulator/binary_format.py	58	7	88%	97, 153, 195, 201, 209, 219, 222
TP/Triangulator/triangulator.py	70	9	87%	84, 136, 198-213
TP/_init_.py	0	0	100%	
TP/tests/TestAutres.py	12	0	100%	
TP/tests/TestIntegrations.py	76	0	100%	
TP/tests/TestPerformance.py	47	0	100%	
TP/tests/TestQualite.py	28	0	100%	
TP/tests/TestRobustesse.py	84	1	99%	98
TP/tests/TestRoutes.py	56	0	100%	
TP/tests/TestSecurite.py	25	0	100%	
TP/tests/TestSysteme.py	66	0	100%	
TP/tests/TestSystemWorkflow.py	168	6	96%	53-56, 137-139
TP/tests/_init_.py	0	0	100%	
TP/tests/conftest.py	11	0	100%	
TP/tests/mocks.py	82	21	74%	61, 79, 88, 99-100, 113-119, 124-129, 134-136, 141-143, 148, 153, 158
TOTAL	855	59	93%	

Wrote HTML report to TP/htmlcov/index.html  
✓ Rapport de coverage généré dans TP/htmlcov/index.html

FIGURE 1 – make coverage  
labelfig :placeholder

- `make doc` génère la documentation API dans :  
`TP/docs`.
- `make clean` : Supprime tous les artefacts de build, caches python, et rapports générés pour repartir sur une base saine.